SBRC 2008

Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

26 a 30 de maio de 2008, Rio de Janeiro, Brasil

ANAIS DO
WCGA 2008

SBRC 2008 − 26° Simpósio Brasileiro de
Redes de Computadores e Sistemas Distribuídos
Rio de Janeiro, RJ – Brasil
26 a 30 de maio de 2008

VI Workshop de Computação em Grade e Aplicações - WCGA

Editora
Sociedade Brasileira de Computação

Organizadores
Bruno Schulze (LNCC)
Michael Stanton (RNP)
Radha Nandkumar (UIUC)
Alexandre Sztajnberg (UERJ)
Artur Ziviani (LNCC)
Célio Vinicius Neves de Albuquerque (UFF)
Luís Henrique Maciel Kosmalski Costa (UFRJ)
Marcelo Gonçalves Rubinstein (UERJ)

Realização
Laboratório Nacional de Computação Científica (LNCC)
Universidade Federal Fluminense (UFF)
Universidade Federal do Rio de Janeiro (UFRJ)
Universidade do Estado do Rio de Janeiro (UERJ)

Promoção
Sociedade Brasileira de Computação (SBC)
Laboratório Nacional de Redes de Computadores (LARC)

Capa: Angela Jaconianni Carneiro Ribeiro
Produção Editorial: Igor Monteiro Moraes
Miguel Elias Mitre Campista

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca da Escola de Engenharia e do Instituto de Computação - Universidade Federal Fluminense (UFF)

W926 Workshop de Computação em Grade e Aplicações (6. : 2008 : Rio de Janeiro, RJ).
Anais / VI Workshop de Computação em Grade e Aplicações ; organização Bruno Schulze ... [et al.]. - Rio de Janeiro, RJ : Sociedade Brasileira de Computação, 2008.

1 CD-ROM.
Evento realizado no período de 26 de maio a 30 de maio de 2008.
ISBN: 978-85-7669-176-1

1. Redes de computadores - Congresso. 2. Sistema de computação em grade - Congresso. 3. Informática - Congresso. I. Schulze, Bruno, org. II. Título.

CDD 004.6

Promoção

Sociedade Brasileira de Computação (SBC)

Diretoria

Presidência
José Carlos Maldonado (USP)

Vice-Presidência
Virgílio Augusto Fernandes Almeida (UFMG)

Diretoria Administrativa
Carla Maria Dal Sasso Freitas (UFRGS)

Diretoria de Finanças
Paulo Cesar Masiero (USP)

Diretoria de Eventos e Comissões Especiais
Marcelo Walter (UFPE)

Diretoria de Educação
Edson Norberto Cáceres (UFMS)

Diretoria de Publicações
Karin Breitman (PUC-Rio)

Diretoria de Planejamento e Programas Especiais
Augusto Cezar Alves Sampaio (UFPE)

Diretoria de Secretarias Regionais
Aline Maria Santos Andrade (UFBA)

Diretoria de Divulgação e Marketing
Altigran Soares da Silva (UFAM)

Diretoria de Regulamentação da Profissão
Ricardo de Oliveira Anido (UNICAMP)

Diretoria de Eventos Especiais
Carlos Eduardo Ferreira (USP)

Diretoria de Cooperação com Sociedades Científicas
Taisy Silva Weber (UFRGS)

Realização

Comitê de Organização

Coordenação Geral
Artur Ziviani (LNCC)
Célio Vinicius Neves de Albuquerque (UFF)
Luís Henrique Maciel Kosmalski Costa (UFRJ)
Marcelo Gonçalves Rubinstein (UERJ)

Coordenação do Comitê de Programa
Nelson Luis Saldanha da Fonseca (UNICAMP)

Coordenação de Tutoriais
Otto Carlos Muniz Bandeira Duarte (UFRJ)

Coordenação de Minicursos
Luciano Paschoal Gaspary (UFRGS)

Coordenação de Painéis e Debates
Luiz Fernando Gomes Soares (PUC-Rio)

Coordenação do Salão de Ferramentas
Flávia Coimbra Delicato (UFRN)

Coordenação de Workshops
Alexandre Sztajnberg (UERJ)

Comitê Consultivo
Antônio Jorge Gomes Abelém (UFPA)
Carlos Alberto Maziero (PUCPR)
Elias Procópio Duarte Jr. (UFPR)
Keiko Verônica Ono Fonseca (UTFPR)
João Crisóstomo Weyl Costa (UFPA)
Joni da Silva Fraga (UFSC)
Lisandro Zambenedetti Granville (UFRGS)
Luci Pirmez (UFRJ)

Mensagem dos Coordenadores Gerais e do Coordenador de Workshops

O Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) é um evento anual, sem fins lucrativos, promovido pela Sociedade Brasileira de Computação (SBC), através da sua Comissão Especial de Redes de Computadores e Sistemas Distribuídos (CE-RESD), e pelo Laboratório Nacional de Redes de Computadores (LARC). Ao longo dos anos, o SBRC tornou-se o mais importante evento científico nacional em redes de computadores e sistemas distribuídos, e um dos destaques na área de Informática no país. O SBRC é composto atualmente pelo Simpósio Principal e por Workshops.

Este documento contém os artigos selecionados para publicação e apresentação na sexta edição do Workshop de Computação em Grade e Aplicações (WCGA). Os textos cobrem assuntos atuais e relevantes para a área de redes de computadores e sistemas distribuídos, em particular na área de grades.

Gostaríamos de agradecer e parabenizar o Prof. Bruno Schulze pela condução dos trabalhos de seleção dos artigos. Gostaríamos de agradecer também aos membros do Comitê de Programa e aos revisores do WCGA.

Artur Ziviani
Célio Vinicius Neves de Albuquerque
Luís Henrique Maciel Kosmalski Costa
Marcelo Gonçalves Rubinstein
Coordenadores Gerais

Alexandre Sztajnberg
Coordenador de Workshops

Mensagem dos Coordenadores do WCGA

As primeiras edições do Workshop de Computação em Grid e Aplicações - WCGA (2003 - 2004 - 2005), foram realizadas em conjunto com o Programa de Verão do LNCC, em Petrópolis-RJ. As edições de 2006 e 2007 foram realizadas em conjunto com o Simpósio Brasileiro de Redes de Computadores - SBRC em Curitiba - PR e Belém - PA. Estas edições anteriores do WCGA receberam uma combinação interessante de diferentes áreas de computação em *grids* como infra-estrutura, *middleware*, redes e aplicações, com apresentações técnicas e pôsteres fomentando discussões em vários tópicos interessantes, incluindo *grids* clássicos, *grids* baseados em objetos e serviços, escalonamento, e aplicações, tais como bioinformática; meteorologia e governo eletrônico, entre outros. Estas edições geraram um interesse crescente na comunidade e nos baseamos nesta tradição para mais esta edição.

A sexta edição do WCGA está pela terceira vez sendo realizada em conjunto com o SBRC, e tem o objetivo de promover o encontro de pesquisadores no campo de computação em *grids*, abordando problemas de larga escala e do mundo real em ambientes de *grids*, incluindo os tópicos mais interessantes e estimulantes que surgiram em anos anteriores, e outros inéditos. Houve vinte e um (21) trabalhos submetidos sendo onze (11) aceitos como artigos completos e sete (7) convidadas para apresentação como pôster. Os autores apontaram problemas e soluções em um ou mais temas dentre aqueles identificados para o workshop.

Finalmente, a coordenação do workshop gostaria de agradecer a todos os autores e participantes pela apresentação dos trabalhos e contribuições de pesquisa em *grids* computacionais e aplicações, agradecer a todos os membros dos comitês de programa e de organização pela ajuda na elaboração do programa do workshop e nas atividades, e adicionalmente agradecer os apoios de LNCC, RNP, NCSA, e SBC.

Bruno Schulze
Michael Stanton
Radha Nandkumar
Coordenadores do WCGA

Message from the WCGA Chairs

The previous editions of the Workshop on Grid Computing and Applications - WCGA were held in conjunction with the Summer Programme of LNCC, in Petrópolis - RJ, Brazil. The editions of 2006 and 2007 were hold together with the SBRC in Curitiba - PR and Belém - PA. The previous editions of WCGA saw a good combination of the many different flavors of grid computing, in terms of infrastructure, middleware, networking and applications, with several technical presentations and posters generating discussions on several stimulating topics including, Classic Grids, Object and Service-based Grids, Scheduling, and Applications such as Bioinformatics, Meteorology and E-Government, among others. The editions generated increasing interest in the community and we built on this tradition for this year.

The sixth edition of WCGA is being held for the third time in conjunction with the Brazilian Symposium on Computer Networks – SBRC, and has the objective to bring together researches in the field of grid computing, addressing large-scale and real world problems in grid environments, including the most interesting and stimulating topics emerged previous years, and some novel. There were twenty one (21) submissions with eleven (11) papers being selected as full papers and seven (7) invited for poster presentation. The authors did point out issues and solutions in one or more of the themes that had been identified for the workshop.

Finally, the workshop coordination would like to thank all the authors, and participants for presenting their work and contributions on research in computational grids and applications, to thank all the members of the program and organizing committee for helping to shape the workshop program and activities, and additionally to thank the sponsors from LNCC, RNP, NCSA, and SBC.

Bruno Schulze
Michael Stanton
Radha Nandkumar
WCGA Chairs

Comitê de Avaliação

Coordenadores
Bruno Schulze (LNCC)
Michael Stanton (RNP)
Radha Nandkumar (UIUC)

Membros
Alba Melo (UnB)
Alexandre Sztajnberg (UERJ)
Antônio Tadeu A Gomes (LNCC)
Artur Ziviani (LNCC)
Celso Mendes (UIUC)
Cesar De Rose (PUCRS)
Claudio Geyer (UFRGS)
Diego Carvalho (CEFET-RJ)
Edison Ishikawa (IME-RJ)
Edmundo Madeira (UNICAMP)
Fabíola Greve (UFBA)
Fabio Costa (UFG)
Fabio Kon (IME-USP)
Fabio Porto (EPFL)
Fabricio Alves Barbosa da Silva (Universidade de Lisboa)
Francisco Brasileiro (UFCG)
Hélio Guardia (UFSCAR)
Jairo Panetta (INPE/CPTEC)
Liria Sato (USP)
Lucia Drummond (UFF)
Luis Carlos De Bona (UFPR)
Marcio Faerman (RNP)
Maria Clicia Castro (UERJ)
Marta Mattoso (UFRJ)
Noemi Rodriguez (PUC-Rio)
Patricia Kayser Vargas (Centro Universitário La Salle)
Philippe Navaux (UFRGS)
Rossana M de Castro Andrade (UFC)
Vinod Rebello (UFF)
Wagner Meira, Jr. (UFMG)
Walfredo Cirne (UFCG e Google)

Revisores

Alba Melo (UnB)
Alexandre Sztajnberg (UERJ)
Antônio Tadeu A Gomes (LNCC)
Artur Ziviani (LNCC)
Bruno Schulze (LNCC)
Celso Mendes (UIUC)
Cesar De Rose (PUCRS)
Claudio Geyer (UFRGS)
Diego Carvalho (CEFET-RJ)
Edison Ishikawa (IME-RJ)
Edmundo Madeira (UNICAMP)
Elizeu Santos Neto (Univ. of British Columbia)
Fabíola Greve (UFBA)
Fabio Costa (UFG)
Fabio Kon (IME-USP)
Fabio Porto (EPFL)
Fabricio Alves Barbosa da Silva (Universidade de Lisboa)
Francisco Brasileiro (UFCG)
Francisco José Silva (UFMA)
Gabriela Ruberg (UFRJ)
Hélio Guardia (UFSCAR)
Jairo Panetta (INPE/CPTEC)
Liria Sato (USP)
Lucia Drummond (UFF)
Luis Carlos De Bona (UFPR)
Marcio Faerman (RNP)
Maria Clicia Castro (UERJ)
Marta Mattoso (UFRJ)
Michael Stanton (RNP)
Noemi Rodriguez (PUC-Rio)
Patricia Kayser Vargas (Centro Universitário La Salle)
Philippe Navaux (UFRGS)
Rodrigo Calheiros (PUCRS)
Rossana M de Castro Andrade (UFC)
Tiago Ferreto (PUCRS)
Vinod Rebello (UFF)
Wagner Meira, Jr. (UFMG)
Walfredo Cirne (UFCG e Google)

Sumário

Sessão Técnica 1

Sessão Técnica 2

Sessão Técnica 3

Sessão Técnica 4

Pôsteres

Um Sistema Baseado em Grid para Interoperabilidade de PACS Distribuídos e Heterogêneos

Ramon G. Costa, Alvaro C. P. Barbosa, Saulo Bortolon

Programa de Pós-Graduação em Informática
Laboratório de Pesquisa em Redes e Multimídia – LPRM
Departamento de Informática
Universidade Federal do Espírito Santo – UFES
Campus de Goiabeiras
Av. Fernando Ferrari, 514, CEP 290750-910 Vitória, ES

{ramoncosta,alvaro,bortolon}@inf.ufes.br

Abstract. *PACS (Picture Archiving and Communication System) integrates image acquisition devices, archiving units, computational processing and healthcare image database in networked systems. PACS, in general, are stand-alone systems having its own image storage and patients records format. It was not designed to enable interoperability. This paper presents a system to integrate distributed and heterogeneous PACS using grid computing for the execution of distributed queries and for image visualization. This system enables the implementation of image queries stored in PACS systems demanding high and distributed processing power.*

Resumo. *PACS (Picture Archiving and Communication System) são dispositivos de aquisição de imagem, unidades de armazenamento, processamento computacional e bancos de dados de imagens médicas integrados em rede. Os PACS, em geral , são sistemas stand-alone que têm seu próprio formato para armazenar imagens e informações de pacientes e não foram projetados para possibilitar a interoperabilidade. Este artigo apresenta um sistema para a integração de PACS distribuídos e heterogêneos usando Grid para a execução de consultas distribuídas e visualização de imagens. O sistema permite a realização de consultas sobre imagens armazenadas em PACS que demandam alto poder de processamento distribuído.*

1. Introdução

Na área da saúde tem se tornado comum a necessidade de se integrar/compartilhar dados e operações entre diferentes sistemas. Neste sentido, diversas soluções têm sido estudadas e propostas [Erberich et al. 2007, Sharma et al. 2007, Gurcan et al. 2007]. Tal necessidade surge tanto entre instituições diferentes quanto entre sistemas de uma mesma instituição, como por exemplo, na integração de sistema de informações hospitalares (HIS - *Hospital Information System*), prontuário eletrônico de pacientes (EPR - *Electronic Pacients Record*), sistema de informações de radiologia (RIS - *Radiology Information System*), entre outros [Bakker 1991, Furuie et al. 2003].

Uma família de sistemas de informação de grande importância na área da saúde, atualmente, é a dos Sistemas de Imagens Médicas. Dentro dessa família destacam-se os

PACS (*Picture Archiving and Communication System* - Sistema de Comunicação e Arquivamento de Imagens) [Huang 2004]: dispositivos de aquisição de imagem, unidades de armazenamento, processamento computacional e bancos de dados de imagens médicas integrados em rede. PACS tem revolucionado principalmente a prática da radiologia por diminuir o uso de papel e filme radiológico, permitir revisões consistentes, melhorar o tempo e precisão dos diagnósticos e compartilhar imagens entre diferentes equipamentos, profissionais de saúde e estações de visualização. O sucesso dessas aplicações vem da adoção generalizada do padrão DICOM (*Digital Imaging and Communication in Medicine*) [NEMA 2008a]: um conjunto de normas para tratamento, armazenamento e transmissão de informação e de imagens médicas, encapsuladas com suas informações inerentes, em um formato eletrônico, padronizando as mensagens trocadas entre os subsistemas. Desse modo, informações sobre determinado paciente, tais como dados demográficos e histórico são encapsulados junto com informações que indicam como a imagem deve ser apresentada, como por exemplo o tamanho em pixels, resolução, contraste, etc.

Os PACS, na sua maioria, são sistemas *stand-alone* que têm sua própria forma de armazenar imagens e informações de pacientes e não foram projetados para possibilitar a interoperabilidade. A utilização do padrão DICOM permite apenas conectividade entre os diferentes PACS, por meio de troca de mensagens que utilizam uma interface comum. Uma alternativa para instituições, clínicas, hospitais e pesquisadores para a interoperabilidade de PACS que estão dispersos geograficamente é a criação de uma organização virtual para integrá-las e compartilhar não somente imagens da área de medicina, tais como radiografias, mamografias, raios-x, tomografias, etc, mas também compartilhar os recursos computacionais necessários para processá-las. Neste sentido devem ser pesquisados e desenvolvidos sistemas middleware de integração de dados voltados para esse domínio de aplicação o que demanda o uso de novas tecnologias e o compartilhamento de recursos computacionais (capacidade de armazenamento, processamento de imagens, reconhecimento de padrões, integridade, transmissão e segurança dos dados).

Por sua vez, *Grid* [Foster et al. 2002] é um ambiente computacional que provê o processamento de aplicações em paralelo que podem cooperar entre si em uma determinada tarefa em um ambiente seguro, transparente, coordenado e escalável onde as aplicações podem estar geograficamente dispersas. Assim, nada mais natural que sistemas middleware de integração de dados voltados para aplicações de saúde tirarem proveito da computação em *Grid* para melhorar o desempenho, a estrutura e o uso estável de recursos compartilhados, por meio de uma organização virtual capaz de criar interoperabilidade entre diversos dispositivos [Montagnat et al. 2004, Erberich et al. 2007].

2. Motivação

Desenvolver sistemas middleware de integração de dados não é uma tarefa simples devido à complexidade de suportar diferentes recursos computacionais, semânticas, tecnologias, modelos de dados, estratégias de processamento de consultas, etc. Integração de dados é um problema ubíquo e criticamente importante que vem demandando pesquisa na área de banco de dados por mais de uma década [Abiteboul 2005, Sheth and Larson 1990], com várias questões ainda em aberto [Ziegler and Dittrich 2004, Chiticariu et al. 2007]. Atualmente, é importante explorar oportunidades para combinar banco de dados e tecnologias relacionadas que podem incrementar o uso da informação [Abiteboul 2005].

Recentemente, nosso grupo de pesquisa passou a desenvolver trabalhos em telemedicina [Andreão et al. 2006] em parceria com pesquisadores do Hospital Universitário da UFES. No momento, existem dois projetos de pesquisa em desenvolvimento: *i)* "Software Livre e Interoperabilidade em Saúde: Aplicações em Radiologia, Cardiologia e Reumatologia" (financiamento 2007 FAPES[1]): objetiva explorar a Infra-estrutura da RUTE (Rede Universitária de Telemedicina) para pesquisar, desenvolver, testar e implantar soluções de software livre para PACS e desenvolver soluções para integrar o prontuário tradicional, baseado em papel, com sistemas de imagens, através de soluções de digitalização de baixo custo; *ii)* "Federação de PACS-DICOM para Telepatologia e Pesquisa de Doenças Infecciosas usando Microscopia"(Financiamento 2008 FINEP[2]): visa adaptar o padrão DICOM a imagens JPEG2000 provenientes de lâminas virtuais, integrar um sistema PACS-DICOM em um *Grid* federativo para suporte à pesquisa em doenças infecciosas e diagnosticar tuberculose em exames de microscopia usando processamento de imagens e reconhecimento de padrões.

Neste cenário, tem-se atualmente os seguintes desafios a serem explorados:

2.1. Quantidade de imagens a serem armazenadas

Com o aumento da digitalização de imagens provenientes de diferentes modalidades (radiografias, mamografias, raios-x, tomografias, etc.) surge o problema de se armazenar e recuperar grandes volumes de dados de servidores PACS, RIS e HIS. Assim, devem ser pesquisadas novas soluções e o uso de tecnologias apropriadas visando um menor tempo para o armazenamento, acesso, transmissão e processamento de tais informações [Huang 2004], garantindo confiabilidade, segurança e integridade.

2.2. Replicação e Back-up

Um aspecto crítico é que a tecnologia de PACS não foi projetada para suportar tolerância a falhas, necessitando de replicação, além de backups de segurança diários que podem demorar horas. Uma alternativa para superar estes problemas é com a utilização de *Grid*, pois além de distribuir o processamento computacional, permite a replicação das aplicações e de seus dados. Conseqüentemente, há um incremento potencial da confiabilidade e disponibilidade associada às aplicações de imagens médicas [Amendolia 2005, Liu 2005, Erberich et al. 2007].

2.3. Controle de Acesso e Privacidade

Diferentes categorias de profissionais de saúde devem ter acesso a diferentes níveis de informações armazenadas. Neste sentido os sistemas devem prover mecanismos para determinar e autorizar acessos a dados de pacientes considerando as normas legais sobre privacidade de dados de pacientes [Huang 2004].

2.4. Tamanho das imagens transportadas

O maior problema em se transportar grandes imagens é o fato delas excederem o limite suportável pelo DICOM de 64k x 64k pixels e 2 Gigabytes. Por exemplo, imagens de lâminas de patologia podem chegar a ter 60 Gigabytes. Uma solução é fragmentar a imagem a fim de que possam ser transportadas como uma série. Geralmente, isso é feito

[1]Fundação de Apoio a Ciência e Tecnologia do Estado do Espírito Santo

[2]Financiadora de Estudos e Projetos

com imagens de tomografia computadorizada e ressonância magnética [NEMA 2008b]. Porém estas séries devem ser processados como um conjunto único de informações conforme apresentado a seguir.

2.5. Composição e processamento de imagens de microscopia

A utilização de imagens no formato digital possibilita o uso de software para diagnósticos de imagens médicas. Tome como exemplo a Figura 1, que mostra a imagens de lâminas de microscopia óptica (MO) para exame de baciloscopia de escarro. A Figura 1(a) mostra um campo da lâmina visualizada no MO, usando fluorescência. Uma única lâmina pode ter de 500 a 2500 imagens como estas. Em cada uma delas, um algoritmo para reconhecimento de bacilos como o apresentado em [Forero et al. 2003] pode ser usado para o diagnóstico automatizado de Tuberculose. Estas milhares de imagens podem ser processadas em nós dispersos em *Grid*, o número total de bacilos somados e usado para acompanhamento da evolução da doença em cada paciente. Desta forma, diversas imagens de uma mesma lâmina podem ser processadas em paralelo. A figura 1(b) mostra assinaturas de bacilos, que são comparadas com as imagens do MO.

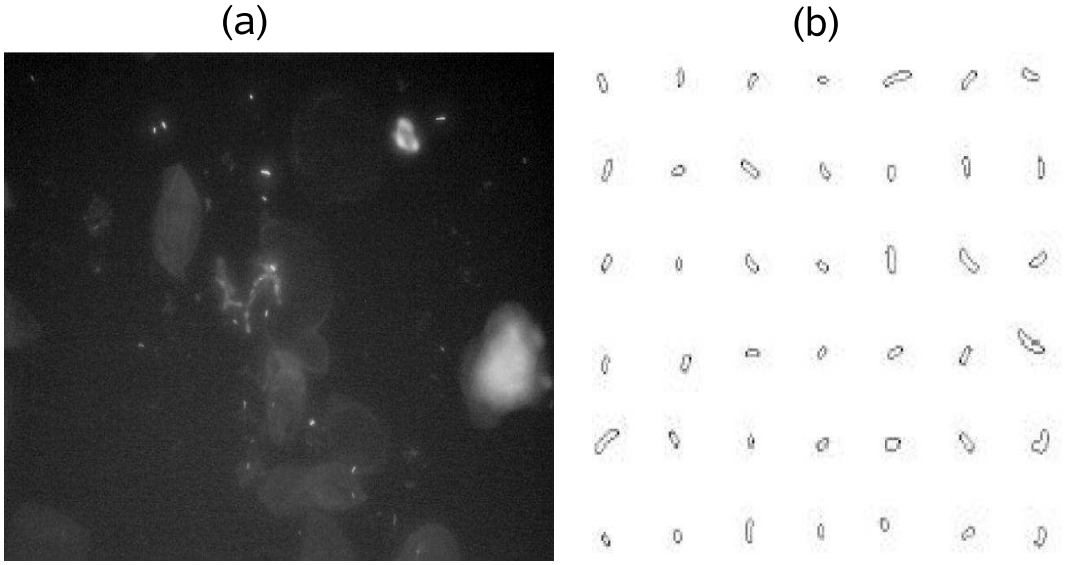


Figura 1. (a) Imagem de Microscopia Óptica em baciloscopia. (b) assinaturas de bacilos para reconhecimento automatizado em baciloscopia

2.6. Descoberta de imagens de um mesmo paciente em diferentes PACS

Uma grande contribuição para a área de saúde é o acesso completo e integrado de registros de um mesmo paciente, a partir de uma mesma estação de trabalho, dados esses armazenados em diferentes PACS, RIS, HIS, etc. de diferentes clínicas/hospitais. Isso possibilitaria a um médico acessar resultados de exames laboratoriais, registros odontológicos, imagens e prontuários, considerando que um dente infeccionado pode ser a causa de uma dor de cabeça, um problema oftalmológico ou auditivo.

2.7. Marcação de Exames

Uma outra necessidade é descobrir, para uma mesma especialidade, agendamentos de consultas e exames em diferentes clínicas/hospitais, considerando o equipamento a ser

utilizado (tempo de uso, resolução, etc.), experiência do profissional e que possa ser agendado para o mais breve possível.

A partir da experiência adquirida na pesquisa e desenvolvimento de sistemas middleware de integração de dados [Barbosa et al. 2002, Fontes et al. 2004, Silva et al. 2006, Trevisol et al. 2007, Oliveira and Barbosa 2007] decidimos investigar novas abordagens a serem usadas em telemedicina. Em particular, estamos interessados na interoperabilidade entre PACS utilizando *Grid*. Neste sentido, este artigo apresenta um sistema para integração de PACS distribuídos e heterogêneos utilizando *Grid* para o acesso, processamento, composição e visualização de imagens. O artigo está assim dividido: a seção 3 apresenta alguns trabalhos relacionados mais citados na literatura. Na seção 4, o sistema é descrito. A seção 5 descreve como o trabalho está sendo implementado e a seção 6 conclui este artigo.

3. Trabalhos Relacionados

3.1. GLOBUS MEDICUS

Globus MEDICUS (*Medical Imaging and Computing for Unified Information Sharing*) é um projeto do Globus Alliance Incubator, um projeto colaborativo entre a Universidade Southern California e o Information Science Institute que busca promover o compartilhamento de informações e imagens médicas [Erberich et al. 2007].

O objetivo é prover transparência e soluções *open-source* para o processamento e compartilhamento de imagens médicas entre clínicas, médicos e centros de pesquisas médicas. Para tanto utiliza o DGIS (*DICOM Grid Interface Service*), um *gateway* entre os PACS e o *Grid*. DGIS usa banco de dados relacional para armazenar informações de metadados e índices de atributos DICOM relevantes. Globus MEDICUS visa criar uma compilação do Globus Toolkit [Globus 2004] para a comunicação de imagens médicas, diagnósticos e dados relacionados entre as instituições que utilizam o ambiente *Grid*. O ponto central é criar uma federação, um compartilhamento e uma virtualização em larga escala de imagens médicas usando o padrão DICOM, utilizando *Open Grid Service Architecture* (OGSA) [Foster and Gannon 2003]. O DGIS centraliza todas as requisições o que gera *overhead*, apesar de utilizar *multithread*: deveria tirar proveito do ambiente *Grid* para tratar as requisições. Além disso, trata a heterogeneidade de modelos e não dos dados.

3.2. VirtualPACS

[Sharma et al. 2007] descreve VirtualPACS como sendo um servidor de PACS-DICOM virtualizado que cria uma federação de múltiplas fontes de dados remotas, incluindo as que não suportam mensagens DICOM, apresentando-as para um cliente DICOM como um único recurso virtual. Uma federação de fontes de dados de imagens requer: *i*) uma representação comum dos dados entre as fontes de dados; *ii*) ferramentas para transformação dos modelos de dados das fontes; *iii*) um mecanismo para descobrir fontes de dados no ambiente; *iv*) um mecanismo para interagir com fontes de dados seguras e inseguras; *v*) um mecanismo para traduzir uma requisição DICOM para uma nova requisição que a fonte entenda. Para atender a estes objetivos, a plataforma VirtualPACS apresenta 3 camadas. A camada de Mediação fica responsável pelos itens *i* e *ii*. A camada de Middleware implementa suporte para os itens *iii* e *iv*. Finalmente, o requisito *v* é atendido pela camada de Apresentação.

Diferente do projeto MEDICUS, que suporta apenas bancos de dados em SQL para publicação e descoberta de serviços, o VirtualPACS disponibiliza uma camada intermediária para que estações de trabalho possam fazer suas requisições e transformar em um modelo comum de linguagem chamado caGrid Query Language (CQL) [CQL , Oster et al. 2007]. Essa transformação para o modelo CQL, necessária neste ambiente, gera *overhead*. Além disso a camada de apresentação (equivalente a tradução DICOM) não está em *Grid* e é necessária a instalação do VirtualPACS em cada cliente. VirtualPACS, tal como o Globus MEDICUS, trata a heterogeneidade de modelos e não dos dados.

Um outro projeto do mesmo grupo de pesquisa do VirtualPACS é o GridIMAGE [Gurcan et al. 2007]. Neste projeto o objetivo é enviar imagens DICOM para uma particular coleção de leitores humanos e algorítmos de detecção, receber os resultados e comparar, posteriormente, a interpretação de cada um deles.

3.3. CoDIMS-Grid

Sistemas de integração de dados vêm sendo desenvolvidos com o objetivo de fornecer, a partir de fontes de dados essencialmente heterogêneas e distribuídas, uma visão única, uniforme e homogênea. Na tentativa de solucionar esse problema foi proposto o CoDIMS (*Configurable Data Integration Middleware System*)[Barbosa et al. 2002]: um ambiente flexível e configurável para gerar sistemas configurados para a integração de dados heterogêneos e distribuídos. Os sistemas assim gerados são denominados instâncias do CoDIMS [Oliveira and Barbosa 2007]. Exemplos de instâncias do CoDIMS desenvolvidas especificamente para aplicações em *Grid* são encontradas em [Fontes et al. 2004, Silva et al. 2006, Trevisol et al. 2007].

[Trevisol et al. 2007] apresenta uma Máquina de Execução de Consultas Distribuída (MECD) inserida em um ambiente de *Grid* para execução de sub-consultas e de operadores de uma forma distribuída e paralela. O uso da MECD possibilita:

1. Realizar o processamento distribuído do Plano de Execução de Consultas (PEC) através de operadores algébricos que serão executados remotamente nos nós do *Grid*;
2. Disponibilizar o resultado gerado pela execução de um operador para ser utilizado por outros operadores de acordo com hierarquia da árvore que descreve o PEC;
3. Alocar e desalocar instâncias dos operadores em nós do *Grid* de modo a otimizar o uso dos recursos computacionais disponíveis e diminuir o tempo de processamento da consulta.

Apesar do desenvolvimento da MECD, o CoDIMS-Grid não foi desenvolvido para acesso a PACS. Deve, então, ser projetada outra instância do CoDIMS para permitir não somente o acesso, mas também o processamento, a composição e a visualização de imagens de PACS usando *Grid*. A nova instância foi denominada CoDIMS-Grid PACS, conforme apresentado a seguir.

4. O CoDIMS-Grid PACS

O CoDIMS-Grid PACS tem por objetivo interoperabilizar PACS heterogêneos e distribuídos, possibilitando o uso de consultas distribuídas que utilizam todas as vantagens

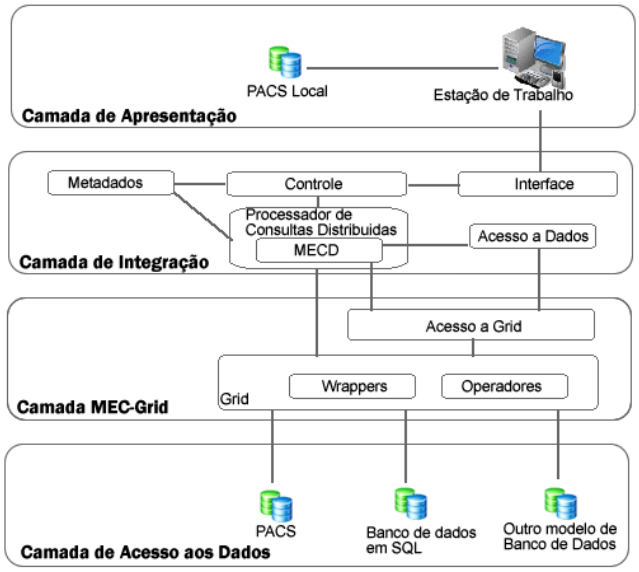


Figura 2. Arquitetura proposta para o CoDIMS-Grid PACS.

do uso de *Grid*. Também permite o acesso a dados armazenados em RIS, HIS, dispositivos DICOM bem como não-DICOM (ver Figura 2(b)).

Consultas globais são submetidas através da camada de Apresentação para fazer requisições de imagens armazenadas em PACS. A Camada de Integração contém um componente Interface que se comunica com as estações de trabalho, recebendo as requisições e encaminhado-as para o componente Controle. A requisição é então repassada ao componente Processador de Consultas para executar as fases de análise, re-escrita e otimização de consultas, gerando assim um PEC. O PEC é então encaminhado à MECD para gerenciar sua execução. O componente Metadados contém a descrição dos dados das fontes de dados. A camada de Integração pode estar dispersa em diversos pontos da rede, inclusive no lado cliente. Isto é possível porque os componentes do CoDIMS-Grid PACS se comunicam através de uma arquitetura orientada a serviços (SOA) [Trevisol et al. 2007]. Além disso, o CoDIMS-Grid PACS se propõe a criar planos de execução de consultas distribuídas e definir a junção de resultados de nós diferentes, utilizando um destes nós para a junção, antes mesmo de retorná-los à MECD na camada de Integração.

A Camada MEC-Grid utiliza *Grid* para executar as sub-consultas sobre as fontes de dados. Nela, cada nó pode prover a funcionalidade de uma máquina de execução de consultas, definidas através de planos criados na camada superior. Quando uma sub-consulta é escalonada para execução em um determinado nó, este receberá apenas os operadores/programas necessários à sua execução. Além disso, os *wrappers* fazem parte desta camada e são instanciados dinamicamente de acordo com a fonte de dados.

Na camada de Acesso aos Dados existe a possibilidade de interface com PACS DICOM e não-DICOM, incluindo RIS e HIS. *Wrappers* precisam ser desenvolvidos e instanciados para dispositivos que não utilizam este padrão. No caso de um modelo integrador que não seja DICOM, um *wrapper* deve ser instanciado para as fontes DICOM compatíveis: uma opção em casos que os PACS não são compatíveis com DICOM.

A cada consulta global executada será gerado um plano de execução de consultas a ser executado pela Máquina de Execução de Consultas Distribuída (MECD) do CoDIMS-Grid PACS, que utilizará os *wrappers* para fazer as requisições. Os resultados serão retornados para visualização. A partir deste momento o usuário na estação de trabalho poderá fazer requisições de visualização de imagens que constarem na lista de resposta. Novas requisições podem ser impostas à interface a cada operação realizada. Os sub-resultados gerados pelos *wrappers* já estarão localizados nos nós do *Grid* [Biancardi 2005]. Neste caso a MECD [Trevisol et al. 2007] enviará os operadores/programas para serem executados nos nós do *Grid* de acordo com o PEC recebido, possibilitando assim a execução paralela das operações descritas pelo plano.

A grande vantagem da abordagem adotada no CoDIMS-Grid PACS é que ela aproveita o fato das fontes de dados executarem as consultas em paralelo nos nós do *Grid*, fazendo a junção dos sub-resultados obtidos nestes mesmos nós. Dessa forma, otimiza o tempo de resposta para aplicações que necessitam de grande poder computacional.

5. Implementação

O trabalho vem sendo implementado no Laboratório de Redes e Multimídia da Universidade Federal do Espírito Santo. Todas as estações de trabalho utilizam sistemas operacionais Linux, Globus Toolkit e a linguagem de programação Java. A implementação da arquitetura orientada a serviços (SOA) é feita utilizando *Grid Services* publicados pelo Globus toolkit e *Web Services* em alguns componentes do CoDIMS-Grid PACS. Estes serviços fazem o papel de comunicar o CoDIMS-Grid PACS com a interface DICOM de comunicação com os PACS. Está sendo desenvolvida uma interface Web para visualização da lista de imagens ou das informações retornadas pela consulta.

Estão sendo utilizados os PACS O3-DPACS [O3] e DCM4CHE [DCM4CHE] como estudos de caso para a recuperação de informações através de consultas utilizando DICOM, alocados em nós da rede participantes do ambiente de *Grid*. O3-DPACS é um PACS em software livre e de plataforma independente que utiliza a linguagem Java, o Sistema Gerenciador de Bancos de dados MySQL ou PostgreSQL e é desenvolvido sobre o Jboss. O O3-DPACS utiliza o DICOM como protocolo de comunicação entre dados clínicos, sinais e imagens, e o HL7 (*Health Level 7*) [Health Level Seven] como protocolo para a comunicação com os dados administrativos. DCM4CHE é um software livre para a gerência de imagens clínicas que implementa o padrão DICOM como protocolo de comunicação e o HL7 para alguns serviços de armazenamento de imagens. Existe ainda uma versão enterprise, chamada DCM4CHEE [DCM4CHE] para a gerência de imagens de acordo com IHE [Siegel and Channin 2001].

A implementação está em desenvolvimento. Na camada de Acesso aos Dados foram instalados o O3-DPACS e o DCM4CHE. Para a camada MEC-Grid foi adaptada uma versão do CoDIMS-Grid e desenvolvidos os *wrappers* necessários para acesso aos PACS. Na camada de Integração, também foi adaptada uma versão do CoDIMS-Grid, ainda no ambiente globus toolkit 3. Na camada de Apresentação está em fase final de desenvolvimento um visualizador de imagens. Na versão inicial pretende-se apresentar uma lista de imagens armazenadas em diferentes PACS usando uma condição de pesquisa (nome do paciente, data da imagem, servidos, etc.). A partir dessa lista pode-se selecionar uma imagem para visualização. Um trabalho posterior é migrar para o ambiente globus

toolkit 4.

Pretende-se futuramente utilizar ontologias para tratamento da heterogeneidade semântica entre os diferentes PACS, utilizando como base o trabalho definido em [Silvestre 2005] estendendo-o para o ambiente *Grid*. Pretende-se também adaptar a MECD [Trevisol et al. 2007] para processamento distribuído na composição de imagens multidimensionais.

Será incluído um banco de dados e um PACS não-DICOM para a interoperabilidade com o ambiente de *Grid*. Para esta fonte será necessária a criação de um *wrapper* que faça a conversão entre o esquema do modelo global e o esquema do modelo de dados da fonte.

6. Conclusão

Sistemas de integração de dados consistem de ambientes que liberam o usuário de ter que localizar as fontes de dados, interagir com cada uma delas isoladamente e combinar manualmente os dados de múltiplas fontes, heterogêneas e distribuídas [Halevy 2003]. A computação em *Grid* cria um ambiente computacional no qual aplicações podem utilizar múltiplos recursos computacionais distribuídos de forma segura, coordenada, eficiente e transparente. PACS consistem de dispositivos de aquisição de imagem, unidades de armazenamento, processamento computacional e bancos de dados de informações e de imagens médicas integrados em rede. O CoDIMS-Grid PACS visa prover o uso racional destes três recursos.

O CoDIMS-Grid PACS consiste de um ambiente que possibilita o armazenamento e a execução de consultas distribuídas em PACS heterogêneos. Através da possibilidade de seu processamento e execução de consultas estar distribuído entre nós do *Grid*, é possível criar planos de execução que otimize o tempo de resposta em aplicações que necessitam de grande poder computacional e necessitam dividir a execução entre diversas aplicações implementadas em pontos dispersos. Tais aplicações podem ser executadas em nós diferentes que cooperam entre si para uma determinada tarefa, como por exemplo, para a procura de doenças em uma imagem que foi fragmentada e alocada em cada um destes nós. Através do uso de *Grid*, é possível manter estes sistemas em execução e criar planos de execução de consultas que façam a junção de resultados obtidos em cada nó.

O CoDIMS-Grid PACS, diferente de outros trabalhos, pretende ser um sistema de integração de dados completo e não apenas um intermediador de requisições entre clientes e PACS. Ou seja, pretende-se unir resultados de trabalhos anteriores, adaptando-os para essa nova área de aplicação. Assim, estão previstos além do uso de *Grid* para execução de consultas [Trevisol et al. 2007], o escalonamento de tarefas em nós do *Grid* [Silva et al. 2006], o uso de ontologias para tratamento semântico dos dados [Silvestre 2005] e o uso de controle de acesso, segurança e privacidade para acesso aos dados [Andreão et al. 2006].

Para suporte a consultas em paralelo de fontes de dados nos nós do Grid pretende-se utilizar o OGSA-DQP (*Open Grid Services Architecture - Distributed Query Processing*) [Alpdemir et al. 2004]. Porém, por utilizar um modelo global pré-definido, pode gerar uma quantidade maior de mapeamentos e acessos ao diretório global quando há poucos ou nenhum modelo local semelhante. No CoDIMS-Grid PACS, por sua vez, utiliza-se um modelo global que é implementado de acordo com o modelo local mais utilizado.

Neste caso, como a maioria dos PACS utilizam DICOM para prover interoperabilidade, o modelo global utilizado será criado através desta especificação. Também no caso dos wrappers, o OGSA-DAI (*Open Grid Services Architecture – Data Access and Integration*) [Antonioletti et al. 2005] será preferencialmente usado, desde que não venha exigir tradução de modelos, mapeamentos e conversões desnecessárias. As especificações, esquema global, mapeamentos, casos de uso e wrappers estarão disponíveis em [CoDIMS].

Esta proposta contribui com o desenvolvimento de dois projetos de pesquisa atualmente em execução. Os projetos visam desenvolver aplicações de telemedicina, em particular o tratamento de imagens digitalizadas de exames médicos, utilizando principalmente software livre. Pretende-se desenvolver aplicações que possam ser utilizadas pelo sistema público de saúde (SUS) para permitir uma melhoria e rapidez no diagnóstico de doenças, o que pode diminuir o tempo de internação, evitar o deslocamento para realização de exames em centros de referência, melhorando o serviço de atendimento à população com menor custo. Em particular pretende-se disponibilizar um sistema de telemedicina no Hospital Universitário da UFES, apoiando a criação e manutenção destes sistemas em hospitais e clínicas da Grande Vitória e do estado.

Referências

Abiteboul, S. (2005). The lowell database research self-assesment. *Communications of the ACM*, 48(5):111–118.

Alpdemir, M. N., Mukherjee, A., Gounaris, A., Paton, N. W., Watson, P., Fernandes, A. A. A., and Fitzgerald, D. J. (2004). *OGSA-DQP: A Service for Distributed Querying on the Grid*, volume 2992/2004 of *Lecture Notes in Computer Science*, pages 858–861. Heidelberg.

Amendolia, S. R. (2005). Deployment of a grid-based medical image application. *Stud Health Tecnol Inform*, 112:59–69.

Andreão, R. V., Pereira Filho, J. G., and Calvi, C. Z. (2006). Telecardio: Telecardiologia a serviço de pacientes hospitalizados em domicílio. In *XX Congresso da Sociedade Brasileira de Informática em Saúde (CBIS 2006)*, pages 1267–1272, Florianópolis, Santa Catarina, Brasil.

Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Hong, N. P. C., Collins, B., Hardman, N., Hume, A., Knox, A., Jackson, M., Krause, A., Laws, S., Magowan, J., Paton, N. W., Pearson, D., Sugde, T., Watson, P., and Westhead, M. (2005). The design and implementation of grid database services in ogsa-dai. pages 357–376.

Bakker, A. R. (1991). His, ris and pacs. *Computerized medical imaging and graphics, Spring-Verlang*, 15:157–60.

Barbosa, A. C. P., Porto, F. A. M., and Melo, R. N. (2002). Configurable data integration middleware system. *Journal of the Brazilian Computer Society*, 8(2):12–19.

Biancardi, C. (2005). Distribuição e execução de wrappers em ambiente de grid para o codims. Master's thesis, UFES. Available in: http://codims.lprm.inf.ufes.br/publicacoes.html.

Chiticariu, L., Hernández, M. A., Kolaitis, P. G., and Popa, L. (2007). Semi-automatic schema integration in clio. In *Proceedings of the 33rd international conference on Very large data bases table of contents*, pages 1326–1329, Vienna, Austria.

CoDIMS. Configurable data integration middleware system. Site Web. http://codims.lprm.inf.ufes.br/.

CQL. cagrid query language. Site Web. http://www.cagrid.org/mwiki/index.php?title=CQL.

DCM4CHE. Dcm4che: Open source clinical image and object management. Site Web. http://www.dcm4che.org.

Erberich, S. G., Silverstein, J. C., Chervenak, A., Schuler, R., Nelson, M. D., and Kesselman, C. (2007). Globus medicus - federation of dicom medical imaging devices into healthcare grids. *Studies in Health Technology and Informatics*, 126:269–278.

Fontes, V., Schulze, B., Dutra, M., Porto, F., and Barbosa, A. C. P. (2004). Codims-g: a data and program integration service for the grid. In *Proceedings of the 2nd workshop on Middleware for grid computing*, volume 76, pages 29–34, Toronto, Ontario, Canada. ACM International Conference Proceeding Series.

Forero, M. G., Cristóbal, G., and Alvarez-Borrego, J. (2003). Automatic identification techiniques of tuberculosis bacteria. *SPIE proceedings of the applications of digital image processing XXVI*, 5203:71–81.

Foster, I. and Gannon, D. (2003). The open grid services architecture platform. Available in: http://www.globalgridforum.org/Meetings/ggf7/drafts/draft-ggf-ogsa-platform-2.pdf.

Foster, I., Kesselman, C., Nick, J., and Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed system integration. *Open Grid Service Infrastructure WG, Global Grid Forum.*

Furuie, S. S., Gutierrez, M. A., Figueiredo, J. C. B., Tachinardi, U., Rebelo, M. S., Bertozzo, N., Moreno, R. A., Motta, G. H. M. B., Nardon, F. B., and Oliveira, P. P. M. (2003). Prontuário eletrônico de pacientes: integrando informações clínicas e imagens médicas. *Revista brasileira de engenharia biomédica*, 19:125–137.

Globus, T. G. A. (2004). The globus toolkit. Technical report. Available in: http://www.globus.org.

Gurcan, M. N., Pan, T., Sharma, A., Tahsin, Oster, S., Langella, S., Hastings, S., Siddiqui, D. M., Siegel, E. L., and Saltz, J. (2007). Gridimage: A novel use of grid computing to support interactive human and computer-assisted detection decision support. *Journal of Digital Imaging*, 20(2):160–171.

Halevy, A. Y. (2003). Data integration: A status report. In *Proceedings of 10th Conference on Database Systems for Business Technology and the Web (BTW 2003)*, Heppenhiem - Germany.

Health Level Seven, I. Hl7 - the health level 7. Site Web. http://www.hl7.org/.

Huang, D. H. K. (2004). *Pacs and Imaging Informatics: Basics Principles and Applications*. John Wiley and Sons, Inc., Hoboken, New Jersey.

Liu, B. J. (2005). Utilizing data grid architecture for the backup and recovery of clinical image data. *Computerized Medical Imaging and Graphics*, 29:95–102.

Montagnat, J., Bellet, F., and Benoit-Cattin, H. (2004). Medical images simulation, storage, and processing on the european datagrid testbed. *Journal of Grid Computing*, 2(4):387–400.

NEMA (2008a). Dicom - digital imaging and communications in medicine. http://medical.nema.org/ and ftp://medical.nema.org/medical/dicom/2008/.

NEMA (2008b). Dicom working group twenty-six - pathology. http://medical.nema.org/DICOM/minutes/WG-26.

O3. O3-dpacs. Open Three (O3) Consortium O3-DPACS, Site Web. http://www.o3consortium.eu.

Oliveira, N. Q. and Barbosa, A. C. P. (2007). Uma arquitetura para integração de dados em sistemas sensíveis ao contexto. In *VII Workshop de Teses e Dissertações (WTDWeb 2007)*, Gramado (RS). XIII Simpósio Brasileiro de Sistemas Multimedia e Web (WebMedia 2007).

Oster, S., Hastings, S. L., Langella, S., and Ervin, D. (2007). cagrid. http://www.cagrid.org.

Sharma, A., Pan, T., Cambazoglu, B. B., Gurcan, M., Kurc, T., and Saltz, J. (2007). Virtualpacs—a federating gateway to access remote image data resources over the grid. *Journal of Digital Imaging*.

Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed heterogeneous and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236.

Siegel, E. L. and Channin, D. S. (2001). Integrating the healthcare enterprise: A primer. *RSNA 2001 - Radiological Society of North America*, 21:1339–1341.

Silva, V. F. V., Dutra, M. L., Porto, F., Schulze, B., Barbosa, A. C. P., and de Oliveira, J. C. (2006). An adaptive parallel query processing middleware for the grid. *Concurrency and Computation - Practice and Experience archive*, 18(6):621–634.

Silvestre, L. J. (2005). Uma abordagem baseada em ontologias para a gerência de metadados do codims. Master's thesis, UFES. Disponivel em: http://codims.lprm.inf.ufes.br/publicacoes.html.

Trevisol, G. G., Biancardi, C., Barbosa, A. C. P., Filho, J. G. P., Costa, R. G., and Cardoso, E. S. (2007). A distributed query execution engine in a grid environment. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid07)*, pages 418–425.

Ziegler, P. and Dittrich, K. R. (2004). Three decades of data integration - all problems solved? In *First International IFIP Conference on Semantics of a Networked World - ICSNW 2004*.

Um portifolio de segurança para um sistema de grade entre pares de livre entrada

Flavio Figueiredo, Matheus Gaudêncio, Thiago Emmanuel, Rodrigo Miranda, Francisco Brasileiro

[1]Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, s/n, Bloco CO
58.109-970, Campina Grande - PB, Brazil

{flaviov,matheusgr,thiagoepdc,vilar,fubica}@lsd.ufcg.edu.br

Abstract. *Security is an important aspect in grid computing, due to the necessity to protect the grid resources, users and middleware. In free-to-join peer-to-peer grids, the security issues are more complex, because there are not strong identities in the system. In fact, there is not a "one size fits all" solution for the users of this kind of grid. This paper adapts the security technologies available in the other kinds of grids and distributed systems, applying then to the free-to-join peer-to-peer grids, in order to build a security portfolio for the distinct user types in this kind of system. The portfolio was developed and validated on OurGrid.*

Resumo. *Segurança é um aspecto importante na computação em grade, pois é preciso proteger os recursos, os usuários e o próprio middleware da grade. Nas grades entre pares e de livre entrada, as questões de segurança se tornam mais complexas, devido à ausência de identidades fortes no sistema. Na realidade, não existe uma solução única que satisfaça os requisitos de todos os usuários deste tipo de grade. Neste artigo, as tecnologias de segurança disponíveis para os outros tipos de grades e sistemas distribuídos são adaptadas e aplicadas às grades entre pares e de livre entrada, a fim de formar um portifolio de segurança disponível para os diferentes perfis de usuários deste tipo de sistema. O portifolio foi implementado e validado no OurGrid.*

1. Introdução

É cada vez mais comum o uso de grades computacionais para o auxílio de pesquisas científicas que exijam um grande poder computacional. Há na literatura propostas de diversos modelos de grades, cada um apresentando requisitos distintos de segurança para atender às necessidades dos seus respectivos usuários. Em geral, existem dois problemas de segurança em grades: (i) a segurança dos usuários da grade contra recursos maliciosos; e (ii) a segurança dos recursos da grade contra usuários maliciosos. Uma forma comum de garantir a proteção de ambas as partes é fazer uso de um esquema de segurança baseado na confiança mútua entre os participantes [Foster et al. 1998].

O OurGrid é um *middleware* de grades computacionais que permite que qualquer usuário possa participar do sistema. Em `http://status.ourgrid.org` pode-se

ver um *snapshot* atual do sistema. Uma grade que use o *middleware* OurGrid é caracterizada como uma rede de livre entrada. Neste tipo de grade, onde não há garantia sobre quem são os participantes do sistema, soluções de segurança baseadas em confiança mútua se tornam inviáveis. Consequentemente existe a necessidade de construir mecanismos explícitos para dar suporte à proteção de recursos e usuários da grade.

Trabalhos anteriores já foram desenvolvidos para implementar segurança no OurGrid. A solução SWAN [Cavalcanti et al. 2006] provê ambientes de execução seguros, a fim de proteger os recursos da grade contra usuários maliciosos. Também existe uma solução para a proteção de usuários contra recursos maliciosos [Oliveira and Brasileiro 2006]. Embora eficazes do ponto de vista da segurança provida, tais soluções não são adequadas para todos os perfis de usuários que fazem parte da grade. O SWAN, por exemplo, demanda um grande esforço de implantação, o que inviabiliza a utilização da grade para alguns usuários mais leigos, enquanto que a solução de proteção contra recursos maliciosos consome muito poder computacional para executar réplicas de segurança das tarefas da grade.

Para disseminar a comunidade OurGrid, implementamos um portifolio de segurança que visa suprir diferentes demandas de diferentes perfis de usuários. O portifolio permite que usuários adotem a melhor solução de segurança para o seu domínio administrativo, levando em consideração questões de eficácia e facilidade de administração.

Na seqüência deste artigo, a Seção 2 expõe a arquitetura do OurGrid e a Seção 3 contextualiza os aspectos de segurança da Computação Voluntária e da computação em Grade. O cerne do artigo está na Seção 4, que mostra em detalhes o portifolio de segurança proposto, discutindo a sua implementação. Por fim, a conclusão enumera as contribuições deste artigo e os trabalhos futuros.

2. OurGrid

O OurGrid é uma grade de livre entrada, onde os participantes compartilham recursos ociosos de processamento e armazenamento de dados, para a execução de tarefas não comunicantes (*BoT*, do inglês *Bag of Tasks*). Uma característica buscada pelo OurGrid é fornecer um sistema facilmente implantável e com componentes bem desacoplados.

A arquitetura do OurGrid, como representada na Figura 1, é composta por quatro componentes principais: o Broker, que é uma interface cliente de submissão de tarefas; o Peer, entidade que agrupa consumidores e recursos de um domínio, controlando a doação e requisição de máquinas; o Worker, recurso executor das tarefas repassadas pelo usuário; e o DiscoveryService, um serviço de descoberta de recursos. Todos os componentes se comunicam utilizando o middleware JIC(*Java Internet Communication*) [Lima et al. 2006].

Um usuário que deseja executar um conjunto de tarefas no OurGrid deve usar o Broker para a submissão deste trabalho. O Broker então pede máquinas ao Peer de seu domínio (Peer local) que deve: fornecer todos os seus Workers disponíveis para a execução desta tarefa ao mesmo tempo em que, através do DiscoveryService, procura Peers remotos que possuam recursos apropriados. O Peer local então pede aos Peers remotos máquinas para o usuário do Broker. Os workers serão doados de acordo com um mecanismo de incentivo, denominado Rede de Favores, (*NoF*, do inglês *Network-of-Favors*), no qual o Peer remoto tenta entregar mais máquinas aos Peers que, no passado,

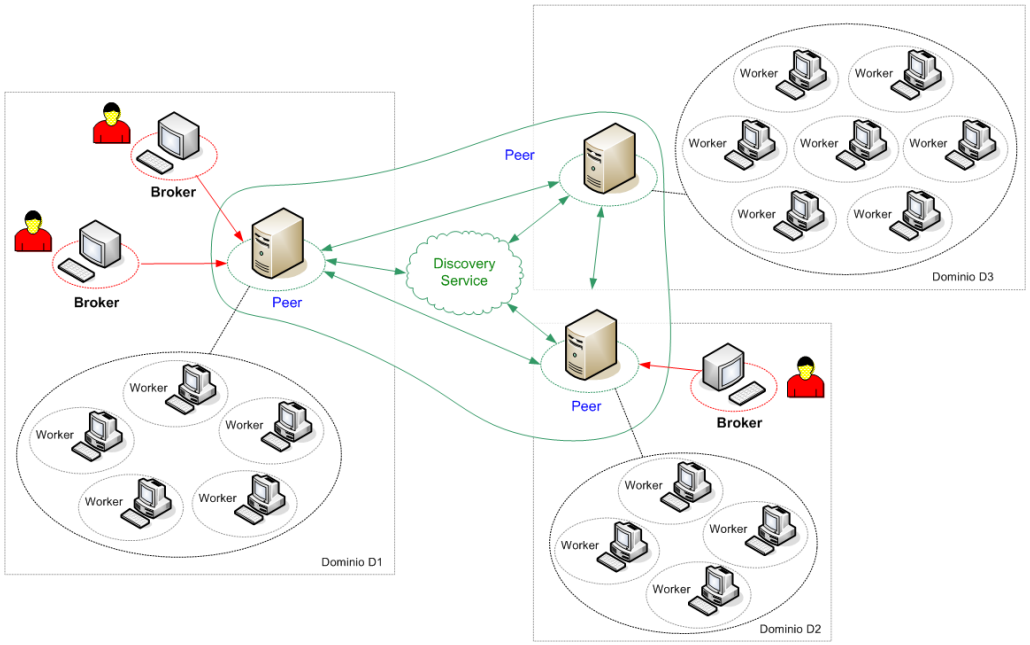


Figura 1. Componentes do OurGrid

lhe doaram mais recursos [Andrade et al. 2007]. O Broker, após receber cada recurso, escalona uma tarefa a ser executada imediatamente no Worker doado.

3. Contextualização

O OurGrid é um sistema de grade que tem características em comum com outros sistemas de computação distribuída. Para desenvolver um portifolio de segurança, inicialmente fizemos um levantamento bibliográfico sobre os diferentes modelos de computação distribuída e como estes provêem segurança para seus usuários.

3.1. Computação Voluntária

Computação voluntária é uma forma de computação distribuída em que recursos computacionais são disponibilizados espontaneamente por seus donos através da Internet. Sistemas que fazem uso deste modelo de distribuição alcançam taxas de processamento equiparáveis aos maiores supercomputadores construídos, como pode ser visto comparando projetos como o BOINC [Boinc Status 2007] e o ranking dos maiores supercomputadores do mundo [TOP 500 2007].

3.1.1. Segurança dos recursos

Para facilitar o ingresso de voluntários no sistema, devem ser fornecidas condições de segurança que satisfaçam as suas necessidades. Uma alternativa comum é controlar o processo de submissão das aplicações, que só ficam disponíveis para execução após serem verificadas como confiáveis pelo administrador da plataforma [Boinc Project 2007, Distributed.net 2007]. Geralmente, esta verificação implica na reescrita do código, o que diminui a facilidade de implantação de novas aplicações no sistema. Além disto,

é necessário que os donos dos recursos doados confiem em quem está realizando a verificação.

Uma estratégia bastante disseminada para a proteção de recursos contra aplicações é o uso de ambientes isolados, denominados de *sandboxes*. Implementações destes mecanismos incluem máquinas virtuais, jaulas no sistema operacional [Kamp and Watson 2000], modificações na visão do sistema de arquivos (*chroot*) e linguagens interpretadas [Gosling et al. 2005]. De forma geral, estes mecanismos mantêm sob controle um subconjunto seguro dos recursos que podem ser utilizados por processos não confiáveis.

Os sistemas para computação voluntária Bayanihan [Sarmenta 1998], Javelin [Neary et al. 1999] e o Unicorn [Ong et al. 2002] fazem uso de *Applets Java* como sandbox para as aplicações [Sun Microsystems 2007]. Esta arquitetura apresenta vantagens como a independência de plataforma e a baixa sobrecarga de configuração. Porém, as aplicações suportadas são restritas à linguagem Java.

O XtremWeb [Cappello et al. 2005] é um sistema entre-pares de computação voluntária, que implementa o isolamento através da *sandboxing* a nível do sistema operacional. No caso do XtremWeb, a limitação do que é permitido ser feito é definida através de políticas de acesso que controlam a manipulação do sistema de arquivos, conexões à rede e sinalização entre processos. Esta técnica é limitada, pois necessita de sistemas operacionais específicos e modificados para este fim.

3.1.2. Segurança das aplicações

Quando não se pode certificar que os voluntários são confiáveis, também é necessário proteger as aplicações contra eventuais sabotagens. Este tipo de falta pode ser tolerada usando técnicas de replicação e execução de tarefas com resultados conhecidos [Sarmenta 2001, Oliveira and Brasileiro 2006], o que torna possível a detecção de recursos maliciosos. Entretanto implica numa perda de processamento útil, proporcional ao grau de replicação.

Outra abordagem possível é a inclusão de marcações no código da aplicação. Esta técnica torna possível checar a integridade do código contra ataques em que recursos modificam a aplicação para retornar resultados inconsistentes. É possível para um usuário malicioso burlar esta técnica atráves de engenharia reversa do código para detecção das marcações. Alguns mecanismos como criptografia e obfuscação [Collberg and Thomborson 2002] se propõem a resolver este problema, dificultando o processo de engenharia reversa. Em grades computacionais entre-pares, diferente de computação voluntária, as aplicações em execução em um nó variam bastante, assim o trabalho de um sabotador em realizar a engenharia reversa não compensa o ganho que ele teria em sabotar a tarefa.

3.2. Computação em Grade

Os domínios administrativos que compõem uma grade possuem políticas locais de segurança. Uma solução de segurança que atua na grade idealmente não deve conflitar com as soluções locais de cada domínio. Outra preocupação é que relações de

confiança entre os participantes da grade não podem ser estabelecidas antes da execução das aplicações devido à natureza dinâmica da grade, em resumo não é possível saber a priori as entidades com as quais um relacionamento será estabelecido.

Foster et al. [Foster et al. 1998] descrevem, no contexto do projeto Globus [Foster and Kesselman 1998], uma arquitetura de segurança baseada em autenticação (ato de assegurar que uma entidade é quem diz ser) e autorização (comprovação do que é permitido ser feito por determinada entidade). Uma característica fundamental desta arquitetura é a independência de implementação, por exemplo, políticas de segurança podem usar qualquer tecnologia baseada em troca de chaves criptográficas. Além do Globus, outros sistemas como PBS [OpenPBS 2007], Sun Grid Engine [Engine 2007], TeraGrid [TeraGrid 2007] e gLite [gLite 2007], permitem apenas usuários autenticados e autorizados.

O projeto PUNCH [Kapadia et al. 2000] é uma plataforma de computação em grade que implementa uma solução de segurança que inclui: um *shell* modificado para aplicações interativas e o monitoramento em tempo de execução das chamadas ao sistema. Este *shell* modificado verifica se os comandos são seguros via políticas baseadas em controle de acesso e descritas através de um arquivo de configuração. Esta abordagem é útil quando utilizada para usuários específicos e aplicações não-arbitrárias, caso contrário a definição de diretivas de controle de acesso pode ser muito restritiva ou mesmo ineficaz desde que não se sabe o que estará em execução.

O Condor [Litzkow et al. 1988] é um sistema de gerenciamento de tarefas para computação intensiva. Este projeto evoluiu de forma que pode ser usado para construir uma infra-estrutura de grade, compartilhando recursos entre diferentes domínios administrativos [Frey et al. 2002]. A solução de segurança no Condor [Thain et al. 2005] incorpora: o gerenciamento das identidades dos participantes e a proteção dos recursos contra aplicações. O gerenciamento de identidades é realizado usando os mesmos protocolos utilizados pelo projeto Globus [Foster and Kesselman 1998]. Atualmente a segurança dos recursos é feita limitando o acesso remoto a uma conta de login restrita (uma modificação da conta padrão *Unix*, *nobody*). Em versões anteriores, as soluções de segurança do Condor incluíram o uso de *chroot*, uma técnica de isolamento que restringe a execução de processos em porções seguras do sistema de arquivos. Esta técnica é considerada ineficaz, pois existem vulnerabilidades conhecidas que permitem aos processos isolados escaparem da barreira de segurança. Também já foi utilizada a instrumentação dinâmica do código em execução. Versões futuras do Condor poderão incorporar o uso de máquinas virtuais [VMCondor 2007].

4. Portifolio de Segurança

Soluções de segurança baseadas somente na confiança mútua não são viáveis para o OurGrid devido a sua natureza de livre entrada. Os usuários e recursos que fazem parte da grade não são signatários de qualquer acordo que possibilite a sanção de usuários que executarem ações maliciosas. Além disto, o sistema tem que suprir as necessidades de segurança de diversos perfis de usuários. Motivos como estes fazem com que um portifolio de segurança seja necessário.

Existem duas classes de problemas de segurança em grades: (i) a segurança dos usuários da grade contra recursos maliciosos; (ii) e a segurança dos recursos da grade

contra usuários maliciosos. Desta forma, cada componente do OurGrid apresenta desafios de segurança característicos de sua funcionalidade.

O Broker (usuário) deve ser responsável por validar e analisar os resultados obtidos pelos Workers (recursos), tentando mitigar qualquer sabotagem originada de um Worker malicioso. Neste caso, o OurGrid utiliza soluções de tolerância a sabotagem. O Worker, por sua vez, deve se proteger contra o uso indevido do recurso. No OurGrid, isto é obtido através do uso de soluções de *sandboxing*. Por terem funcionalidades distintas, as soluções de segurança no Broker são transparentes ao Worker e vice-versa, permitindo que cada componente adote soluções independentes.

Ademais, todos os componentes da grade precisam ter mecanismos para garantir que uma entidade não consiga se passar por outra. Isto é uma necessidade principalmente para os Peers, que são responsáveis pelo controle de recursos no sistema, nos quais uma identidade forjada pode burlar este controle. O OurGrid faz uso de técnicas de autenticação e autorização para garantir que os pares não tenham identidades forjadas.

No restante desta seção serão descritas as soluções de segurança que fazem parte do middleware OurGrid.

4.1. Worker: execução em máquinas virtuais

O conceito de virtualização, considerando o contexto deste trabalho, pode ser definido como a divisão dos recursos de um computador (hospedeiro) em diversos ambientes de execução virtuais (hóspedes ou máquinas virtuais). Soluções de virtualização são implantadas para melhor utilizar os recursos de servidores com múltiplos ambientes configurados para fins específicos, para execução de aplicações legadas, ambientes de teste de softwares, isolamento de recursos computacionais, etc [VMWare 2007]. Em um ambiente de grades computacionais, a virtualização é uma ferramenta de alta aplicabilidade. Devido a heterogeneidade dos recursos, o uso da virtualização pode fazer com que aplicações desenvolvidas para uma plataforma específica sejam executadas em outras dentro de uma máquina virtual. Também se pode fazer uso de virtualização para facilitar a gerência da grade e oferecer soluções de tolerância a faltas. Como argumentado na Seção 3, virtualização também pode ser utilizada para garantir o isolamento de tarefas em uma grade, aumentando a segurança do sistema. Existem diversos tipos de virtualização, cada um apresenta vantagens e desvantagens [Jones 2007]. Estes são descrito abaixo. Apresentamos na Figura 2 uma classificação dos tipos, considerando desempenho e nível de abstração.

Emulação Nesta abordagem a máquina hóspede executa dentro de um emulador de hardware. Com o uso de emulação cada instrução executada dentro do emulador é traduzida para uma ou mais instruções da máquina hospedeira. A maior vantagem da emulação é a capacidade de executar máquinas virtuais com arquiteturas de hardware diferentes da arquitetura da máquina hospedeira. Devido à necessidade de traduzir todas as instruções, a emulação tem um desempenho baixo em relação aos outros tipos.

Virtualização Total Virtualização total faz uso de um monitor de máquinas virtuais que pode executar instruções privilegiadas na máquina hospedeira. Uma máquina hóspede faz uso do monitor para executar as instruções feitas pela máquina hospedeira: caso sejam privilegiadas o monitor faz a devida tradução; caso não sejam,

o monitor executa a instrução direto no hardware. Embora tenha um desempenho melhor do que a emulação, esta abordagem é limitada já que necessita que as máquinas hóspede e hospedeira sejam da mesma arquitetura.

Paravirtualização Fazendo com que o sistema operacional executado na máquina hóspede seja consciente que está sendo executado em um ambiente virtualizado, é possível que o monitor de máquinas virtuais implemente um melhor mecanismo de tradução das chamadas feitas na máquina hóspede. Com isto, é possível alcançar um melhor desempenho ao custo do uso de sistemas operacionais modificados.

Virtualização no nível do Sistema Operacional Neste tipo de virtualização máquinas hóspedes do sistema usam o mesmo sistema operacional da hospedeira. Neste caso, o sistema operacional é capaz de limitar o acesso aos recursos feitos pela máquina hóspede, criando compartimentos isolados de execução. A maior vantagem deste tipo de virtualização é a pequena carga extra, pois não existe indireção com relação às instruções executadas na máquina hóspede.

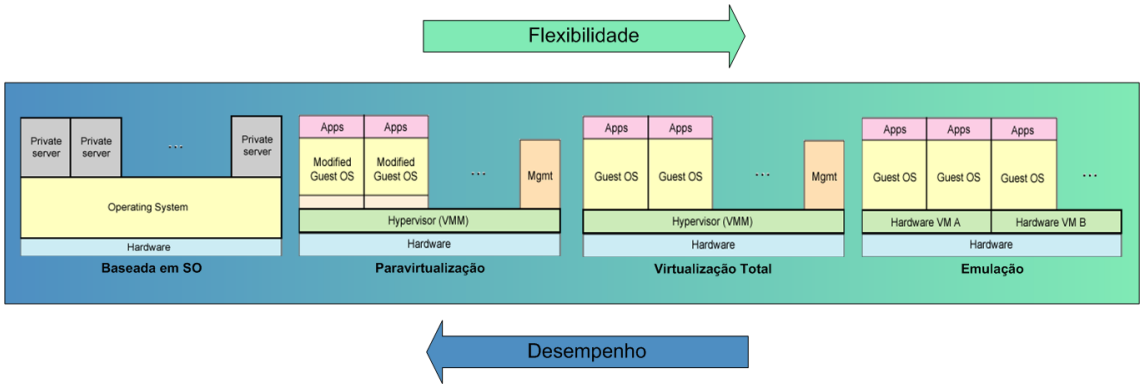


Figura 2. Diferentes tipos de Virtualização

No OurGrid optamos por criar ambientes seguros de execução de tarefas fazendo uso de virtualização, em especial utilizamos um recipiente de tarefas sem acesso à rede [Santhanam et al. 2005][1]. A Figura 3 esquematiza a seqüência de passos da execução de uma tarefa no compartimento seguro de execução. Antes do início da tarefa, toda entrada necessária é copiada do Broker para uma pasta compartilhada, que é acessível pela máquina virtual. Durante a execução da tarefa, a aplicação não tem maneira de se comunicar com os processos da máquina hospedeira nem tem acesso à rede, garantindo assim o seu isolamento. Após o término da aplicação, os dados de saída são copiados da pasta compartilhada para o Broker.

Para suprir a demanda de diferentes usuários, o OurGrid provê compartimentos seguros de execução baseados em diversas tecnologias de virtualização: QEMU[2] para oferecer emulação; VirtualBox[3] para virtualização total em computadores com arquitetura x86; XEN[4] que oferece paravirtualização também na arquitetura x86; e por fim, VServer[5]

[1] Os autores denominam este tipo de sandboxing de *Eager prefetching, whole-file-caching sandboxes*
[2] http://www.fabrice.bellard.free.fr/qemu/
[3] http://www.virtualbox.org/
[4] http://www.xensource.com/
[5] http://www.linux-vserver.org/

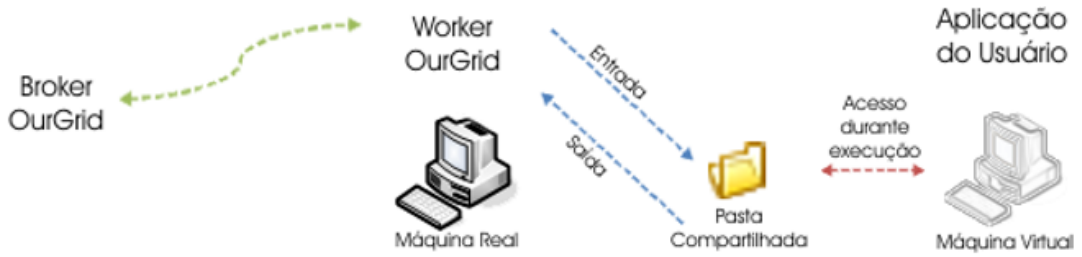


Figura 3. Esquema de funcionamento dos Compartimentos Seguros de Execução

para virtualização no nível do sistema operacional para Linux. O administrador de um domínio, no processo de implantação de um *Peer*, precisa tão somente escolher a versão do *Worker* diponibilizada com a tecnologia que achar mais adequada às restrições do seu domínio.

4.2. Broker: tolerância a sabotagem

Para proteger as aplicações dos usuários da grade contra recursos maliciosos, o OurGrid faz uso de uma solução de tolerância a sabotagem com uso de marcas d'água (também conhecido como códigos de certificação) similar à solução descrita em [Collberg and Thomborson 2002].

De forma resumida, o usuário da grade modifica sua aplicação adicionando uma marca d'água, esta introduz uma forma de identificar se o resultado da aplicação foi alterado. Um exemplo simples deste método é uma aplicação que computa uma série funções de saída não conhecida, dentre estas funções o usuário pode adicionar uma nova função de saída conhecida. Após o término da execução a saída da função conhecida é verificada para determinar se houve uma sabotagem. No OurGrid, o usuário adiciona à descrição de sua tarefa uma etapa adicional que contém o procedimento para verificação da marca d'agua, esta verificação é executada pelo Broker, o componente responsável pelo escalonamento de tarefas, que pode banir, por exemplo, um recurso tido como malicioso das próximas execuções.

Ademais, o portifolio de segurança também inclui a solução já existente de tolerância a sabotagem independente de aplicação [Oliveira and Brasileiro 2006], a qual exige um alto custo computacional, pois exige que para cada tarefa, réplicas sejam criadas e escalonadas em recursos confiáveis, a fim de comparar os resultados das mesmas tarefas escalonadas em nós desconhecidos. Esta solução pode ser adotada, se os requisitos da aplicação que está sendo executada na grade justificarem o custo computacional.

4.3. Autenticação e Autorização

O OurGrid faz uso de um mecanismo de incentivo para a troca de recursos [Andrade et al. 2007]. Nesse mecanismo, cada Peer contabiliza a doação dos seus recursos aos outros pares e o seu consumo dos recursos dos outros pares. Portanto, é preciso que cada um dos Peers seja identificado unicamente no sistema. Se for utilizado

um identificador público na rede do OurGrid, uma entidade maliciosa pode facilmente se passar por outro Peer. É preciso garantir que cada interação entre-pares tenha sua origem confirmada e associada ao identificador único de cada Peer, impedindo que mensagens forjadas, alteradas ou repedidas possam ser aceitas como válidas por qualquer componente do OurGrid.

Existem diversos mecanismos de autenticação que buscam contemplar tais funcionalidades, entretanto, alguns requerem o uso de servidores, como o Kerberos [Kohl and Neuman 1993], ou mesmo da configuração e uso de uma entidade certificadora para controlar a troca de identificações no sistema [Adams and Farrell 1999]. No OurGrid, usamos uma solução mais simples, próxima ao SPKI [Ellison et al. 1999], onde uma chave pública (de um par de chaves assimétricas [Kaliski 1998]) é a própria identidade de cada participante. Isto permite que o OurGrid mantenha suas características de livre entrada e fácil implantação. Note que cada Peer é autônomo para gerar sua identidade e o importante é que esta seja a mesma em todas as interações entre-pares do sistema. Uma característica inerente ao OurGrid é de que o mecanismo de incentivo garante que a troca de identidade de um Peer não é vantajosa, de modo que os Peers têm interesse em manter sua identidade.

Este mecanismo de segurança é implementado na camada de comunicação do OurGrid, o JIC. Cada mensagem trocada no sistema é assinada digitalmente através da chave privada e recebe em anexo a chave pública em questão, que serve para identificar o emissor da mensagem. O receptor por sua vez verifica a assinatura e, caso seja válida, pode então usar a chave pública para identificar a entidade emissora. O JIC usa chaves RSA [Jonsson and Kaliski 2003] de 1024 bits, pois é garantida, atualmente, a insuficiência de poder computacional para quebrar tal tipo de chave. Isto garante integridade e autenticação de cada mensagem. Usuários que desejem um maior nível de segurança podem fazer uso de entidades certificadoras para associar as identidades do sistema às identidades dos usuários, mas isso é uma funcionalidade opcional, já que dificulta a implantação do software.

Outras melhorias de segurança surgem ao se expandir o uso do identificador aos demais componentes do sistema. Por exemplo, pode-se garantir que um Worker só vai aceitar requisições do Broker ao qual foi alocado. Neste caso, ao alocar o Worker, o Peer envia uma mensagem para o mesmo com a identificação (chave pública) do Broker que vai utilizar os seus recursos. O Worker por sua vez só aceita pedidos de mensagens assinadas daquele Broker.

4.4. Assegurando o Código do Middleware

É necessário assegurar o código do middleware OurGrid contra ataques, como também bibliotecas e *containers* de terceiros utilizados na sua construção. Ataques ao código da aplicação, como *Buffer-Overflow*, são comuns e podem eventualmente prover acesso privilegiado para um atacante. Por outro lado, já descrevemos como o uso de virtualização limita as aplicações executadas na grade, agora também levantamos a necessidade de limitar os recursos utilizados pelos componentes do OurGrid para compor a grade. Caso um atacante consiga controlar de alguma maneira um dos componentes do OurGrid, este atacante pode fazer uso dos recursos que são providos para o componente.

O código do OurGrid é desenvolvido na linguagem Java, que é interpretada por

uma Máquina Virtual Java (*Java Virtual Machine ou JVM*). A linguagem Java e a JVM oferecem segurança no nível do código da aplicação [Sun Microsystems 2005]. Java também provê um conjunto de bibliotecas de segurança que permitem ao programador implementar diversos mecanismos para o desenvolvimento de aplicações seguras. Estas bibliotecas também são utilizadas pelo OurGrid como, por exemplo, os provedores de segurança.

Informações e detalhes sobre como a linguagem Java e seu interpretador provêem segurança podem ser encontradas na Internet no sítio sobre segurança em Java [Sun Microsystems 2007], em *white-papers* [Sun Microsystems 2005], como também na especificação da linguagem [Gosling et al. 2005] e da JVM [Lindholm and Yellin 1999].

5. Conclusão

Neste trabalho apresentamos um portifolio de segurança para um sistema de grade entre pares de livre entrada. Para definir o portifolio, fizemos um levantamento sobre grades computacionais, quais mecanismos de segurança são utilizados e são aplicados. Com este estudo foi possível levantar as necessidades de segurança do OurGrid e definir quais mecanismos de segurança devem fazer parte da sua solução. De fato, não existe uma solução única que atenda a todos os usuários de grades entre pares de livre entrada, portanto deve ser provido um portifolio com várias soluções de segurança.

Embora o portifolio implementado no OurGrid possa ser considerado abrangente, ainda pode ser expandido. Por exemplo, incorporar virtualização no nível do *hardware* [Habib 2008]. Esta tecnologia, a primeira das soluções de virtualização incorporadas no kernel estável do linux, implementa virtualização total se beneficiando da aceleração causada pelo suporte do processador.

Durante o desenvolvimento das soluções descritas nos deparamos com uma dificuldade adicional, a gerência de máquinas virtuais. A implantação e manutenção das imagens apresenta um alto custo administrativo. Uma possível solução para este problema seria a implantação de *VirtualWorkspaces* [Keahey et al. 2005] no OurGrid. Este serviço, além de facilitar a gerência das máquinas virtuais, expõe um protocolo de configuração que poderia ser usado pelo usuário da grade na criação de ambientes específicos para sua tarefa, por exemplo, com pacotes e utilitários que atendam de forma mais satisfatória suas submissões.

Agradecimentos

Este trabalho foi desenvolvido em colaboração com a HP Brasil P & D. Francisco Brasileiro é pesquisador do CNPq-Brasil.

Referências

Adams, C. and Farrell, S. (1999). RFC 2510: Internet X.509 public key infrastructure certificate management protocols. Request for Comments (RFC) 2510, Network Working Group.

Andrade, N., Brasileiro, F., Cirne, W., and Mowbray, M. (2007). Automatic grid assembly by promoting collaboration in peer-to-peer grids. *Journal of Parallel and Distributed Computing*, 67(8):957–966.

Boinc Project (2007). Boinc project website, http://boinc.berkeley.edu/.

Boinc Status (2007). Boinc status, http://boincstats.com/stats/project_graph.php?pr=sah.

Cappello, F., Djilali, S., Fedak, G., Hérault, T., Magniette, F., Néri, V., and Lodygensky, O. (2005). Computing on large-scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid. *Future Generation Comp. Syst*, 21(3):417–437.

Cavalcanti, E., Assis, L., Gaudencio, M., Cirne, W., Brasileiro, F., and Novaes, R. (2006). Sandboxing for a free-to-join grid with support for secure site-wide storage area. In *1st International Workshop on Virtualization Technology in Distributed Computing*.

Collberg, C. S. and Thomborson, C. (2002). Watermarking, tamper-proofing, and obfuscation - tools for software protection. *Software Engineering, IEEE Transactions on*, 28(8):735–746.

Distributed.net (2007). Distributed.net project, http://www.distributed.net/.

Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., and Ylonen., T. (1999). RFC 2693: Spki certificate theory. Request for Comments (RFC) 2693, Network Working Group.

Engine, S. G. (2007). Sun grid engine project web page, http://gridengine.sunsource.net/.

Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. (1998). A security architecture for computational grids. In *ACM Conference on Computers and Security*, pages 83–91. ACM Press.

Foster, I. T. and Kesselman, C. (1998). The globus project: A status report. In *Heterogeneous Computing Workshop*, pages 4–18.

Frey, J., Tannenbaum, T., Livny, M., Foster, I. T., and Tuecke, S. (2002). Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5(3):237–246.

gLite (2007). glite project web page, http://glite.web.cern.ch/glite/.

Gosling, J., Joy, B., Steele, G., and Bracha, G. (2005). *Java(TM) Language Specification, The (3rd Edition) (Java Series)*. Addison-Wesley Professional.

Habib, I. (2008). Virtualization with kvm. *Linux J.*, 2008(166):8.

Jones, M. T. (2007). Virtual linux, http://www.ibm.com/developerworks/library/l-linuxvirt/index.html.

Jonsson, J. and Kaliski, B. (2003). RFC 3447: Public-key cryptography standards (pkcs) #1: Rsa cryptography specifications version 2.1. Request for Comments (RFC) 3447, Network Working Group.

Kaliski, B. (1998). RFC 2315: Pkcs #7: Cryptographic message syntax. Request for Comments (RFC) 2315, Network Working Group.

Kamp, P. H. and Watson, R. N. M. (2000). Jails: Confining the omnipotent root. In *In Proceedings of the 2nd International SANE Conference*.

Kapadia, N. H., Figueiredo, R. J. O., and Fortes, J. A. B. (2000). PUNCH: Web portal for running tools. *IEEE Micro*, 20(3):38–47.

Keahey, K., Foster, I., Freeman, T., Zhang, X., and Galron, D. (2005). Virtual Workspaces in the Grid. *Lecture Notes in Computer Science*, 3648:421–431.

Kohl, J. T. and Neuman, B. C. (1993). The Kerberos network authentication service (V5), citeseer.ist.psu.edu/article/kohl93kerberos.html. Technical Report 1510.

Lima, A., Cirne, W., Brasileiro, F., and Fireman, D. (2006). A case for event-driven distributed objects. In *8th International Symposium on Distributed Objects and Applications (DOA)*, pages 1705–1721, Berlin / Heidelberg. Springer.

Lindholm, T. and Yellin, F. (1999). *The Java(TM) Virtual Machine Specification (2nd Edition)*. Prentice Hall PTR.

Litzkow, M. J., Livny, M., and Mutka, M. W. (1988). Condor - A hunter of idle workstations. In *ICDCS*, pages 104–111.

Neary, M. O., Christiansen, B. O., Cappello, P. R., and Schauser, K. E. (1999). Javelin: Parallel computing on the internet. *Future Generation Comp. Syst*, 15(5-6):659–674.

Oliveira, A. C. and Brasileiro, F. (2006). Escalonamento tolerante a sabotagem para grades computacionais entre-pares. In *VII Workshop de Testes e Tolerância a Falhas (WTF) in conjunction with Simpósio Brasileiro de Redes de Computadores (SBRC)*.

Ong, T. M., Lim, T. M., Lee, B. S., and Yeo, C. K. (2002). Unicorn: voluntary computing over Internet. *Operating Systems Review*, 36(2):36–51.

OpenPBS (2007). Open source workload management software, http://www.openpbs.org/.

Santhanam, S., Elango, P., Arpaci-Dusseau, A., and Livny, M. (2005). Deploying virtual machines as sandboxes for the grid. In *WORLDS'05: Proceedings of the 2nd conference on Real, Large Distributed Systems*, Berkeley, CA, USA. USENIX Association.

Sarmenta, L. F. G. (1998). Bayanihan: Web-based volunteer computing using Java. *Lecture Notes in Computer Science*, 1368.

Sarmenta, L. F. G. (2001). Sabotage-tolerance mechanisms for volunteer computing systems. In *CCGRID*, pages 337–346. IEEE Computer Society.

Sun Microsystems (2005). Java security overview - white paper, http://java.sun.com/developer/technicalarticles/security/whitepaper/js_white_paper.pdf.

Sun Microsystems (2007). Java se security website, http://java.sun.com/developer/technicalarticles/security/whitepaper/js_white_paper.pdf.

TeraGrid (2007). Teragrid project web page, http://www.teragrid.org/.

Thain, D., Tannenbaum, T., and Livny, M. (2005). Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356.

TOP 500 (2007). Top 500 supercomputer sites, http://www.top500.org/.

VMCondor (2007). Virtual machines in condor, http://www.cs.wisc.edu/condor/pcw2007/condor_presentations.html/.

VMWare (2007). Virtualization overview, http://www.vmware.com/files/elqnow/elqredir.htm?ref=http://www.vmware.com/pdf/virtualization.pdf.

GPOL: Uma Linguagem para Workflows de Serviços em Grades Computacionais

Carlos R. Senna e Edmundo R. M. Madeira

Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6.176 – 13.083-970 – Campinas – SP – Brasil

{carlos.senna,edmundo}@ic.unicamp.br

Abstract. *The Computational Grids collaborative environment makes easy the users interaction, but requests an additional level of workflow management. This article presents the GPOL (Grid Process Orchestration Language), a compact user-oriented language which allows the user to describe the relationship between strong-coupled tasks and services. The GPOL's workflows use the Web service orchestration concepts applied on the grid services.*

Resumo. *O ambiente colaborativo das grades computacionais facilita a interação entre os usuários, mas requer um nível adicional de gerência de workflow. Esse artigo apresenta a GPOL (Grid Process Orchestration Language), uma linguagem compacta orientada ao usuário, que permite descrever o relacionamento entre tarefas e serviços fortemente acoplados. Os workflows GPOL usam o conceito de orquestração de serviços Web aplicado aos serviços das grades computacionais.*

1. Introdução

A Internet e seus padrões abertos acrescidos das facilidades disponíveis nos sistemas distribuídos permitem que as organizações possam compartilhar recursos, desde a fase inicial de desenvolvimento de produtos até as etapas finais do processo de negócio. A união desses conceitos e tecnologias resultou no aparecimento das *Organizações Virtuais* que são multi-institucionais e dinâmicas [1] tendo a grade computacional [2, 3] como uma ferramenta adequada para a sua implementação. Em uma organização virtual, *serviço* passa a ser não só a base das aplicações como também a peça fundamental do processo de colaboração entre os participantes da organização.

Esse novo ambiente torna os padrões atuais de *workflow* limitados para suportar a interação entre os participantes nas organizações virtuais, notadamente no que se refere à coordenação e *composição* dinâmica de processos e serviços requeridos por essas organizações. Para permitir um desenvolvimento simples e extensível torna-se importante um mecanismo para *workflow* controlável pelo usuário.

Esse artigo apresenta a GPOL (*Grid Process Orchestration Language*), uma linguagem compacta que permite ao usuário descrever o relacionamento entre tarefas e serviços fortemente acoplados. Através do uso da GPOL o usuário pode controlar com eficiência a execução de tarefas distribuídas, principalmente as que tem alto grau de inter-relacionamento.

A GPOL é a linguagem para descrição de *workflows* que faz parte de uma infra-estrutura para orquestração de serviços e processos em grades computacionais denominado GPO (*Grid Process Orchestration*). O GPO é um *middleware* para interoperabilidade de aplicações distribuídas que requerem composição de serviços em uma grade computacional [4].

O GPO permite a criação e gerência de fluxos de aplicações, tarefas e principalmente serviços das grades computacionais. No projeto do GPO foram consideradas aplicações fortemente acopladas cujas tarefas apresentam dependências entre si, precisam se comunicar para que a aplicação faça progresso, em geral requerem processamento paralelo e que tipicamente são modeladas como um grafo acíclico direcionado (DAG - *directed acyclic graph*). A arquitetura GPO consiste de três componentes principais, *GPO Run*, *GPO Maestro Factory Service* e *GPO Maestro Service Instance*, além de arquivos especificados com GPOL. A Figura 1 mostra esses componentes.

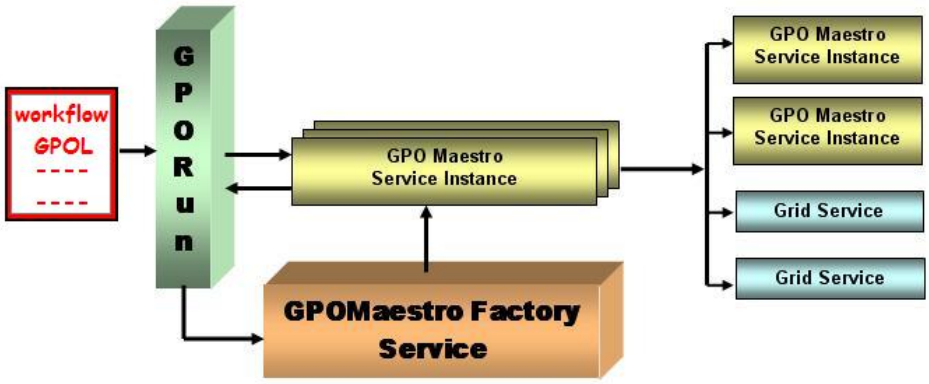


Figure 1. A arquitetura GPO e seus componentes

O *GPO Run* é um utilitário independente que recebe arquivos escritos em GPOL, verifica sua correção sintática, e inicia a execução do *workflow* recebido.

Para executar o *workflow*, o *GPO Run* usa a "*fábrica de serviço*" *GPO Maestro Factory Service* para criar uma nova instância do *GPO Maestro* que fica responsável pela execução do *workflow*. A instância do *GPO Maestro* processa o *workflow* retornando um código indicando o resultado da execução. O *GPO Run* destrói a instância do *GPO Maestro*, entregando o resultado ao usuário e finalizando o processo.

O GPO usa os conceitos das grades computacionais baseadas na arquitetura OGSA (*Open Grid Services Architecture*) [3] e acrescenta a essas grades, através da GPOL, facilidades para orquestração de serviços. A seguir na Seção 2 o artigo discute sobre gerência de *workflow* em grades computacionais e os trabalhos relacionados. Na Seção 3 descreve a linguagem para submissão de tarefas GPOL, na Seção 4 apresenta um exemplo de aplicação e na Seção 5 mostra a conclusão.

2. Gerência de *workflow* em grades computacionais

O ambiente heterogêneo provido pelas grades computacionais torna a gerência de *workflow* mais complexa, sendo necessário, por exemplo, gerenciar as múltiplas políticas de segurança com regras definidas pelas *organizações virtuais* participantes da grade. Em grades orientadas a serviços como OGSA é necessário um novo conceito

para gerenciar diretamente os serviços da grade sem a necessidade da construção de aplicações cliente para usá-los em sua plenitude.

A WfMC [5] define um *sistema de workflow* como sendo: *"Um sistema que define, cria e gerencia a execução de workflows através do uso de um software, executando um ou mais motores de workflow, capazes de interpretar a definição do processo, interagindo com os workflows participantes, invocando o uso de ferramentas e aplicações"*. A definição destaca a necessidade de uma linguagem apropriada que descreva o *processo de negócio* e permita expor claramente as *regras e procedimentos* que devem ser respeitados.

2.1. Sistemas de Gerência

A comunidade de grades tem proposto várias soluções de propósito geral para gerência de *workflow*. *Triana* [6] oferece um modelo para programação visual a partir de módulos pré-definidos. Em [7] são discutidos e propostos algoritmos para composição de serviços e escalonamento [8] direcionado a *workflow* em grades. Em [9] é apresentada a *Grid Workflow Infrastructure*, baseada em OGSA e que usa os conceitos de WS-BPEL (*Web Services Business Process Execution Language*). Outros trabalhos são destacados a seguir:

GridFlow: é um sistema de gerência de *workflow* baseado em agentes [10];

GridAnt: faz orquestração de atividades e suas dependências expressando-as em XML [11];

WebFlow: é um sistema multi-nível para computação distribuída com arquitetura em três camadas [12]. A camada superior é uma ferramenta baseada na *Web* para programação visual e monitoramento.;

Condor: Condor é um *batch queueing system* [13]. Reuni as funções de um escalonador, um gerente de recursos e um sistema de balanceamento de carga. Outro componente do projeto Condor é o meta-escalonador DAGMan (*Directed Acyclic Graph Manager*), um serviço para execução de múltiplas tarefas com dependências, na forma declarativa; e

CoG Kits: não é um sistema de gerência de *workflow*, mas um conjunto de ferramentas de *software* (*open source*) com *frameworks* de alto nível que permitem usar, desenvolver e administrar grades [14].

2.2. Linguagens

A escolha ou definição de uma linguagem para que o usuário possa descrever os *workflows* é fundamental. A linguagem deve descrever o *processo de negócio* e expor claramente as *regras e procedimentos* que devem ser respeitados. A seguir são apresentadas algumas linguagens propostas para *workflow*.

GSFL: *Grid Services Flow Language* [15] é uma linguagem baseada em XML que permite descrever *workflows* envolvendo serviços em grades baseadas em OGSA, cuja arquitetura é composta por um conjunto de modelos;

JSDL: *Job Submission Description Language* é uma especificação do OGF para descrever os requerimentos de uma tarefa computacional para submissões em recursos das grades [16];

Karajan: parte integrante do *Java CoG Kit* [14], é um *framework* composto de uma linguagem e um motor de *workflow*. A linguagem projetada oferece uma sintaxe simplificada para gerar *scripts* ou permite o uso de XML; e

XML-based Description Language: é uma linguagem e um ambiente de execução para orquestração de tarefas em grades [17]. A linguagem oferece suporte a vários tipos de tarefas como execução computacional, transferência de arquivos, invocação de serviços, e uso de *scripts* e objetos *Java*.

3. *Grid Process Orchestration Language* (GPOL)

As grades baseadas em OGSA com WSRF (*Web Services Resource Framework*) [18] combinam Serviços *Web* e Serviços da grade, tornando esse ambiente apropriado tanto aos processos de negócios como também ao processamento científico. Nesse ambiente, qualquer que seja o tipo de processo, é importante a forma como os Serviços são compostos. Para fazer essa composição foi criada a GPOL.

A GPOL é uma linguagem compacta baseada em XML, que contém primitivas que permitem ao usuário, descrever o relacionamento entre tarefas e serviços fortemente acoplados, formando um *workflow* para ser usado em grades OGSA. GPOL une os conceitos de orquestração de serviços propostos em WS-BPEL [19] às facilidade oferecidas pela *Globus WS-GRAM Job Description*. Além disso, acrescenta diretivas para tratar requisitos específicos da grade, como: manutenção de estado, serviços potencialmente transientes, notificação, serviços orientados a dados e serviços orientados a grupos.

A GPOL herda de WS-BPEL o forte suporte a *Serviços Web* e usa a notação *WS-Addressing* de pontos de referência (EPR), para simplificar a passagem de endereços entre os Serviços participantes do *workflow*. Essa mesma notação é usada nas grades baseadas em OGSA com WSRF para endereçar os *Serviços da Grade*.

Na GPOL optou-se por criar uma linguagem simplificada e não usar a especificação WS-BPEL nos *workflows*. Essa escolha está relacionada ao objetivo central da WS-BPEL, que é um "*processo de negócio*" e pode envolver vários participantes. Para atingir esse objetivo WS-BPEL usa troca de mensagens (<*message*>) e estabelece ligações (<*partnerLink*>) entre os parceiros participantes.

O objetivo do usuário da grade é executar tarefas e usar serviços. O usuário estabelece no *workflow* as regras para a execução das tarefas e dos serviços. Não existem usuários participantes e a necessidade de estabelecer ligações, ou troca de mensagens entre eles. Além disso, a GPOL atende requisitos específicos das grades, como por exemplo, transiência, ciclo de vida, transferência de arquivos e execução de tarefas. A GPOL oferece um conjunto completo de facilidades.

- Ela tem tratamento de variáveis, recurso importante na manutenção das informações entre os serviços (<*variables*>, <*assign*>, <*clear*>);
- Oferece as principais diretivas para controle de fluxo (<*if*>, <*while*>);
- Permite execuções seqüenciais e paralelas (<*sequence*>, <*flow*>). Isso possibilita o uso eficiente dos recursos disponíveis na grade, inclusive com facilidades para a descoberta de recursos similares em grades oportunistas;

- Transferência de arquivos (<*transfer*>) e execução de tarefas (<*execute*>);
- Permite invocar serviços da grade (<*invoke*>) diretamente no *workflow*, não sendo obrigatório o uso de programas clientes para fazê-lo;
- Aceita código *Java* no *workflow* (<*javacode*>). Esse recurso usado em conjunto com o elemento <*import*>, permite ao usuário criar e agregar novas facilidades ao *middleware* proposto;

A GPOL é compacta reunindo as principais facilidades apresentadas pelas outras linguagens. Como diferenciais a GPOL oferece:

- Tratamento para variáveis e recursos para execução de tarefas e invocar serviços (JSDL não trata serviços);
- Estruturas para controle de fluxo (JSDL e *XML-based Description Language* não oferecem);
- Diretivas para que o usuário possa usar o conceito de fábrica/instância dos serviços da grade (WS-BPEL não oferece esse recurso);
- Endereçamento de serviços (*Endpoint Reference - WS-Addressing*) conforme a especificação WSRF (JSDL e *XML-based Description Language* não oferecem); e
- Aceita código Java como parte do *workflow* (WS-BPEL não oferece esse recurso).

Além disso, a GPOL é direcionada ao usuário da grade (mesmo iniciantes), usando diretivas e expressões (dialeto) mais coerentes com esse ambiente, como por exemplo, *provider*, *transfer* e *host*. Outras linguagens, como WS-BPEL e GSFL por exemplo, são direcionadas a participantes em processos de negócios, e usam expressões que não são comumente usadas nas grades (*partner*, *controlLink*, etc).

3.1. A estrutura de um *workflow* GPOL

Em uma visão geral, um *workflow* GPOL é similar a um *processo de negócio* WS-BPEL. Ele é composto por dois elementos: definições (<*definitions*>) e processo (<*process*>). Nas *definições* são feitas as declarações das variáveis e identificados os serviços que serão usados na execução do *workflow*. No *processo* é definida a ordem na qual os serviços são acionados, como tratar seus retornos, o que fazer em caso de falha, indicar a destruição de instâncias e indicar o valor de retorno após a execução. Ou seja, as definições indicam o que será usado, e o processo indica como será usado.

A Figura 2 mostra um *workflow* que apresenta a mensagem "GPOL – Hello!" na saída padrão escrito em GPOL.

```
<workflow name="Hello">
 <definitions name="HelloDefs">
  <variables>
   <variable name="sMsg" type="string"
          value="GPOL - Hello!"/>
  </variables>
 </definitions>
 <process name="HelloProcess">
  <execute>
    executable="/bin/echo"
    <argument variable="sMsg"/>
  </execute>
 </process>
</workflow>
```

Figure 2. Exemplo de *workflow* em GPOL

Uma submissão GPOL usa como *raiz* do documento XML, o elemento <*workflow*>. No exemplo da Figura 2, o *workflow* definido é chamando de "*Hello*". Ele contém a variável "*sMsg*", do tipo "*string*", cujo valor inicial é a mensagem "GPOL - Hello!". O elemento <*variables*> está inserido nas definições <definitions>. O processo do *workflow* (<*process*>) tem um único elemento <*execute*>. Em <*execute*> está especificado que deve ser executado o utilitário *echo* (*/bin/echo*), cuja *string* argumento é o conteúdo da variável *sMsg*, a mensagem "*GPOL - Hello!*".

3.2. Elementos da linguagem

Em um *workflow* GPOL a área de definições pode conter os elementos <*variables*> e <*services*>, e o processo pode conter qualquer combinação das atividades GPOL (<*invoke*>, <*assign*>, <*if*>, <*while*>, <*clear*>, <*sequence*>, <*flow*>, <*return*>, <*exit*>, <*echo*>, <*execute*>, <*transfer*> e <*javacode*>). A seguir são apresentados detalhes de alguns desses elementos.

workflow: é o elemento raiz do documento GPOL. Ele pode conter atributos com *targetNamespace* (*targetNamespace*="*xsd:anyURI"*) e aceita outras indicações de esquemas (*XML Schema*);

import: é usado para declarar uma dependência que está em um arquivo externo. É possível indicar um *namespace* (<*import namespace* ="*xsd:anyURI"*>) ou um arquivo (<*import file* ="*xsd:string"*>);

service: A GPOL permitir ao usuário invocar serviços da grade diretamente no *workflow*. Para invocar um serviço, o usuário deve declará-lo em *definitions* e usá-lo em *process*. A definição do serviço informa sua localização, seu tipo e fornece um nome único para referenciá-lo no processo. Essa definição é feita através do elemento *service*, e várias definições podem estar agrupadas no elemento *services*. O elemento *service* tem como atributos: *name*, *host*, *uri* (*endpoint reference*), *fal* (*factory addressing locator*), *sal* (*service addressing locator*), *type*, *gsh*, *provider* e ret-type. O atributo *name* indica o nome (apelido) de cada serviço, que deve ser único dentro do *workflow*, e que será usado para invocação no processo. O atributo *type* indica qual o tipo de serviço a ser invocado e pode ser *Factory* ou *Instance*. Esse atributo está relacionado com o conceito de "*Fábrica de Serviços*" usado nas grades baseadas em OGSA. Para o tipo *Factory* é criada uma nova instância desse serviço para ser usada durante a execução do processo.

invoke: é usado para invocar uma operação de um serviço da grade permitindo o uso dos atributos *service*, *operation*, *argument*, *return* e *faultHandlers*. O atributo *service* indica o serviço alvo e *operation* a operação desejada desse serviço. Os elementos *argument* e *return* estão associados logicamente ao atributo *operation*. O elemento *argument* deve ser usado para definir argumentos a serem passados para a operação do serviço e *return* indica o que deve ser feito com o valor retornado pela operação.

assign: A GPOL permite ao usuário criar variáveis de *workflow*. Essas variáveis servem para manter o estado do *workflow*, armazenando valores e resultados intermediários que são usados em cálculos ou como argumentos em serviços e tarefas. Para atribuir valores a essas variáveis é usado o elemento *assign*.

flow: permite ao usuário usar os recursos computacionais da grade simultaneamente, disparando a execução de tarefas ou executando serviços em cada um deles. Para cada

atividade é criada uma *thread* responsável pelo seu processamento, garantindo a execução independente mesmo quando for invocado um serviço da grade, pois o elemento *invoke* tem processamento síncrono quando usado fora de um *flow*. Após disparar a execução de todas as atividades relacionadas, o *flow* entra em estado de espera aguardando o término de todas as atividades, para só então, permitir a continuidade do processo. Ou seja, cada atividade é executada de forma assíncrona e o final do *flow* é usado como ponto de sincronização até que todas as atividades sejam executadas e as dependências resolvidas para então, dar continuidade na execução do processo. Qualquer atividade pode fazer parte de um *flow*, inclusive *sequence* e o próprio *flow*.

sequence: permite ao usuário indicar que as atividades devem ser processadas sequencialmente conforme a sua ordem. Isso permite ao usuário sinalizar execuções seqüenciais dentro de atividades para execução em paralelo (*flow*).

transfer: aplicações que usam grandes volumes de dados são comuns nas grades. Antes de iniciar um processamento em um recurso remoto, é necessário garantir o acesso aos dados usados nesse processamento. Uma das formas de disponibilizar os dados é transferir o(s) arquivo(s) que os contém para o *host* onde será feito o processamento. Uma das maneiras de transferir arquivos em GPOL é através do elemento *transfer*. O elemento *transfer* usa dois conjuntos de atributos (*srcHost*, *srcFile, srcProvider e dstHost*, *dstFile*, *dstProvider*), para indicar a origem e o destino do arquivo na transferência.

execute: Uma parte significativa do processamento feito nas grades envolve aplicações BoT (*Bag of Tasks*), que são caracterizadas pela independência das tarefas relacionadas na aplicação. O elemento *execute* dispara a execução de tarefas (*jobs*) e possui os atributos *executable*, *host* e *provider*, e permite o uso dos elementos *argument* e *return*.

javacode: contém código fonte *Java*. Esse recurso pode ser usado para pequenos cálculos, extração de *strings* ou acessar fragmentos de arquivos, por exemplo. O seu atributo *file* pode indicar o arquivo onde está o código a ser executado.

Maiores detalhes sobre a linguagem podem ser obtidos em [20].

4. Um exemplo de *workflow* GPOL

Para ilustrar as facilidades da GPOL será apresentado o *workflow* correspondente a uma aplicação envolvendo um fragmento dos cálculos feitos em uma seção de choque de fotoionização relativística por luz linearmente polarizada, aplicada em átomos alcalinos [21].

A fórmula da seção de choque, mostrada na Figura 3, usa duas somatórias. A somatória interna é formada pela multiplicação dos cálculos feitos pelos termos dos retângulos nomeados de *A* a *G*, onde cada iteração depende dos valores associados aos parâmetros *L, k, j* e *q*.

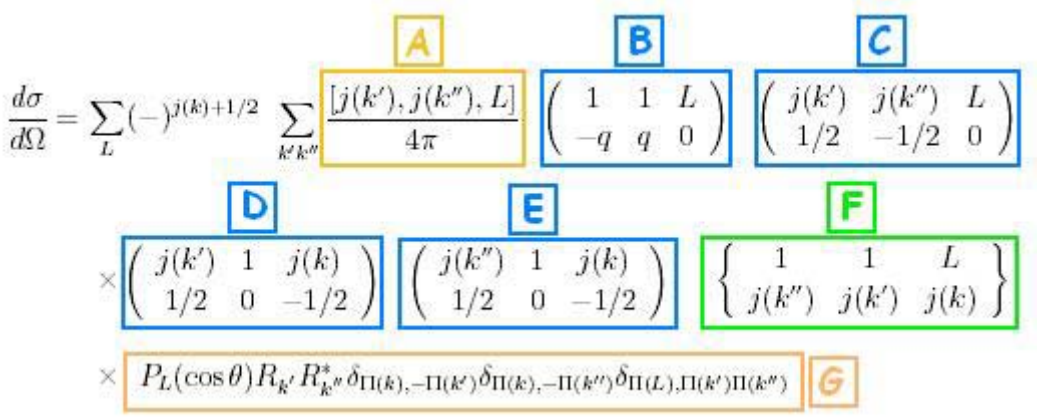


Figure 3. Fórmula da seção de choque de fotoionização de átomos relativísticos

As seções de choque são usadas de maneira ampla em estudos de Física em áreas como *Eletrônica Quântica* e *Física da Matéria Condensada*. Nessa aplicação o objetivo é a fotoionização de átomos relativísticos por incidência de luz linearmente polarizada. No processo de cálculo essa restrição é representada pelo parâmetro *q=0*. A medida da "*seção de choque relativística total*" é traduzida no cálculo pela integração dos seus parâmetros e, no caso aqui estudado, *L=0* contribui para essa medida. Os valores para o parâmetro *j* dependem das possibilidades do parâmetro *k*. São três possibilidades para *k* (*k, k' e k''*) que implicam em três possibilidades para *j* (*j(k), j(k') e j(k'')*).

4.1. A modularização da fórmula

Cada um dos termos da fórmula (de *A* a *G*) envolve uma teoria que gera uma seqüência específica de cálculos. Essas teorias são usadas em outros trabalhos e em outras áreas do estudo da Física e, portanto, são de aplicação geral. Os conceitos envolvidos em cada termo são:

Termo A: é um termo condicional que depende das condições de contorno do problema. Sua fórmula de cálculo é: [*j(k'), j(k''), L*] = (*2j(k')+1*)(*2j(k'')+1*)(*2L+1*);

Termos B, C, D e E: esses quatro termos calculam os "*coeficientes 3j de Clebsch-Gordon*". A variação está na matriz de parâmetros fornecida como argumento;

Termo F: esse termo é similar aos anteriores, mas usa uma outra fórmula para cálculo. Esse termo calcula o "*coeficiente 6j de Clebsch-Gordon*"; e

Termo G: esse termo está relacionado ao "*Polinômio de Legendre*", às *funções radiais dos átomos* e aos "*deltas de Kronecker*".

4.2. Visão geral da implementação do *workflow*

O *workflow* GPOL implementado faz a somatória das multiplicações dos termos de *A* a *F*, e o diagrama da Figura 4 mostra as dependências entre eles.

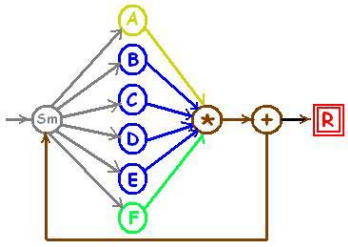


Figure 4. Diagrama com as dependências entre os termos da fórmula

A Figura 5 mostra uma visão geral do *workflow Fotoion*. O *workflow* começa definindo as variáveis e os serviços que serão usados (*FotoionDefs*) e seu processo (*FotoionProcess*) é um ciclo de leitura de tuplas, cujas informações são usadas nos cálculos dos termos de *A* a *F*. Cada linha do arquivo de tuplas contém valores para os parâmetros da fórmula, e estão na seguinte ordem: *L, q, k, k', k'', j(k), j(k') e j(k'')*.

```
- <workflow name="Fotoion">
  - <definitions name="FotoionDefs">
    + <variables></variables>
    + <services></services>
   </definitions>
  - <process name="FotoionProcess">
     <!-- Abre o arquivo com as tuplas -->
    + <invoke service="accessFile" operation="openFile"></invoke>
     <!-- Le primeira tupla -->
    + <invoke service="accessFile" operation="readLine"></invoke>
     <!-- Verifica a leitura feita -->
    + <invoke service="strFunctions" operation="isNull"></invoke>
    - <!-- Enquanto ler tuplas faz o calculo
          acumulando valor resultante -->
    + <while condition="iEoF eq 0"></while>
     <!-- Retorna o valor para o Cliente GPO -->
     <return variable="dSomatoria"/>
   </process>
</workflow>
```

Figure 5. Visão geral do *workflow Fotoion*

No *workflow* são usados quatro *Serviços da Grade*: *accessFile*, *strFunctions*, *CG3j* e *CG6j*.

accessFile: Todo o acesso ao arquivo de tuplas é feito através desse serviço. Ele oferece as operações *openFile*, *readLine* e *closeFile*.

strFunctions: é um conjunto de operações para tratamento e conversões de *strings*.

CG3j e ***CG6j***: calculam os "*coeficientes 3j e 6j de Clebsch-Gordon*" respectivamente e tem as operações *calc*, *getValue* e *clearValue*.

4.3. Detalhes do ciclo somatório

O ciclo que implementa o somatório pode ser definido logicamente como mostrado na Figura 6:

```
<while condition="iEoF eq 0">
  <!-- Prepara variaveis para o calculo. Obtem os valores de j(k), j(k') e j(k'')-->
  <flow>
    <sequence>
      <!-- Termo A ((2j(k') + 1)*(2j(k'') + 1)*(2L + 1))/4Pi -->
    </sequence>
    <invoke service= "CG3j" operation= "calc" ... </invoke> <!-- Termo B -->
    <invoke service= "CG3j" operation= "calc" ... </invoke> <!-- Termo C -->
    <invoke service= "CG3j" operation= "calc" ... </invoke> <!-- Termo D -->
    <invoke service= "CG3j" operation= "calc" ... </invoke> <!-- Termo E -->
    <invoke service= "CG6j"  operation= "calc" ... </invoke> <!-- Termo F -->
  </flow>
  <!-- Faz multiplicacao dos Termos e Soma no Geral -->
  <assign name= "dAux1" operator= "=" variable= "dResTA" />
   " ..."
</while>
```

Figure 6. Ciclo somatório do *workflow Fotoion*

Na seqüência lógica mostrada acima têm-se: a preparação das variáveis que compõem as matrizes que são argumentos dos termos de *A* a *F*, o cálculo individual de cada termo, a multiplicação dos cálculos e a soma acumulada. Os cálculos fazem parte de um *flow*, ou seja, são executados em paralelo. É importante destacar que o "*Termo A*" não usa um serviço da grade. Os seus cálculos são feitos através de uma seqüência (*sequence*) de atribuições (*assign*).

4.4. Execução do *workflow* exemplo

A execução do *workflow Fotoion* é feita através do *middleware* GPO, cujos componentes são mostrados pela Figura 1, na Seção 1 (Introdução). Para iniciar a execução o usuário aciona o GPO Run com a sintaxe mostrada na Figura 7.

```
$java GPORun \
-submit \
-factory http://143.106.24.147:8080/wsrf/service/GPOFactoryService \
-file Fotoion.gpol
```

Figure 7. Execução do *workflow Fotoion* usando GPO Run

O GPO Run inicia a execução do *workflow* criando uma instância do *GPO Maestro Service Instance*, usando a fábrica *GPO Maestro Factory Service*. A instância é criada no mesmo recurso computacional onde está a fábrica. No exemplo, a fábrica está localizada no recurso identificado pelo endereço *143.106.24.147*. A instância inicia a execução do *workflow* tratando as *definições*, criando as variáveis (*dSomatoria*, *dResTA*, etc) e os serviços a serem usados no processo. Se o serviço for do tipo *Factory*, o GPO Maestro cria uma instância do serviço. Esse procedimento é executado para os serviços *accessFile*, *strFunctions*, *CG3j* e *CG6j*, que são os serviços usados no *workflow Fotoion*.

O *processo* é iniciado com uma seqüência de três *invokes* (abrir o arquivo, ler uma tupla e verificar a leitura, mostrado na Figura 5). Em seguida tem início o ciclo somatório (Figura 6) que começa com uma seqüência de *invokes* para preparar as

matrizes. Com as matrizes preparadas os cálculos são executados em paralelo por um *flow*.

No *workflow Fotoion*, o *flow* contém não só *invokes*, mas também um *sequence*. Nesse caso o GPO Maestro cria uma *thread* para a execução do *sequence* da mesma forma como cria uma *thread* para cada *invoke* que calcula os termos *B* a *F*. Ou seja, qualquer atividade participante do *flow* é executada por uma *thread* exclusiva. Ao encontrar o final do *flow* (</*flow*>), o GPO Maestro fica esperando que todas as *threads* terminem seu processamento, para então continuar a execução do *workflow*. A parte final do ciclo usa elementos *assign* para multiplicar os retornos e somar o valor obtido na variável *dSomatoria*. Essa variável será retornada ao GPORun encerrando o processamento.

5. Conclusão

As grades computacionais são ambientes para computação distribuída de alto desempenho, que facilitam a alocação e escalonamento de recursos distribuídos, atendendo aos requisitos de segurança exigidos em organizações multi-institucionais, fazendo da grade um ambiente altamente colaborativo. No entanto esse ambiente ainda não oferece suporte adequado a *workflows*, principalmente para o usuário.

Este artigo mostra a linguagem GPOL, parte integrante do *middleware* GPO, uma infra-estrutura que faz orquestração de serviços e processos em grades computacionais. A linguagem GPOL é compacta, de fácil aprendizado, e compatível com os padrões atuais da *Web*. Ela tem recursos para execução de tarefas e permite a invocação de serviços da grade. Além disso, pode ser continuamente expandida pelo usuário, na medida que aceita código *Java* como parte do *workflow*. A GPOL aproxima o ambiente das grades dos usuários permitindo a orquestração de serviços tornando esse ambiente mais colaborativo.

6. Referências

[1] I. Foster, C. Kesselman e S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International Journal of High Performance Computing Applications, 15(3):200-222, 2001.

[2] B. Schulze e R. Nandkumar, “Special Issue: Middleware for Grid Computing”, Concurrency and Computation: Practice and Experience 18(6):549-552 (2006).

[3] I. Foster and C. Kesselman and J. Nick and S. Tuecke, “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration”, Globus Project, 2002, www.globus.org/research/papers/ogsa.pdf.

[4] C. R. Senna and E. R. M. Madeira, “A middleware for instrument and service orchestration in computational grids”, Seventh IEEE International Symposium on Cluster Computing and the Grid —CCGrid 2007, Rio de Janeiro, Brasil, Maio, 2007.

[5] Workflow Management Coalition, http://www.wfmc.org.

[6] Triana, “Triana Workflow,” http://triana.co.uk.

[7] S. Zhang, Y. Zong, Z. Ding e J. Liu, “Workflow-oriented grid service composition and scheduling”, International Sysmposium on Information Technology: Coding and Computing, USA, Volume 2, pp. 214-219, IEEE Computer Society, Abril 2005.

[8] Bittencourt, L. F. e Madeira, E. R. M., "A performance oriented adaptive scheduler for dependent tasks on grids". Concurrency and Computation: Practice & Experience, John Wiley & Sons, Online (In press), 2007.

[9] D. Cybok, "A Grid Workflow Infrastructure", GGF10 - The Tenth Global Grid Forum, Berlim, Alemanha, Março 2004.

[10] J. Cao, S. A. Jarvis, S. Saini e G. R. Nudd, "GridFlow: WorkFlow Management for Grid Computing", In Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03), pp. 198-205, 2003.

[11] G. von Laszewski, K. Amin, M. Hategan, N. Zaluzec, S. Hampton e A. Rossi, GridAnt: A Client-Controllable Grid Workflow System, 37th Hawaii International Conference on System Science, Havaí, Jan 5-8, 2004.

[12] D. Bhatia, V. Burzevski, M. Camuseva, G. Fox, W. Furmanski e G. Premchandran, "WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing", Concurrency - Practice and Experience, volume 9, número 6, pp. 555-577, 1997.

[13] D. Thain, T. Tannenbaum e M. Livny, "Condor and the Grid", Grid Computing: Making The Global Infrastructure a Reality, Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors. John Wiley, 2003.

[14] Globus Alliance, "Commodity Grid (CoG) Kits", http://wiki.cogkit.org/index.php/Main_Page.

[15] S. Krishnan, P. Wagstrom e G. von Laszewski, "GSFL: A Workflow Framework for Grid Services", Preprint ANL/MCS-P980-0802, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, Estados Unidos, 2002.

[16] OGF-Open Grid Forum, "Job Submission Description Language (JSDL) Spec., Version 1.0.", http://forge.gridforum.org/projects/jsdl-wg, Novembro 2005.

[17] S. Reynaud e F. Hernandez, "A XML-based Description Language and Execution Environment for Orchestration Grid Jobs", Proceedings of the 2005 IEEE International Conference on Services Computing (SCC'05), Volume 2, pp. 192-199, Orlando, Estados Unidos, Julho 2005.

[18] OASIS, "Web Services Resource Framework (WSRF) v1.2", OASIS Standard, http://www.oasis-open.org/specs/#wsrfv1.2 , Abril 2006.

[19] OASIS, "Web Services Business Process Execution Language Version 2.0 Public Review Draft, http://docs.oasis-open.org/wsbpel/2.0/, Agosto, 2006.

[20] C. R. Senna, "GPO – Um Middleware para Orquestração de Serviços em Grades Computacionais", Dissertação de Mestado, Instituto de Computação, Universidade Estadual de Campinas - UNICAMP, Fevereiro, 2007, http://libdigi.unicamp.br/document/?code=vtls000420528 .

[21] M. G. O. Pinho and L. G. Ferreira, "The Asymmetry Parameter and the Cooper Minimum in the Photoionization of Relativistic Atoms", XXV International Conference on Photonic, Electronic and Atomic Collisions", Konzerthaus, Freiburg, Germany, July 25 - 31, 2007.

Simulação de tomografia computadorizada de raios x utilizando programação paralela em sistemas de processamento de alto desempenho

Mauricio Quelhas Antolin[1], Marcelo Portes de Albuquerque[1], Luís Fernando de Oliveira[2]

[1]Coordenação de atividades técnicas - Centro Brasileiro de Pesquisas Físicas (CBPF)
Rua Dr. Xavier Sigaud, 150 - Urca - Rio de Janeiro - RJ - Brasil - CEP: 22290-180

[2] Departamento de Física Aplicada e Termodinâmica – Universidade do estado do Rio de Janeiro (UERJ)
R. São Francisco Xavier, 524 - Maracanã - Rio de Janeiro - RJ – CEP: 20550-900

antolin@cbpf.br, marcelo@cbpf.br, lfolive@oi.com.br

Abstract. *Computer tomography (CT) is a non destructive technique applied largely in medicine and industry. This work intends to develop a Monte Carlo simulator for CT using X ray diffraction running over a high-performance distributed system and verify its applicability and performance.*

Resumo. *A tomografia computadorizada (TC) é classificada como uma técnica de ensaio não destrutivo com grande aplicabilidade na medicina e na industria. Este trabalho tem por objetivo desenvolver um simulador Monte Carlo para TC usando difração de raio X para ser executado em um sistema distribuído de alta performance e verificar tanto sua aplicabilidade como seu desempenho.*

1. Introdução

A tomografia computadorizada (TC) é uma ferramenta importante no estudo de amostras, pois fornece informações que não estão disponíveis através da inspeção visual direta. Ela é classificada como uma técnica de ensaio não-destrutivo por não comprometer diretamente a integridade da amostra. Em função disto, sua aplicação nas áreas de saúde (médica) e tecnológicas (industrial) tornou-se imprescindível. No entanto, antes de disponibilizar uma nova técnica tomográfica, é necessário avaliá-la quanto a sua aplicabilidade e toda tomografia necessita de uma reconstrução de imagem dos dados adquiridos (e, eventualmente, uma visualização volumétrica no caso da tomografia 3D).

A TC por transmissão de raios x é uma técnica bem estabelecida, com algoritmos de reconstrução bem conhecidos e largamente aplicada, tanto na reconstrução de dados bidimensionais como tridimensionais. Já as TC por espalhamento coerente (difração) de raio x e por fluorescência de raio x precisam ser melhor analisadas.

A aplicação de ambas as técnicas na área médica é promissora, pois elas são modalidades de TC seletivas, isto é, são capazes de reconstruir partes específicas de uma amostra. A TC por espalhamento coerente é capaz de reconstruir numericamente a seção de choque diferencial para o espalhamento da estrutura analisada, e a tomografia por fluorescência reconstrói as concentrações dos elementos químicos selecionados. Para o presente trabalho, somente a TC por difração de raio x será abordada.

Como aplicação da TC por difração de raio x, considere uma amostra biológica composta por tecido sadio e tecido comprometido por algum tipo de anomalia. Se esta anomalia se caracteriza por mudar a estrutura molecular do tecido, a TC por difração de raio x é capaz de reconstruir uma imagem onde somente a anomalia está presente (Barroso *et al*, 2001). Isto é possível uma vez que o espalhamento coerente (modelado pela equação de Bragg) depende da estrutura molecular da amostra. Normalmente, o tecido sadio possui um perfil de espalhamento e o tecido com anomalia, outro. Localizando detectores sensíveis ao raio x em ângulos específicos (associados aos picos de espalhamento do tecido analisado), pode-se registrar as contribuições do tecido sadio ou do tecido com anomalia e, conseqüentemente, reconstruir um ou outro. Isto caracteriza a seletividade da TC por espalhamento coerente.

Para garantirmos a fidelidade dos dados simulados em relação aos dados experimentais, o simulador Monte Carlo para tomografia computadorizada deve incorporar os mecanismos de interação mais relevantes. A quantidade de eventos simulados **deve ser grande o suficiente** para que tenha uma **correspondência estatística com os resultados experimentais**. Além disso, a simulação deve ser capaz de reproduzir espacialmente as trajetórias percorridas pelos fótons gerados e modelar amostras complexas. Logicamente, quanto maior o número de eventos, quanto maior a complexidade da amostra, quanto maior for o número de voxels da amostra (“pixels” tridimensionais), maior é o tempo de simulação.

Estes fatos são os grandes motivadores da implementação da simulação (Zaidi *et al*, 2007) em um sistema de processamento distribuído. Além disso, o estado da arte das técnicas tomográficas aponta para a expansão destas para o campo da tomografia tridimensional. Isto leva a simulação na direção da modelagem de detectores bidimensionais. Por fim, se o número de eventos, a modelagem da amostra, dos mecanismos de interação da radiação e dos detectores pesam no tempo de simulação, mais especializada deverá ser o sistema de simulação e a arquitetura de processamento para suportá-la.

Para ilustrar uma situação a ser simulada, pode-se imaginar um detector bidimensional de 2000x2000 pixels (câmeras CCD convencionais para tomografia). Além disso, a resolução de cada elemento do detector como sendo de 8 bits. Tomando-se a situação do detector registrar todos os eventos gerados pela fonte, isto é, que cada

elemento detector registre 256 eventos, tem-se um total de eventos gerados pela fonte de $1{,}02 \times 10^9$ eventos. Uma simulação completa pede, em média, mais que 360 projeções (imagens geradas). Então, no mínimo, o total de eventos simulados será $3{,}96 \times 10^{11}$. Com a presença da amostra entre a fonte e o detector, a probabilidade de um evento não ser detectado aumenta. Logo, o número de eventos gerados deve ser maior que este valor ilustrativo. Se o tempo médio de vida de um evento estiver em torno de $0{,}6 \times 10^{-3}$ segundos, então o tempo de uma máquina para resolver completamente a simulação tomográfica exigiria um tempo total de $2{,}21 \times 10^8$ segundos (7 anos).

2 Objetivo

A TC por difração de raio x de primeira geração é obtida iluminando-se um corpo de prova com um feixe de raio x, fixando o detector num ângulo de espalhamento característico (ângulo de Bragg) de um material cristalino ou de máximo espalhamento para um material amorfo (por exemplo, tecido biológico) e realizando a translação do corpo de prova em uma trajetória perpendicular ao feixe com um pequeno passo de rotação associado a cada translação a fim de se garantir que toda a amostra seja iluminada pela fonte.

Para TC de segunda e terceira gerações, a fonte de raio x é uma fonte de feixes divergentes e cônico respectivamente. Portando, não há a necessidade de realizar a translação da amostra, uma vez que os feixes divergente e cônico são capazes de iluminar toda a amostra.

Como as trajetórias dos fótons não coincidem quando a amostra gira 180 graus, é necessário que esta sofra uma rotação completa, ou seja, uma rotação de 360 graus para completar o levantamento de dados tomográficos.

Este trabalho pretende mostrar os resultados preliminares de um simulador de TC de terceira geração (feixe cônico com detectores bidimensionais) desenvolvido para gerar dados de difração utilizando processamento distribuído e de alto desempenho. As imagens apresentadas são imagens de transmissão. Além das imagens obtidas pelo simulador, será apresentada uma análise do desempenho do simulador que foi implementado utilizando Charm++ (Parallel Programming Laboratory), uma biblioteca paralela para C++.

3 Materiais e Métodos

O programa de simulação desenvolvido no *cluster* SSolar do Centro Brasileiro de Pesquisas Física (CBPF) está sendo implementado em linguagem Charm++ que é uma evolução natural da ferramenta de processamento distribuído MPI. Este simulador baseou-se no simulador piloto desenvolvido na Universidade do Estado do Rio de janeiro (UERJ) em linguagem C (versão estruturada).

A tomografia, por ser um processo markoviano, pode ser facilmente paralelizada, pois o histórico de vida de cada fóton é completamente independente de qualquer outro, tanto dos que já foram gerados, quanto dos que ainda o serão.

O Charm++ se tornou interessante por já apresentar algumas estruturas (mecanismos) internos de distribuição de tarefas chamadas *chares*. Por exemplo, existe uma estrutura (*chare*) denominada *array*, na qual o nó zero distribui a execução do programa em número de processos determinados pelo usuário. O total de processos não precisa coincidir com o número de nós escravos (ou processadores disponíveis). A estrutura *array* se responsabiliza (de forma transparente ao usuário) de gerenciar a execução dos processos nos nós (processadores).

Os nós escravos recebem os *chares* que ficam esperando a desocupação do(s) processador(es) organizados em fila aguardando a execução. Os *chares* só serão processados a medida que os nós forem terminando os processos atuais. Quando o nó escravo termina um processo, ele manda uma mensagem para o nó zero notificando-o e começa a processar o próximo processo da fila.

A comunicação entre o nó zero e escravos é realizada de forma transparente para o programador (como já foi mencionado), diferentemente da programação em MPI, onde o programador se preocupa em mandar a mensagem de comunicação para os nós.

É importante ressaltar que a estrutura *array* não é muito recomendada para um *cluster* de computadores que não possuem uma rede dedicada, caso que pode gerar erro na execução do programa se o número de processos é grande acarretando no aumento do tempo de espera na fila do processador.

Uma estrutura de pode substituir o *array* é a estrutura de grupos de *chares* (*group*) que também é nativa do Charm++. Esta estrutura é um pouco mais sofisticada que o *array*, pois o conteúdo das mensagens para a comunicação do nó zero com os escravos deve ser feita pelo programador, mas o envio das mensagens permanece transparente ao programador. O *group* traz uma vantagem importante em relação ao *array*, pois o nó zero manda um único processo para cada escravo deixando a fila deste processador vazia. O nó zero só enviará um segundo processo para um determinado nó quando este terminar de executar o processo anterior. A peculiaridade do *group* é a distribuição de um número de processos igual ao número de escravos (processadores) disponíveis.

Este envio é feito de forma independente com relação aos outros nós. Por exemplo, se um programa está rodando em 5 máquinas e somente o nó número 3 terminou o processo, o nó zero mandará um outro processo somente para o nó 3. Está característica é mais adequada para *clusters* que não possuem uma rede dedicada. Ou seja, são computadores que estão ligados em rede, sem o propósito de formar um *cluster* propriamente dito.

Essas são apenas duas estruturas fornecidas pelo Charm++, por isso foi realizado a mudança da linguagem de programação do simulador, que antes era desenvolvido basicamente em C. A linguagem Charm++ nos oferece uma maior facilidade de programação para programas que possam ser paralelizados, pois não é necessário se preocupar como será feita a comunicação entre o nó zero e os escravos. A programação

em Charm++ é basicamente a mesma que a programação em C++, incluindo a utilização das bibliotecas existentes em C++.

Dados de entrada:

Os parâmetros do simulador são inseridos neles através de leitura de arquivos texto. Esses parâmetros são: números de raios-soma, número de projeções, número de eventos (fótons emitidos), posição da fonte, posição da amostra, posição dos detectores e o tamanho e a forma geométrica da amostra.

Os dados das seções de choque dos elementos que compõe a amostra são introduzidos no simulador através da leitura de dois arquivos distintos, um gerado pelo programa Xcom (NIST - National Institute of Standards and Thecnology) e o outro gerado pelo programa XmuDat (IAEA - International Atomic Energy Agency). Estes arquivos estão disponíveis para download nos respectivos sites. Na Tabela 1 mostra os arquivos de entrada que são utilizados no simulador.

Tabela 1: Relação de arquivos de entrada e seus conteúdos.

Arquivo	Conteúdo
Sistema.txt	Número de projeções, raios-soma, eventos e passo de translação.
Amostra.txt	Dimensão espacial, dimensão matricial e tabela de dados
Fonte.txt	Dimensões da fonte, tamanho do foco, distância da fonte ao centro de rotação do sistema, direção de emissão e espectro de emissão.
Detector.txt	Dimensão espacial, dimensão matricial dos detectores
Dados.txt	Arquivo que contém a forma geométrica da amostra, índice dos elementos da amostra com a densidade de cada material relacionando os elementos da amostra com a densidade de cada material
Componentes.txt	Arquivo que determina a relação entre o índice dos elementos da amostra com os dados dos arquivos gerados pelo Xcom e pelo XmuDat.

Dados de saída

Durante o processo de simulação, as contagens registradas pelos detectores são armazenadas na memória e descarregadas em arquivo ao final do processo.

Os detectores estão organizados em forma matricial, assim, quando um “raio-soma”, isto é, uma trajetória específica de fótons desde a fonte emissora até uma posição específica no detetor, atinge um determinado detector, somente a contagem deste

detector é atualizada e os demais detectores continuam com os mesmos dados, sendo que cada passo angular do sistema fonte-detector gera uma imagem (projeção).

4- Processo de simulação

A simulação tenta reproduzir a seqüência de eventos que ocorrem durante o processo de tomografia (Miceli *et al*, 2007, Malusek *et al*, 2008). O primeiro passo da simulação é a leitura da configuração do s dados do simulador (passo 1 da Figura 1). Após este passo o simulador verifica em número de nós a ser utilizado e distribui os *chares* para os nós no *cluster* (passo 2 da Figura 1). É neste ponto em que realmente começa o processamento da simulação. Cada nó irá emitir um fóton que é gerado a partir do espectro de energia fornecido no início da simulação. Este fóton tem, inicialmente, a mesma posição e direção da fonte. Como estamos simulando uma fonte de feixes divergentes, o fóton, logo antes de ser emitido, sofre uma rotação aleatória em torno da abertura especificada da fonte.

Logo após a emissão do fóton, computa-se sua coordenada de impacto com a amostra. Se o fóton não colide com a amostra, ele pode incidir diretamente sobre a matriz de detectores ou não atingir a matriz.

Se o fóton colide com a amostra, verifica-se o índice do elemento da amostra que sofreu o impacto. Se o índice do elemento corresponde ao vazio, o fóton simplesmente é propagado (transmitido) para uma nova posição seguindo sua direção atual. Este passo permanece até que o fóton saia da amostra ou encontre um elemento com índice diferente de vazio. Caso o fóton saia da amostra, verifica-se a direção dele e se ele irá atingir um detector. Se sim, o detector conta o fóton, senão o fóton é perdido e inicia-se uma nova fase de simulação com a geração e emissão de outro fóton.

Se o fóton encontra um elemento da amostra diferente de vazio, é necessário determinar o tipo de material, recuperar as seções de choque e o perfil de espalhamento. A partir destes dados, é possível reproduzir-se os mecanismos de interação (passo 3 da Figura 1).

Após o processamento de todos os fótons o simulador recolhe os dados de todos o nós e armazena em disco as informações dos detectores.

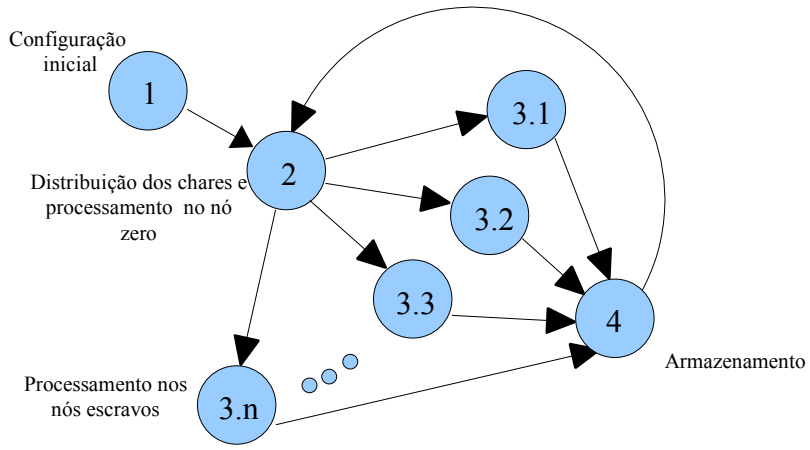


Figura 1: Maquina de estado do simulador.

5- Resultados preliminares:

Já existe uma primeira versão do simulador de TC rodando no *cluster* SSolar do CBPF, esta primeira versão foi implementada com três mecanismos de interação da radiação com a matéria. São eles o efeito Compton, o efeito Rayleigh e efeito fotoelétrico.

A Figura 2 ilustra a saída do detector de transmissão e a rotação do sistema fonte-detectores do simulador, pois cada projeção está a 0º, 45º, 90º e 135º da posição inicial da amostra. Esta figura mostra algumas projeções de uma simulação de uma tomografia com uma amostra simulada composta de gordura (esferas) e tecido com algum tipo de anomalia (cilindro central). Esta simulação foi realizada com 10^7 eventos por raio-soma e um raio-soma por projeção. Cada projeção (figura) tem 128 por 128 *pixels* de tamanho.

Com estes parâmetros, a simulação completa da tomografia (360 projeções) com um total de oito passos angulares teve duração de cerca de 9 minutos sendo simulada em seis nós.

O tamanho reduzido das imagens é devido exclusivamente ao espaço de armazenamento disponível no *cluster* Ssolar. O simulador é capaz de gerar imagens com um maior número de *pixels*, porém isto acarreta um maior disponibilidade de espaço em disco no *cluster*.

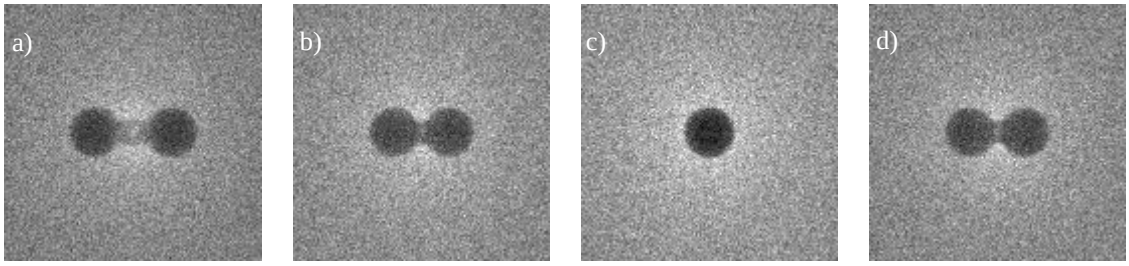


Figura 2: Imagens simuladas com 10^7 eventos, ilustrando a saída do detector de transmissão do simulador. a) 0°, b) 45°, c) 90° e d) 135°.

As figuras 3, 4 e 5 mostram a linha de tempo de simulações feitas com 10^3, 10^5 e 10^6 eventos. Estes gráficos são gerados diretamente pelo Charm++ sem a necessidade de se adicionar linhas no código e podem ser visualizados através da ferramenta nativa do Charm++, baseada em java, chamada projections. (Albuquerque, *M.P. Et* al, 2006). Nestas figuras podemos ver o tempo de duração em milessegundos (eixos das abcissas) de cada estágio do simulador individualmente para cada nó.

Na Figura 3 podemos ver que o tempo de processamento (333ms) é muito inferior ao tempo de arquivamento (1028ms), mostrando que não é interessante realizar a simulação com um pequeno número de eventos por raio-soma. Além disso, um baixo número de eventos gera uma imagem de baixa estatística em que não é possível fazer a distinção dos elementos da amostra. Também foram realizados teste com 10^4 eventos por raio-soma e neste caso comportamento do simulador foi parecido com o teste realizado com 10^3 eventos.

As figuras das linhas de tempo do simulador mostram que o processo de simulação só se tornam interessantes quando o tempo de processamento (951ms) se torna compatível com o tempo de armazenamento (1042ms). Isto ocorre quando realizamos a simulação a partir de 10^5 eventos por raio-soma (Figura 4). Mesmo assim, para este número de eventos podemos verificar que os nós escravos ficam muito tempo com o processador ocioso e continua não sendo interessante realizar a simulação com este número de eventos.

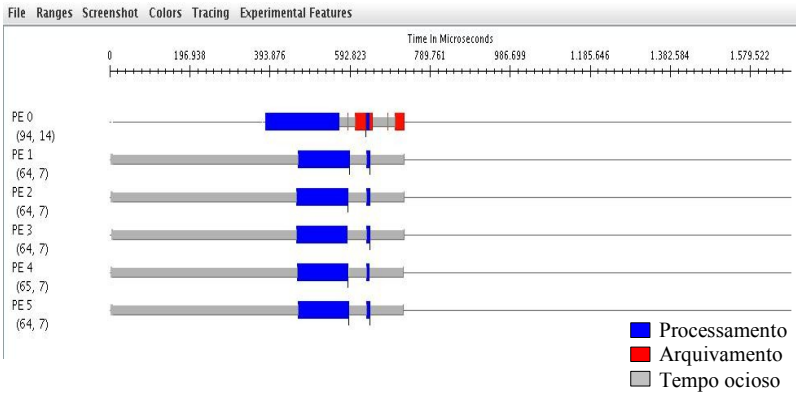


Figura 3: Linha de tempo de uma simulação para um raio-soma e 10^3 eventos por raio-soma

Outra informação que podemos retirar da linha de tempo é que o tempo de troca de mensagens não é crítico para o desempenho do simulador, já que o nó zero manda uma mensagem para cada nó e cada nó e cada nó, ao final do processamento de uma projeção, retorna uma mensagem para o nó zero. Após este ciclo o nó zero manda mais uma mensagem para os nós escravos para eles realizarem mais uma projeção (backgruond), retornando mais uma mensagem para o nó zero.

Para a simulação realizada neste trabalho foram enviadas 24 trocas de mensagem. Lembrando que a simulação aqui a presentada foi realizada apenas com uma projeção por raio-soma.

Os números na Figura 4 representam os passos descritos na maquina de estado do simulador (Figura 1).

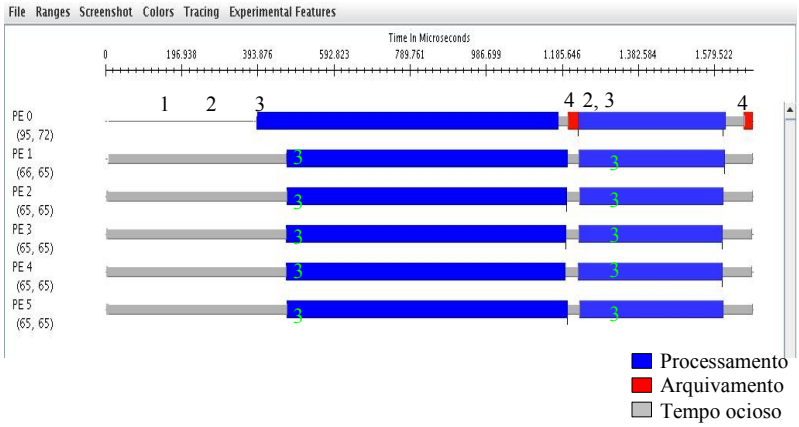


Figura 4: Linha de tempo de uma simulação para um raio-soma e 10^5 eventos por raio-soma. Os números correspondem ao estado da Figura 1

Quando realizamos a simulação com um total de 10^6 eventos o tempo de armazenamento (27 ms) se torna inda menor em comparação ao tempo de processamento das interações (6268 ms) tronando o uso de um sistema de processamento distribuído ainda mais justificável (Figura 5).

Figura 5: Linha de tempo de uma simulação para uma projeção com 10^6 eventos.

O gráfico da Figura 6 mostra o *speed up* do simulador para um máximo de seis nós para uma simulação feita com 10^5, 10^6 e10^7 eventos. Podemos ver que o ganho de performance para 10^5 eventos tem uma estabilidade a partir de 3 nós, isto ocorre devido ao tempo que os nós escravos ficam ociosos aguardando o armazenamento dos dados no nó zero.

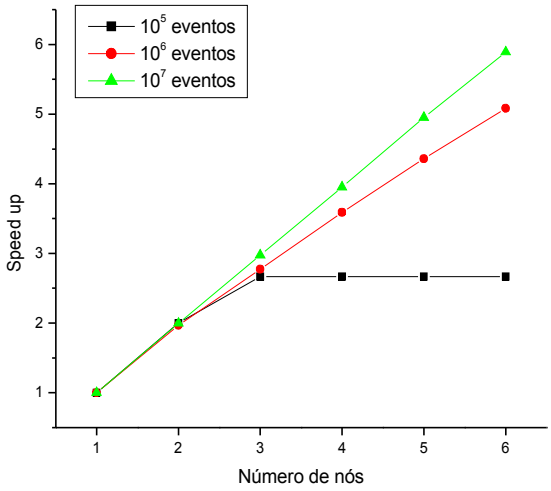


Figura 6: Speed up do simulador para um máximo de seis nós.

6 – Conclusões

Como esperado o simulador é capaz de identificar os elementos de que compõe a amostra gerando imagens de transmissão para realizarmos uma reconstrução da amostra.

Para termos um melhor resultado devemos simular a tomografia com pequenos passo angulares e com altos número de eventos por raio-soma. Isto irá melhorar signifi-

cativamente a qualidade da imagem e a reconstrução das imagens. Inevitavelmente, para esta versão do simulador, devemos sempre utilizar um grande número de passo angulares, devido ao uso do feixe divergente.

Os teste realizados mostram que o simulador tem um desempenho satisfatório e que os tempos de simulação por projeção não são muito grandes, o que nos da a possibilidade de aumentarmos o número de projeções para termos uma melhor qualidade dos resultados. Um fator que costuma ser importante na computação distribuída é o tempo de comunicação entre o nó zero e os nós escravos. Como cada um dos nós tem uma "cópia" de todos os dados do simulador, não temos problema com o tempo de comunicação já que o simulador faz basicamente quatro envios de mensagem por projeção (entre um nó e nó zero).

Ainda deve ser realizado um estudo mais detalhado sobre o *speed up*, ou seja submeter esta versão do simulador em um número de processadores muito maior para descobrirmos qual é o número ótimo de processadores para este programa e também podermos aumentar tanto o número de projeções quanto o de raio-soma.

Como a simulação Monte Carlo permite uma paralelização de auto grau, o uso de um sistema distribuido para o processo tomográfico é justificavel para uma simulaçao que reproduz os passos observados em laboratório, isto é, o simulador realiza exatamente todos os passo que o processo experimental.

Esta primeira versão mostra a potencialidade do simulador, que é capaz de identificar diferentes componentes da amostra e gerar imagens para podermos visualizar e realizar comparações com tomografias de amostras reais para validarmos o resultado do simulador. O projeto do simulador de TC prosseguirá com a implementação dos mecanismos de difração, novas análises de desempenho e métodos para a otimização do código.

7 – Referências

Parallel Programming Laboratory, http://charm.cs.uiuc.edu/, Março, 2008

Xcom - National Institute of Standards and Thecnology, http://physics.nist.gov/PhysRefData/Xcom/Text/XCOM.html, Março, 2008

Xmudat - International Atomic Energy Agency, http://www-nds.iaea.org/reports/nds-195.htm, Março, 2008

Albuquerque M. P., Albuquerque M. P, Alves N., Ribeiro D.P., Moyano L.G., Tsallis C., Baldovin F., e Giupponi G., Dinâmica molecular em ambiente de grade computacional, preprint (2005).

Albuquerque M. P, Almeida A.M., Lessa L.H.P., Albuquerque M. P, Alves N., Moyano L.G. e Tsallis C., NextComp - Molecular dynamics application for long-range

interacting systems on a computational grid environment, communication at IV Workshop on Computational Grids and Applications (Curitiba, June 2006).

Barroso R. C., et al., X-ray synchrotron diffraction tomography for using in biomedical applications, in: IFMBE Proceedings Medicon 2001, IX Mediterranean Conference on Medical and Biological Engineering and Computing, v.1, pp.508-511, Croácia, 2001.

Zaidi H., Ay, M. R., Current status and new horizons in Monte Carlo simulation of X-ray CT scanners, Med Bio Eng Comput 45:809–817 (2007)

Miceli A., Thierry, R., Flisch A., Sennhauser U. Cassali F., Simon M., Monte Carlo simulations of a high-resolution X-ray CT system for industrial applications (2007)

Malusek A., Sandborg M., Carlsson G. A., CTmod—A toolkit for Monte Carlo simulation of projections including scatter in computed tomography, (2008)

Métodos de Escalonanento de Tarefas para Otimização por Simulação em Grade Computacional

Patricia A.P. Costa, Franklin J. Lima, Eduardo L.M. Garcia, Hélio J.C. Barbosa, Bruno R. Schulze

Laboratório Nacional de Computação Científica – (LNCC)
Av. Getúlio Vargas, 333 – Quitandinha – Petrópolis – RJ – Brazil

{pcosta,joffly,bidu,hcbm,schulze}@lncc.br

Abstract. *This paper presents a perfomance analysis of self-scheduling schemes implemented in a distributed system that gives support to a simulation-optmization methodology used for contaminated aquifer remediation numerical modeling. The system uses a master/slave model, where the master performs an evolutionary optimization algorithm and assigns the simulations to the slaves (agents) distributed throughout the grid nodes, which start the simulator execution and calculate the candidate solutions objective functions. A comparative study of self-scheduling schemes was carried out and the results obtained show the influence of task partioning on the system performance.*

Resumo. *Neste trabalho faz-se uma análise de desempenho de métodos de auto-escalonamento para um sistema distribuído que dá suporte a uma metodologia de otimização por simulações para modelagem numérica de remedição de aquíferos contaminados. O sistema usa um modelo mestre/escravo, onde o mestre executa um algoritmo de otimização evolucionista e distribui as simulações entre os escravos (agentes) distribuídos pelos nós da grade, que acionam as execuções do simulador e calculam as funções objetivos das soluções candidatas. Um estudo comparativo entre métodos de auto-escalonamento mostra a influência do particionamento das tarefas no desempenho do sistema.*

1. Introdução

Grades computacionais tem o potencial de se tornarem plataformas poderosas para a simulação de problemas que requerem alto esforço computacional [Berman et al. 2003], em particular, aqueles que usam otimização por simulação, dada a necessidade de pouca comunicação entre as máquinas [Yang et al. 2007] e [Lim et al. 2007]. Esta metodologia tem sido explorada para remediação de águas subterrâneas contaminadas, um problema ambiental relevante. A modelagem computacional deste fenômeno físico pode auxiliar de forma decisiva a busca de soluções para este problema. Neste trabalho, modela-se a ocorrência de contaminação de um aquífero por derramamento de substância tóxica e analisa-se a solução de descontaminação baseada na retirada do contaminante por bombeamento feito por poços de extração [Cunha 2002].

O projeto para a solução deste problema envolve a escolha da quantidade de poços, suas localizações e suas vazões, visando a máxima retirada do contaminante com custo mínimo. A escolha da solução ótima pode ser feita utilizando uma combinação

da experiência de engenheiros com processos de tentativa e erro ou uma combinação de alguma estratégia de otimização e simulação computacional [Mayer et al. 2002] e [Ólafsson and Kim 2002]. No primeiro método, critica-se a falta da formalização matemática necessária para garantir uma solução ótima. A metodologia de otimização por simulação é proposta para automatizar o método de tentativa e erro. O algoritmo de otimização toma decisões quanto à escolha, baseado em avaliações de soluções candidatas que são feitas através de simulações de modelos matemáticos. Entretanto, uma das principais dificuldades é a necessidade de um grande número de simulações até que um resultado satisfatório seja obtido. As simulações são tarefas dispendiosas computacionalmente, porém podem ser realizadas independentemente, o que propicia o uso de grade [Alba and Tomassini 2002] e [Sayeed et al. 2007]. Neste cenário, apresentamos um sistema computacional que implementa o algoritmo de otimização e gerencia a distribuição e a execução das simulações para o modelo de aquífero contaminado entre os nós da grade.

Uma grade computacional, em geral, se apresenta como um ambiente heterogêneo, não somente devido às diferentes características de cada máquina (CPU, clock, RAM, SO, entre outros), mas também devido à existência de taxas de ocupação diversas. Assim, a distribuição de tarefas de maneira dinâmica passa a ter um papel fundamental para que se atinja o balanceamento de carga e obtenha-se desempenho. Neste trabalho serão comparados resultados de experimentos que utilizam diferentes métodos de escalonamento estático e dinâmico [Cheng et al. 2004] e [Hummel et al. 1992].

Na seção 2, é apresentada a arquitetura de implementação do sistema distribuído. Na seção 3 descreve-se a técnica de otimização e na seção 4 é apresentado o modelo de contaminação. Os resultados são mostrados e analisados na seção 5 e finalmente, na seção 6, apresenta-se as conclusões.

2. Arquitetura de Implementação

O sistema considerado é constituído por 3 módulos principais:

- mestre: escrito na linguagem JAVA, executa o algoritmo de otimização e é responsável pela partição e distribuição das simulações
- agente: também escrito em JAVA, o agente é replicado nos nós computacionais da grade e é responsável pela chamada ao simulador e cálculo da aptidão (função objetivo)
- simulador: programa em Fortran que resolve o modelo matemático do aquífero contaminado.

Estes três programas são portáveis. Essa portabilidade é obtida pelo uso de JAVA e pela existência de uma versão do simulador para cada ambiente. Java foi utilizado pela sua portabilidade e o programa em Fortran já existia [Loula et al. 1999].

2.1. Modelo mestre/escravo

Na busca da solução ótima, o algoritmo de otimização precisa avaliar, via simulação, um grande número de soluções candidatas. Neste sistema, essas simulações são feitas concorrentemente, utilizando diversas máquinas da grade. O modelo adotado é o Mestre/Escravo, onde o escravo é implementado pelo módulo agente. Essa denominação se aplica devido à sua autonomia: o agente é acionado de forma independente do mestre e após o término deste, permanece em execução, podendo ser utilizado

em outras simulações; o agente porém não deve ser entendido como módulo inteligente. O agente é replicado nos nós computacionais disponíveis e mestre e agente comunicam-se através de método de invocação remota (RMI), um recurso da linguagem Java que permite a invocação de um método em um objeto remoto.

O mestre implementa os passos do método de otimização, apresentados no Algoritmo 1. No momento da avaliação das soluções candidatas, a parte com custo computacional alto, o mestre particiona a tarefa (simulações) em blocos de subtarefas, de acordo com um algoritmo de escalonamento, e distribui os blocos entre os agentes (1). No início do processo de distribuição, o mestre atribui blocos aos agentes de acordo com uma lista sequencial que contém o nome dos agentes disponíveis. A partir daí, quando um agente termina de avaliar o bloco recebido, retorna ao mestre os resultados obtidos (2) e solicita um novo bloco, colocando-se numa fila de agentes prontos para receber novos blocos (3). Este modelo está representado na Figura 1. O mestre usa os valores obtidos pela simulação para classificar a solução candidata e dar seguimento ao processo de otimização até que alguma condição de parada seja alcançada.

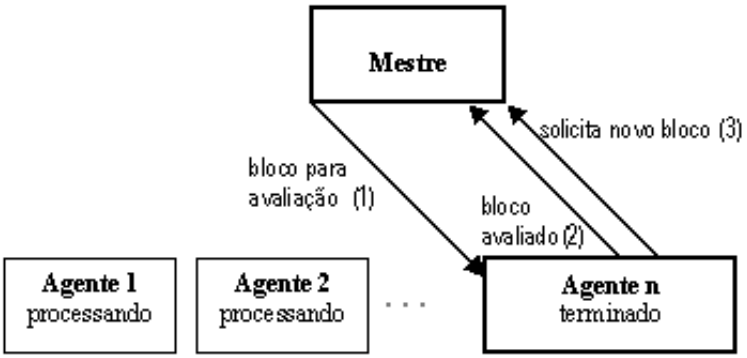


Figura 1. Modelo mestre/agente

2.2. Balanceamento de Carga

Para aplicações que são naturalmente paralelas, a utilização de múltiplos processadores irá certamente diminuir o tempo de resposta. Porém, para obtenção de alto desempenho em ambientes heterogêneos, como em grades, um dos fatores críticos é a distribuição da carga computacional, que pode causar desiquílibrio no sistema, fazendo com que processadores fiquem ociosos em regiões que necessitam sincronismo. O escalonamento dinâmico de tarefas pode ser uma alternativa para solucionar este problema.

2.3. Escalonamento das Tarefas

O escalonamento envolve duas etapas: a partição da tarefa em blocos de subtarefas e a distribuição desses blocos aos processadores disponíveis. Isto pode ser feito de forma estática ou dinâmica. No escalonamento estático, a tarefa é dividida igualmente pelos processadores, e no início da execução do laço de repetição (passos 2 e 7 do Algoritmo 1), o número de subtarefas que cada processador executará já está definido. Este tipo de escalonamento é eficiente sob duas condições: (i) as subtarefas tem todas o mesmo custo e (ii) as máquinas utilizadas são homogêneas e dedicadas.

No sistema apresentado neste trabalho, a primeira condição é satisfeita, mas a segunda não é, sendo mais conveniente usar o escalonamento dinâmico, onde a tarefa é particionada em blocos de subtarefas que são distribuídos durante a execução do laço de repetição. Neste tipo de escalonamento, cria-se uma lista de M blocos de subtarefas, com M maior que o número de processadores. No início da distribuição, cada processador recebe um bloco e, ao final do processamento deste bloco, outro bloco da lista é enviado até que todos os blocos tenham sido processados. A ordem em que os blocos serão distribuídos entre os processadores e a quantidade de blocos processada por cada um, varia em função da sua capacidade de processamento.

Na arquitetura mestre/agente implementada, o mestre tem a função de particionar a tarefa e distribuir os blocos. O mestre aguarda que os agentes terminem o processamento e coloquem-se disponíveis para receber novos blocos. Os primeiros a tornarem-se disponíveis serão os primeiros atendidos. Devido a este tipo de participação dos agentes, este mecanismo é chamado auto-escalonamento. Mecanismos de auto-escalonamento podem diferenciar-se quanto à forma com a qual o mestre calcula o tamanho dos blocos de subtarefas [Hummel et al. 1992], [Chronopoulos et al. 2002] e [Yang et al. 2007]. Neste artigo, são apresentados e comparados resultados de experimentos que utilizam o método de escalonamento estático e os métodos de auto-escalonamento dinâmicos, descritos a seguir, para a execução de uma tarefa de tamanho N em P processadores.

- **PSS** (pure self-scheduling) - neste método, os blocos de tarefas são todos de tamanho 1, gerando N blocos. De implementação simples, o PSS garante o balanceamento de carga, porém pode gerar alto custo de comunicação. É eficiente em ambientes onde o custo de comunicação é baixo ou muito menor que o tempo de execução de 1 tarefa.
- **GSS** (guided self-scheduling) - produz blocos de tamanho decrescente. Os tamanhos dos blocos são determinados pela divisão do número restante de subtarefas pelo número de processadores disponíveis, gerando assim blocos maiores no começo - com intuito de diminuir o número de comunicações - e blocos menores no final - visando garantir o balanceamento de carga.
- **FSS** (factorial self-scheduling) - assim como o GSS, também produz blocos de tamanho decrescente. Porém, no FSS a divisão da carga é feita em etapas. Parte das subtarefas restantes, geralmente a metade, é dividida igualmente pelo número de processadores, gerando, a cada passo, P blocos de tamanhos iguais. De maneira semelhante ao GSS, o FSS gera blocos maiores no começo e blocos menores no final.

A Tabela 1 mostra os partionamentos produzidos pelos diferentes métodos para um conjunto de 200 subtarefas, a serem distribuídas por 8 processadores.

Os fatores que influenciam o desempenho estão relacionados com a razão entre a computação e a comunicação e são diretamente afetados pelo número e tamanho de blocos de subtarefas. O balanceamento de carga requer particionamento de blocos com tamanhos pequenos, enquanto a restrição de pouca comunicação exige poucos blocos [Hummel et al. 1992]. O alto desempenho atingido atendendo a estas duas diretrizes e os resultados dos experimentos comprovam esta afirmativa.

Os métodos apresentados nesta seção são centralizados, ou seja, o trabalho de escalonamento é feito pelo mestre. [Chronopoulos et al. 2002] e [Penmatsa et al. 2007]

Método de partição	Tamanho dos Blocos	Total de blocos
ESTÁTICO	25,25,25,25,25,25,25,25	8
PSS	1,1,1,1,1,1,1,1...,1,1,1,1	200
GSS	25,21,19,16,14,13,11,10,8,7,7,6,5,4,4,3,3,3,2,2,2, $\underbrace{1,1,..,1}_{15}$	36
FSS	$\underbrace{12,..,12}_{8}$, $\underbrace{6,..,6}_{8}$, $\underbrace{3,..,3}_{8}$, $\underbrace{2,..,2}_{8}$, $\underbrace{1,..,1}_{16}$	48

Tabela 1. Exemplos de partições de 200 subtarefas por 8 processadores

descrevem métodos de auto-escalonamento descentralizados e com organização hierárquica. Outras metodologias de escalonamento levam em consideração diferentes fatores como capacidade da CPU [Cheng et al. 2004], estimativas de desempenho [Yang et al. 2007], tamanho da fila de execução [Penmatsa et al. 2007], entre outros.

3. Otimização

3.1. Soluções Candidatas

Para remediar o aquífero contaminado é preciso encontrar uma solução ótima, dentre as várias possíveis. Uma solução candidata consiste em: número de poços de extração que serão instalados, suas localizações e vazões.

Devido ao alto número de soluções candidatas, uma busca linear torna-se inviável. Na prática, um engenheiro, por conhecimento do problema, já descartaria grande parte destas possibilidades, mas mesmo assim, pode-se continuar com um número elevado de soluções a serem analisadas.

Na modelagem computacional destes problemas, outras restrições devem ser consideradas, relativas aos contornos do domínio e locais de colocação dos poços (nós e elementos). Tais restrições podem tornar o modelo ainda mais complexo, além de não reduzir o número de soluções viáveis a um nível em que se pudesse testar todas elas.

Um algoritmo de otimização, a ser visto a seguir, pode ser utilizado neste caso para auxiliar a escolha da solução de forma automatizada, descartando muitas possibilidades que conduzem a uma solução de baixo desempenho.

3.2. Algoritmo Genético (AG)

Neste trabalho, utiliza-se algoritmos genéticos que são uma subclasse dos algoritmos evolucionistas, uma família de modelos computacionais inspirados na teoria evolucionista de Darwin, utilizados em problemas de otimização [Eiben and Smith 2003] e [Goldberg 1989]. A idéia desta técnica é: dada uma população de indivíduos, a pressão do ambiente causa seleção natural - sobrevivência dos mais aptos - e isto causa um aumento na aptidão da população ao longo do tempo. Baseado na aptidão, os melhores indivíduos da população são escolhidos para reprodução, gerando novos indivíduos. A nova geração de indivíduos é obtida através das operações genéticas: crossover (ou recombinação) e mutação. O operador recombinação é aplicado em pares de pais, gerando os filhos, enquanto o operador mutação é aplicado nos filhos, alterando-os. Os filhos gerados substituirão na população atual, os indivíduos menos aptos. Este processo é repetido até que

uma condição de parada seja encontrada, como por exemplo, uma solução de qualidade satisfatória é alcançada ou um número determinado de gerações tenha sido gerada. O processo está representado no Algoritmo 1.

```
1: Inicializar a população com indivíduos escolhidos aleatoriamente
2: Avaliar os indivíduos
3: Repita
4:     Selecionar os indivíduos com maior aptidão (pais)
5:     Recombinar pares de pais, gerando filhos
6:     Aplicar mutação nos filhos
7:     Avaliar os novos indivíduos
8:     Substituir os menos aptos pelos novos indivíduos gerados
9: Até encontrar condição de parada
```

Algoritmo 1: Algoritmo genético

A avaliação dos indivíduos - o cálculo da aptidão - é feita através de uma função objetivo a ser maximizada ou minimizada. Como vários problemas de interesse prático, a remediação de aquíferos contaminados é caracterizada por múltiplos objetivos, entre eles a maximização da quantidade de contaminante extraído e a minimização do custo da remediação. Uma das estratégias para a solução de problemas multi-objetivos é combinar, de alguma forma, os objetivos numa só função. Problemas de remediação são geralmente otimizados através da minimização de uma função custo que engloba o custo dos poços, o custo das bombas, o custo relativo ao tempo de descontaminação e penalidades cobradas sobre a quantidade de contaminante não retirada [Finsterle 2004]. A otimização implementada neste trabalho utiliza essa estratégia, e a função objetivo a ser minimizada considera o custo de implantação, o custo operacional dos poços de extração, o custo das bombas, que varia em função da vazão, e uma multa pela retirada incompleta do contaminante.

Funções objetivo podem ser realistas e complexas ou simplificadas e de desenvolvimento mais rápido. [Ren and Minsker 2005] apresentam um estudo comparando as duas abordagens. A função do presente modelo é simples, pois o objetivo deste trabalho é criar uma ferramenta que possibilita a execução paralela do problema de otimização por simulação.

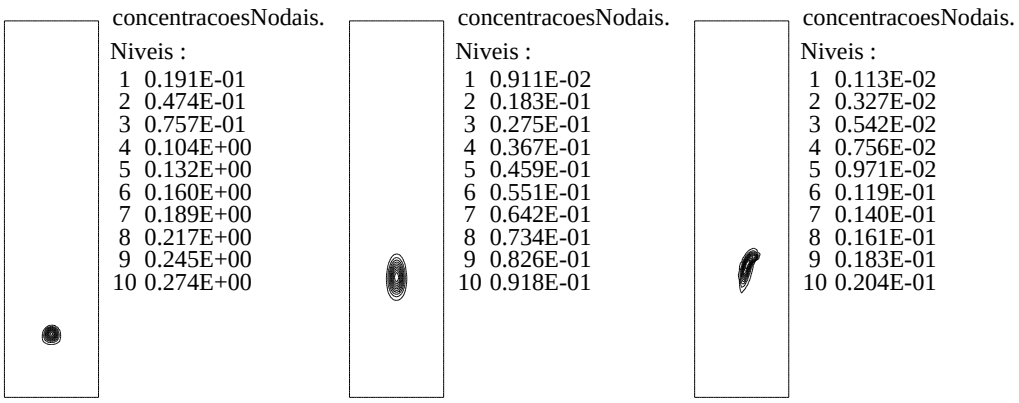
Com a utilização desta técnica, a quantidade de soluções candidatas a ser analisada é bem menor. Ainda assim, milhares de chamadas ao simulador são necessárias para localizar uma solução próxima do ótimo. Porém, os algoritmos genéticos são facilmente paralelizáveis pois a população pode ser particionada e os subgrupos de indivíduos, distribuídos entre os nós computacionais disponíveis, diminuindo assim o tempo de processamento.

4. Simulação

O aquífero contaminado é modelado matematicamente através de um sistema de equações diferenciais parciais que calcula o campo de velocidades e posteriormente o transporte do contaminante, usando um modelo de escoamento miscível [Loula et al. 1999]. O simulador resolve este sistema utilizando o Método de Elementos Finitos e determina a quantidade de contaminante retirada do aquífero pelos poços

de extração. Como dados de entrada, o simulador precisa conhecer os parâmetros que definem a geometria do aquífero, as propriedades físicas do meio poroso e dos fluidos, as condições de contorno e iniciais do modelo matemático, a malha de elementos finitos usada na discretização, o número de poços instalados, a localização e a vazão de cada poço. Os dados de saída consistem em uma lista de concentrações de cada poço em cada passo de tempo, que é utilizada, no cálculo da aptidão, para o computar o volume de contaminante extraído.

Um aquífero saturado, homogêneo e isotrópico foi usado como primeiro experimento. As simulações foram realizadas em uma área total de $2.000,0$ por $8.000,0$ metros, com uma malha de 40×160 elementos quadriláteros bilineares de tamanho $50,0 \times 50,0$ metros. As condições de contorno no lado inferior e superior do domínio são de pressão constante $50,0m$ e $0,0m$ respectivamente, o que produz um escoamento com velocidade constante em todos $8.000,0$ metros do domínio. No lado esquerdo e direito, uma condição de fluxo normal nulo é imposta. A condutividade hidráulica é $100m/dia$ e a porosidade é 0.20. O volume de contaminante é igual a $1000,0$ metros cúbicos. A difusão hidrodinâmica longitudinal é $10,0m$ e a transversal é $1,0m$. O passo de tempo é $2dias$ e cada simulação foi feita para $1200dias$.



(a) condição inicial (b) após 400 dias sem remediação (c) após 400 dias com remediação

Figura 2. Isocurvas

Os resultados apresentados ilustram o problema físico e o resultado da remediação. As primeiras figuras mostram isolinhas de concentração do que seria contaminante. Ao lado das figuras estão valores numéricos que representam as magnitudes das linhas e estão entre 0.0 e 1.0. A figura 2 mostra a condição inicial do modelo - 2(a), a pluma após 400 dias do escoamento sem poços de remediação - 2(b), e a pluma após 400 dias para o caso em que foram incluídos no modelos 2 poços de extração que foram solução do algoritmo de otimização -2(c). Os valores dos níveis das isoconcentrações já demonstram redução na quantidade de contaminante.

A figura 3 mostra o volume de contaminante durante o período da simulação para dois casos: sem remediação e com remediação. Percebe-se a diminuição do volume de contaminante no caso da existência de 2 poços de remediação.

O modelo de aquífero adotado para os experimentos computacionais é bastante simplificado se comparado com uma situação real, porém traz questões relevantes no de-

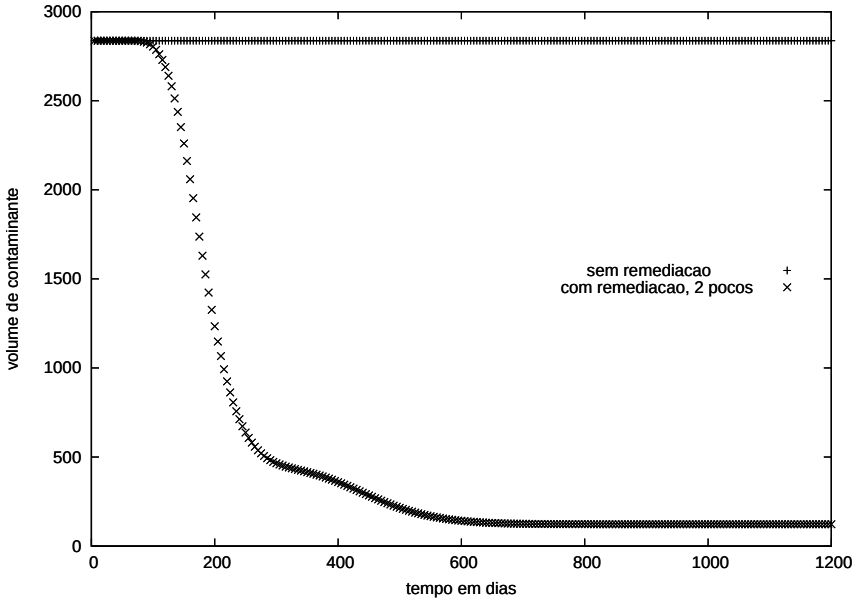


Figura 3. Volume de contaminante no aquífero

senvolvimento da metodologia de otimização por simulação. Devido a esta simplicidade, não é difícil encontrar uma boa solução através de tentativas. A instalação de poços no meio da região do aquífero é uma solução natural para esta situação. Ainda assim este caso pode ser usado para validação do algoritmo genético implementado.

Modelos com meios heterogêneos (permeabilidade e porosidade) e restrições para os locais de instalação dos poços trazem maior dificuldade na escolha da solução e justificam a metodologia adotada.

5. Experimentos

5.1. Ambiente de Desenvolvimento

Para o desenvolvimento dos experimentos vem sendo utilizada a grade computacional VCG (Virtual Community Grid), implementada no LNCC em parceria com outros grupos e apoio da RNP (Rede Nacional de Ensino e Pesquisa). A grade é acessada via portal web, no endereço **http://portalvcg.lncc.br**, e fornece conexões seguras.

A arquitetura da grade é baseada no middleware Globus Toolkit (versões 2.4 e 4.0.5) que inclui serviços e bibliotecas para descoberta, monitoração e gerenciamento de recursos, implementação de segurança e gerenciamento de arquivos. Sobre este middleware foram construídos serviços específicos para atender às necessidades dos usuários do portal. As principais funcionalidades deste portal são: controle de adesão, geração de certificados temporários de máquina e usuários, scripts customizados de instalação, verificação do nível de segurança dos nós, transferência de arquivos, edição e submissão de tarefas, verificação do status das tarefas submetidas, monitoramento da ocupação dos recursos computacionais, entre outras.

A Grade VCG vem disponibilizando até 20 máquinas, entre Core 2, Pentium 4, Pentium D, Xeon Core 2, com memória aproximada de 1Gb por máquina. O sistema operacional utilizado é o Debian Linux Versão 4.0 e a versão de Java instalada nas máquinas

é a 1.5.0.05. Os experimentos apresentados utilizaram 8 máquinas: grade01, grade03 e grade04 (Intel Pentium 4, 3.2 GHz), grade08 e grade18 (Intel Xeon 2 Core, 1.86 GHz) e grade10, grade11 e grade12 (Intel Core 2, 1.8 GHz), escolhidas para forçar uma situação de hetereogeneidade.

As máquinas não estavam rodando em sistema dedicado: 4 delas - grade01, grade03, grade08 e grade10 - estavam rodando outras tarefas.

5.2. Desempenho dos Métodos de Auto-Escalonamento

Cada um dos experimentos executou 1120 simulações, divididas em 120 indivíduos da população inicial e mais 5 gerações de 200 indivíduos candidatos cada, num total de 6 gerações. O algoritmo genético implementado é síncrono, ou seja, a cada geração todos os indivíduos precisam ser avaliados antes que a próxima geração seja iniciada, resultando, para os experimentos feitos aqui, em 6 momentos de sincronismo.

As Tabelas 2 e 3 são usadas para comparar o desempenho dos métodos de escalonamento adotados. A Tabela 2 apresenta, para cada metodologia, o tempo total de execução do sistema de otimização e o número de comunicações entre o mestre (que rodou na máquina grade04) e seus agentes (que rodaram em todas as 8 máquinas). O tempo foi medido no processador mestre, entre os passos 1 e 9 do Algoritmo 1.

Os tempos foram bem diferentes: no pior caso, o tempo foi de 7647s, obtido com a utilização do escalonamento estático, enquanto no melhor caso, obtido com o uso do método PSS, o tempo foi de 3840s, evidenciando a influência da escolha da metodologia de escalonamento no desempenho do sistema.

Os números de comunicações em cada método estão relacionados à quantidade de blocos criados a cada geração de indivíduos. Estes números também são bem diferentes e em certas situações podem influenciar o desempenho - maior quantidade de comunicação pode significar maior tempo de resposta. Os números apresentados resultam da distribuição da população inicial mais a distribuição das 5 gerações. No caso estático: $(120/8+5\times200/8)=48$ e no caso do PSS: $(120+5\times200)=1200$. No GSS e PSS, é necessária uma análise mais detalhada e a Tabela 1 pode ajudar no entendimento.

O FSS e o PSS obtiveram melhor desempenho, com resultados bem próximos, mesmo com o número de comunicações bastante diferente, 48(FSS) e 1120(PSS), o que mostra que neste caso particular, onde foram utilizados apenas máquinas locais, a aplicação não produz gargalo de comunicação. Apesar de neste caso, a comunicação não ter sido relevante, é preciso ficar atento aos seus efeitos, que não podem ser desprezados em aplicações que rodam em grades.

	Tempo Total (hh:mm:ss)	Tempo Total (seg)	nº comunicações
ESTÁTICO	02:07:27	7647	48
PSS	01:04:01	3840	1120
GSS	01:23:12	4992	212
FSS	01:05:12	3912	280

Tabela 2. Tempo total dos métodos

Na Tabela 3 são apresentados os números relativos a cada nó computacional para cada um dos métodos, com uma coluna para o número de indivíduos avaliados por cada processador durante as 6 iterações e outra para o tempo que cada processador levou para avaliar esses indivíduos. Os maiores e menores elementos de cada coluna estão indicados por um sinal ”>”e ”<”respectivamente. Os tempos foram medidos nos passos 2 e 7 do Algoritmo 1.

No escalonamento estático, todos os processadores avaliaram o mesmo número de indivíduos, 15 da população inicial e 25×5 das 5 gerações, num total de 140 avaliações por processador, porém os tempos foram bem heterogêneos. O processador mais lento levou 7645s e o mais rápido, 2450s, o que demonstra desbalanceamento de carga. O GSS tem resultados melhores que a distribuição estática de tarefas, numa tentativa de balanceamento, mas que ainda não produz o desempenho esperado. O tempo máximo de execução de uma simulação, obtido pelo processador mais lento foi de 55s. Como existem 6 momentos onde há necessidade de sincronismo, pode-se afirmar que numa metodologia de escalonamento eficiente, a diferença entre o maior e menor tempo não poderia exceder $6 \times 55s = 330s$. Na última linha da Tabela 3 encontram-se as diferenças entre o maior e o menor tempo de processamento da cada nó para cada um dos métodos. Com a utilização dos métodos PSS e FSS, o número de indivíduos avaliados varia e os tempos de execução são bastante semelhantes, sendo a diferença entre os tempos maior e menor de 92s (PSS) e 232s (FSS). Tal comportamento pode ser atribuído aos blocos de tamanho 1 no final da lista de partição de tarefas (16 blocos). Isto ocorre com o GSS também, porém o balanceamento de carga não é verificado.

	ESTÁTICO		**PSS**		**GSS**		**FSS**	
Nó	nº aval	tempo(s)	nº aval	tempo(s)	nº aval	tempo(s)	nº aval	tempo(s)
grade01	140	6339	< 74	3746	99	4330	85	3779
grade04	140	< 2450	> 211	3696	> 195	3419	> 207	3630
grade08	140	5714	88	3711	118	> 4428	95	3715
grade03	140	5972	86	3668	< 75	3782	84	3724
grade18	140	2656	200	< 3663	186	3412	197	< 3610
grade12	140	2690	193	3700	176	< 3380	188	3611
grade11	140	2680	192	3681	178	3420	189	3629
grade10	140	> 7645	76	> 3755	93	4166	< 75	> 3842
Diferença entre maior e menor tempo	5195s		92s		1048s		232s	

Tabela 3. Tempo (em seg) de cada processador

5.3. Desempenho do modelo de otimização por simulação

A Tabela 4 apresenta o tempo de execução de experimentos feitos para analisar o desempenho do sistema de otimização por simulação. Estes experimentos foram executados em máquinas dedicadas. Na primeira linha temos o tempo de 1120 execuções do simulador em 1 máquina (grade04), sem o sistema de otimização. Esse tempo é semelhante ao da linha 2, obtido para a mesma quantidade de indivíduos com a introdução da otimização, com mestre e 1 agente rodando na mesma máquina (grade04), o que mostra

o baixo custo do sistema de otimização. O tempo obtido com a utilização de 8 processadores mostra ganho: $\frac{T_1}{T_8} = \frac{19453}{2642} = 7,36$. Neste último experimento o mestre rodou na grade04 e os agentes nas 8 máquinas citadas na Seção 5.1; o método de escalonamento foi PSS.

Em cada um desses 3 experimentos, o tempo médio foi obtido através da divisão do tempo total pelo número de simulações. O aumento do tempo médio no experimento feito com 8 processadores é devido não só ao custo de comunicação e escalonamento, mas também à utilização de um conjunto de máquinas heterogêneas, algumas com capacidade menor do que a máquina usada nos experimentos com 1 máquina.

	Máquinas utilizadas	Tempo Total (hh:mm:ss)	Tempo Total (seg)	Tempo Médio de simulação (seg)
1120 simulações	1	05:24:59	19499	17,41
1120 simulações+otimização	1	05:25:43	19543	17,45
1120 simulações+otimização	8	00:44:02	2642	18,87

Tabela 4. Análise de desempenho

6. Conclusão

Este trabalho mostra o desenvolvimento de uma metodologia de otimização por simulação para um problema de remediação de aquíferos contaminados, utilizando uma grade computacional. Para a otimização da solução de descontaminação, foram utilizados algoritmo genético e modelo computacional de aquífero. A otimização por simulação é naturalmente paralelizável, porém para obtenção de alto desempenho em ambientes heterogêneos, estratégias de escalonamento estático e dinâmicos (PSS, GSS e FSS) foram implementadas e experimentadas. O PSS e o FSS obtiveram melhor desempenho, minimizando o tempo de sincronismo entre as iterações, através do balanceamento da carga.

Embora o exemplo modelo numérico de aquífero tratado seja simples do ponto de vista de remediação ele contem os atributos fundamentais para análise desempenho de escalonamento de tarefas em ambiente de grade. Para modelos de aquíferos mais próximos da realidade, é necessário e é possível preparar o algoritmo genético levando em consideração informações específicas deste domínio de aplicação, aumentando assim a sua eficiência.

Para um estudo de escalabilidade é necessário fazer experimentos que utilizem um número bem maior de processadores a fim de poder analisar a influência da comunicação e do algoritmo centralizado de escalonamento, incluindo inclusive máquinas externas.

Referências

Alba, E. and Tomassini, M. (2002). Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462.

Berman, F., Fox, G., and Hey, A. (2003). *Grid Computing: making the global infrastructure a reality*. John Wiley and Sons Ltd.

Cheng, K., Yang, C., Lai, C., and Chang, S. (2004). A parallel loop self-scheduling on grid computing environments. *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks*.

Chronopoulos, A., Penmatsa, S., and Yu, N. (2002). Scalable loop self-scheduling schemes for heterogeneous clusters. *Proceedings of the IEEE International Conference on Cluster Computing*.

Cunha, M. C. (2002). Groudwater cleanup: The optimization perspective (a literature review). *Eng. Opt*, 34:389–702.

Eiben, A. and Smith, J. (2003). *Introduction to Evolutionary Computing*. Springer-Verlag.

Finsterle, S. (2004). Demonstration of optimization techniques for groundwater plume remediation using itough2. Technical report, Earth Science Division, Lawrence Bekerley National Laboratory.

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimization and machine learning*. Addison Wesley Publishing Company.

Hummel, S., E.Schonberg, and Flynn, L. (1992). Factoring, a method for scheduling paralle loops. *Communications os the ACM*, 35(8).

Ólafsson, S. and Kim, J. (2002). Simulation optimization. *Proceedings of the 2002 Winter Simulation Conference*.

Lim, D., Ong, Y., Jin, Y., Sendhoff, B., and Lee, B. (2007). Efficient hierarchical parallel genetic algorithms using grid computing. *Future Generation Computer Systems*, 23:358–370.

Loula, A., Garcia, E., and Coutinho, A. (1999). Miscible displacement simulation by finite elements methods in distributed memory machines. *Computer methods in applied mechanics and engeneering*, 174:339–354.

Mayer, A., Kelley, C., and Miller, C. (2002). Optimal design for problems involving flow and transport phenomena in saturated subsurface systems. *Advances in Water Resources*, 25:1223–1256.

Penmatsa, S., Chronopoulos, A., Karonis, N., and Toonen, B. (2007). Implementation of distributed loop scheduling schemes on the teragrid. *Parallel and Distributed Processing Symposium, IEEE*, pages 1–8.

Ren, X. and Minsker, B. (2005). Which groundwater remediation objective is better, a realistic one or a simple one? *Journal of water resources planning and management*, 131(5):351–361.

Sayeed, M., Mahinthakumar, K., and Karonis, N. (2007). Grid-enabled solution of groundwater inverse problems on the teragrid network. *Simulation*, 83(6):437–448.

Yang, C., Cheng, K., and Shih, W. (2007). On development of an efficient parallel loop self-scheduling for grid computing environments. *Parallel Computing*, 33:467–487.

Uma Arquitetura para Integração de Dados de Impressão Digital através de uma Grade Computacional*

Jurema R. Barretto, Wilson de O. Leite Júnior, Filipe F. Alves, Amadeu A. Barbosa Junior, Fabíola Greve

[1]Departamento de Ciência da Computação – Instituto de Matemática
Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n – 40.170-110 – Salvador – BA – Brazil

{jurema,wilsonjr,lipeta,amadeu,fabiola}@dcc.ufba.br

Resumo. *Este artigo apresenta uma arquitetura para integração de dados de impressão digital dispersos geograficamente com base no uso de uma infra-estrutura de grades computacionais. A arquitetura provê o compartilhamento de informações biométricas através de um ambiente seguro, consistente, independente e de baixo custo, fundamentando-se em soluções de código aberto e hardware heterogêneo. Para avaliar a viabilidade da arquitetura foi desenvolvida uma aplicação para identificação criminal de indivíduos através do reconhecimento de impressão digital. Testes preliminares foram feitos como forma de validar o seu funcionamento e avaliar o seu desempenho.*

Abstract. *This paper presents an architecture for integration of fingerprint records, distributed geographically using grid computing infrastructure. This architecture provides sharing of biometric data through a low cost, secure, consistent and independent environment, based on open-source solutions and heterogeneous COTS hardware. In order to assess the viability of the architecture, an application for criminal identification through fingerprint recognition was developed. Preliminary tests were made as a way to validate the functionality and evaluate performance.*

1. Introdução

A biometria é a ciência que permite que a identidade de uma pessoa seja auferida a partir de características físicas inerentes a ela, tais como, impressões digitais, face, voz, etc. [Costa 2001]. Sistemas automáticos de identificação biométricos vêm sendo cada vez mais utilizados como alternativas aos recursos tradicionais, tais como senhas, códigos de barra e identificação em papel. Uma das suas maiores aplicações está na segurança pública, na identificação civil e criminal de cidadãos. Os Estados brasileiros utilizam essencialmente a impressão digital como recurso para identificação dos indivíduos. Processos automáticos empregados no reconhecimento de duas impressões digitais são simples e eficientes. O grande entrave consiste em efetuar verificações de um para muitos. A depender da ordem de grandeza da base de dados, esse procedimento pode levar muito tempo. Para minimizar esse tempo de busca, costuma-se adotar gerenciadores de banco de dados especializados e supercomputadores, cujo custos são elevados, dificultando assim a implantação de um sistema de reconhecimento em escala nacional.

*Trabalho com apoio da FAPESB, Bahia. Fabíola tem apoio do CNPQ, Brasil.

As grades computacionais, por sua vez, viabilizam a execução de aplicações sobre processadores dispersos geográfica e administrativamente com a promessa de maior capacidade e melhor custo-benefício que as plataformas tradicionais (como supercomputadores). A computação em grade baseia-se no compartilhamento coordenado de recursos para a solução de problemas em uma organização virtual, dinâmica e multi-institucional, onde diversos domínios administrativos coexistem podendo ter suas próprias políticas de utilização [Cirne and Santos 2005]. Ao mesmo tempo, ao usar as grades é possível compartilhar grande quantidade de dados distribuídos (geograficamente), e dispor de meios para implementar recursos de segurança e autenticação afim de manter o sigilo dos dados [Foster et al. 2002]. Dessa maneira, a utilização dessa infra-estrutura apresenta-se como uma boa opção para o processo de identificação de indivíduos, através da impressão digital, numa grande escala.

Este artigo apresenta uma arquitetura para integração de dados de impressão digital dispersos geograficamente com base no uso de uma infra-estrutura de grades computacionais. A arquitetura proposta permite o compartilhamento de informações através de um ambiente seguro, consistente, multi-plataforma, independente e fundamentada em soluções de código aberto e hardware heterogêneo. Para avaliar a viabilidade da arquitetura foi desenvolvida uma aplicação para identificação criminal de indivíduos através do reconhecimento de impressão digital. Testes preliminares foram feitos como forma de validar o seu funcionamento.

O restante do artigo está dividido da seguinte forma: a seção 2 apresenta uma visão geral sobre o processo de reconhecimento automático através da impressão digital. A seção 3 apresenta a arquitetura proposta. A seção 4 descreve a aplicação baseada nessa arquitetura, relatando as tecnologias utilizadas e os principais componentes da aplicação. A seção 5 apresenta os testes preliminares efetuados e a seção 6 conclui o artigo.

2. Reconhecimento Automático de Impressão Digital

A biometria fornece recursos para a medição de características que definem a individualidade das pessoas [Costa 2001]. Essas características podem ser tanto físicas quanto comportamentais. Alguns elementos que fazem parte dos métodos biométricos para reconhecimento de indivíduos são: impressão digital, face, íris, retina, assinatura manuscrita, entre outros. A identificação de pessoas, através da impressão digital, pode ser empregada em diversas áreas como: sistemas de segurança, identificação de criminosos, controle de acesso a locais restritos e comprovação de identidade.

Os seguintes elementos constitutivos de uma impressão digital podem ser destacados [Kazienko 2003]: (i) linhas pretas – também chamadas de cristas papilares, são as linhas impressas da impressão digital; (ii) linhas brancas – também chamadas de vales, são as linhas que separam as linhas impressas de uma impressão digital; (iii) delta – é o triângulo formado pela bifurcação de uma linha simples ou pela brusca divergência de duas linhas paralelas; (iv) minúcias – são pontos característicos que ocorrem nas linhas pretas e que distinguem uma impressão digital de outra, são conhecidos por final de linha ou bifurcação; (v) núcleo – ponto localizado na área central da impressão digital.

O grande crescimento do número de registros de impressões digitais tornou praticamente inviável o processo de identificação manual por um perito, fazendo surgir, assim, a criação de um método automático. Em torno de 1960, o FBI, nos Estados Unidos, e

algumas instituições de segurança pública da Inglaterra e da França iniciaram o desenvolvimento do sistema automático de identificação através das impressões digitais, denominados AFIS que é o acrônimo de *Automated Fingerprint Identification System*. O AFIS envolve diversas etapas, dentre as quais destacam-se: aquisição, classificação, extração de características e verificação [Costa 2001].

A aquisição consiste da captura da impressão digital que pode ser feita pela aquisição com tinta em papel ou através de um sensor específico. A aquisição em papel, denominada método *ink and paper*, consite em friccionar o dedo em uma tinta e em seguida em um papel, normalmente formulário específico. Para armazenamento em base de dados e/ou utilização por sistemas computacionais é necessário a utilização de um digitalizador de papel (*scanner*). Em geral, esse método apresenta borrões ou manchas ocasionadas pelo excesso ou falta de tinta afetando, assim, a qualidade da imagem. É necessário que sejam aplicados algoritmos de segmentação para melhoramento da imagem e remoção de informações desnecessárias como letras, números e borrões. Na utilização de um sensor, a captura é feita através do toque da superfície do dedo no mesmo, que é capaz de digitalizar a impressão digital através do contato. É o método considerado mais eficiente, pois apresenta as imagens com qualidade superior às capturadas pelo método *ink and paper* e nos padrões determinados para os sistemas AFIS.

A classificação consiste em atribuir a uma impressão digital uma classe, pré-definida, criada por Henry [Costa 2001, Maltoni et al. 2005], visualizada na Figura 1, e dividida em: (i) arco plano – as linhas correm de um lado a outro em uma forma convexa; (ii) arco angular – apresenta uma acentuada elevação das linhas do centro; (iii) verticilo – possuem dois deltas um à direita e outro à esquerda do observador e linhas nucleares que assumem formatos variados; (iv) presilha externa – as linhas seguem à direita do observador e tendem a voltar para o mesmo lado, apresentando um delta à esquerda; (v) presilha interna – as linhas seguem à esquerda do observador e tendem a voltar para o mesmo lado, apresentando um delta à direita.

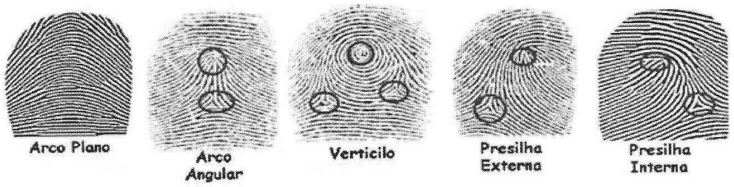


Figura 1. Classificação de uma Impressão Digital

A extração de características consiste em adquirir da imagem peculiaridades que a individualiza. Essas peculiaridades são definidas como minúcias que são pequenas imperfeições das cristas papilares. Esta etapa é crucial para a etapa de verificação (determinação se duas impressões digitais são iguais ou não), pois é através das minúcias que as impressões digitais são comparadas.

Como exemplo de um software que utiliza a tecnologia AFIS, pode-se citar o NFIS. O NFIS é um software de distribuição gratuita (inclusive com fontes), controlada mediante solicitação ao FBI (*Federal Bureau of Investigation*). Foi desenvolvido pelo NIST, Instituto Nacional de Padrões e Tecnologias dos Estados Unidos, em parceria com o FBI. O NFIS é composto por sete pacotes [Watson et al. 2004] com funcionalidades dis-

tintas, mas que podem ser usados em conjunto para prover o processo de reconhecimento de impressão digital. Dentre os pacotes do NFIS, vale destacar o PCASYS, MINDTCT e BOZORTH3, reponsáveis pela classificação, extração das minúcias e comparação, respectivamente.

3. Arquitetura para Integração de Dados de Impressão Digital através de Grades Computacionais

Apresentamos nessa seção uma proposta de arquitetura para a integração de dados de impressão digital num ambiente de grade computacional.

Esta arquitetura é classificada como uma grade de dados (*data grid*) por propor acesso, pesquisa, manipulação e gerência de grandes volumes de dados distribuídos [Santos 2006]. De acordo com a taxonomia apresentada em [Venugopal et al. 2006], esta arquitetura refere-se a um modelo de *data grid* federativo, por envolver instituições compartilhando base de dados já existentes, onde cada instituição é responsável pelo controle dos seus dados. O escopo envolvido é intra-domínio, ou seja, restrito a uma única área - a de Segurança Pública. Além disso, é formada por uma organização virtual colaborativa composta por entidades que juntas compartilham recursos e dados e colaboram num objetivo comum.

O *Globus Toolkit 4* (GT4) [GT4 2007] foi o *middleware* de grade utilizado para prover comunicação e integração de recursos e dados. Ele possui as ferramentas necessárias para prover segurança, gerenciamento de aplicações e movimentação de dados. Além disso, ele segue o padrão OGSA (*Open Grid Services Architecture*) [Berry et al. 2007], que é uma arquitetura de grades fundamentada na tecnologia de *Web Services*, denominada, *grid service* [Foster et al. 2002]. Os componentes do GT4 utilizados foram o GSI (*Grid Sercurity Infrastructure*) e o OGSA-DAI (*Open Grid Services Architecture - Data Access Information*). Em particular, o transporte das mensagens é feito via o protocolo SOAP (*Simple Object Access Protocol*) que converte os dados a serem transmitidos em XML (*eXtensible Markup Language*) [Booth et al. 2004].

O modelo da arquitetura pode ser visualizado na Figura 2. Este modelo envolve três camadas divididas em aplicação, serviços de grades, representado pelo GT4, e Sistema Gerenciador de Banco de Dados (SGBD). A camada de aplicação envolve a aplicação de reconhecimento de impressão digital que é executada na grade, através da utilização dos pacotes PCASYS, MINDTCT e BOZORTH3, do NFIS. O acesso a aplicação da grade pode ser feito através de uma interface para *desktops* ou dispositivos móveis.

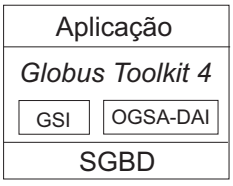


Figura 2. Modelo da Arquitetura

A segurança do sistema é provida pelo GSI no nível de transporte (*transport-level security*) [Sotomayor 2005] onde toda informação trocada entre o cliente e o servidor é

criptografada (inclusive as mensagens SOAP entre a aplicação e o OGSA-DAI). O GSI utiliza uma entidade certificadora *SimpleCA*, que é reponsável pela geração de certificados digitais a usuários e servidores. Cada máquina que faz parte da grade computacional possui um certificado digital, bem como todo usuário precisa ter um certificado próprio (instalado em sua estação) para utilizar os serviços. O mecanismo de autorização utiliza um arquivo de mapeamento de usuários, denominado *grid-mapfile*.

Para prover a distribuição heterogênea das bases de dados utiliza-se o *middleware* OGSA-DAI (*Open Grid Services Architecture - Data Access Information*). Esse disponibiliza um conjunto de ferramentas, implementadas em Java, para permitir acesso e integração de dados para aplicações constituídas por vários domínios administrativos [GT4 2007]. As bases de dados são utilizadas como serviços de grade (*Grid Services*) constituindo uma organização virtual. Os dados que compõem esse ambiente são apresentados aos usuários como uma única base de dados. Através do recurso *ResourceGroup* é possível submeter um único comando de consulta (SQL) e obter os dados de origens diferentes que são agrupados transparentemente. Ainda no OGSA-DAI é possível que o cliente escolha o método de entrega dos dados seja, dentre outros, (i) pelo uso da especificação GridFTP ou (ii) pela resposta direta do agrupamento das consultas (*ResultSet*) dos bancos de dados. A literatura recomenda usar o GridFTP para transferências de grandes volumes de dados.

3.1. Validação da Arquitetura

Para validar a arquitetura apresentada, definimos como estudo de caso uma aplicação para identificação criminal através do reconhecimento da impressão digital.

A identificação criminal no Brasil tem passado por um processo de automatização, onde alguns Estados, como é o caso da Bahia, têm investido na aquisição e implantação de sistemas AFIS para melhor combater a criminalidade. Para que se obtenha informações criminais de todo o Brasil, o Estado deve fazer acesso a base criminal da Polícia Federal onde essas informações estão centralizadas. Levando-se em consideração a quantidade de registros criminais no Estado da Bahia, em torno de 60.000 (dados de [SSP 2005]), e projetando essa quantidade para os demais Estados (sem considerar os Estados com mais ou menos números), nota-se que a base da Policia Federal armazena mais de um milhão de registros, requerendo, assim, uma máquina com capacidade de armazenamento e processamento muito grande.

A arquitetura aqui apresentada, mostra-se como alternativa a uma solução centralizada via supercomputadores. Ela provê autonomia dos dados entre os Estados Brasileiros, além de evitar o investimento em máquinas muito especializadas. Na criação do sistema automático de identificação criminal em uma grade computacional deve-se levar em consideração os seguintes requisitos que a aplicação demanda.

3.1.1. Requisitos Não Funcionais

- Segurança: utilizar mecanismos de segurança que garantam a integridade e privacidade, através da criptografia, dos dados trafegados. Esse requisito é de fundamental importância visto que envolve informações civis e criminais de um indivíduo, consideradas informações sigilosas, e só pessoas autorizadas devem ter

acesso as mesmas.
- Disponibilidade: os dados devem estar sempre disponíveis para acesso.
- Consistência das informações: as informações disponibilizadas devem ser consistentes, independente da arquitetura de software ou hardware utilizada.
- Integridade: deve-se ter certeza de que as informações obtidas são de fontes corretas e confiáveis.
- Concorrência: o ambiente deve permitir múltiplas requisições simultâneas.
- Transparência: a integração das bases de dados deve ser transparente ao usuário final.
- Crescimento em Escala: o sistema deve permitir a inclusão de novas bases de dados e recursos, bem como, usuários, sem comprometer o desempenho do sistema.
- Independência: o sistema deve depender pouco ou não depender de licenças proprietárias de softwares, de maneira que, possa ser reutilizado com facilidade, garantindo independência tecnológica ao usuário.

3.1.2. Requisitos Funcionais

- Manipular Imagem da Impressão Digital: a imagem da impressão digital deve ser capturada através de um sensor específico. Em seguida deve ser enviada à grade para classificação e extração das minúcias, através dos módulos específicos do NFIS.
- Solicitar Busca: de acordo com a classe identificada na impressão capturada e nos filtros (sexo, dedo, etc.), opcionais, definidos pelo usuário, o sistema efetua a busca nas bases distribuídas. Ao identificar registros relacionados à busca efetuada, o sistema deve gerar os arquivos de minúcias e efetuar a comparação através da execução do módulo específico do NFIS.
- Visualizar Suspeitos: apresenta os dados do(s) indivíduo(s) identificado(s) como suspeito(s). Essas informações são os dados civis do indivíduo.
- Visualizar Crimes: lista as informações sobre os crimes cometidos pelo(s) indivíduo(s) identificado(s) como suspeito(s).

Os seguintes passos são executados para que os requisitos funcionais sejam atendidos:

1. O usuário solicita a captura da impressão digital através da interface cliente;
2. A interface cliente, então, envia a imagem capturada para a aplicação que executa na grade. Neste momento, podem ser enviados filtros por sexo e dedo;
3. A aplicação cliente solicita à aplicação da grade, a classificação da impressão digital enviada;
4. A aplicação da grade executa o módulo PCASYS, do NFIS, para classificação da imagem, retornando sucesso ou falha para a classificação. Se a falha estiver relacionada a um tipo incompatível de arquivo, não reconhecido pelo PCASYS, a aplicação cliente é informada para que seja possível enviar um novo arquivo da impressão digital;
5. A aplicação cliente solicita à aplicação da grade, a extração das minúcias da impressão digital enviada;

6. A aplicação da grade executa o módulo MINDTCT, do NFIS, para que a extração das minúcias seja efetuada, retornando sucesso ou falha para esse procedimento. Em caso de falha, o procedimento é re-executado, se a falha persistir uma mensagem de erro é retornada à aplicação informando que não foi possível efetuar a comparação;
7. A aplicação cliente solicita à aplicação da grade, a comparação das minúcias das impressões digitais;
8. A aplicação da grade, então, submete uma consulta (*query*) ao OGSA-DAI para que este efetue o agrupamento das minúcias, distribuídas nas bases que compõe a grade, de acordo com a classe identificada e com os filtros aplicados;
9. Após o agrupamento dos dados, o OGSA-DAI retorna à aplicação da grade as minúcias encontradas;
10. A aplicação da grade efetua a comparação através da execução do módulo BOZORTH3, do NFIS, retornando para a aplicação cliente a lista de suspeitos;
11. Ao visualizar a lista de suspeitos, a aplicação cliente solicita a aplicação da grade a lista de crimes cometidos;
12. A aplicação da grade solicita ao OGSA-DAI o agrupamento dos crimes do suspeito solicitado;
13. O OGSA-DAI retorna a lista de crimes a aplicação da grade e esta a aplicação cliente.

A modelagem de banco de dados para esta aplicação é composta por tabelas que armazenam registros civis do indivíduo, informações sobre os crimes cometidos pelo mesmo, tabelas com as opções de filtro para aplicação e informações relacionadas à impressão digital. A Figura 3 ilustra a modelagem de dados. A tabela **cidadao** armazena os dados civis dos indivíduos e está relacionada com as tabelas **crimes_cidadao** e **imagem**. A tabela **crimes_cidadao** compõe o relacionamento entre a tabela **crimes** e a tabela **cidadao**, registrando assim, os crimes cometidos pelo **cidadao**. A tabela **imagem** armazena as informações sobre a imagem da impressão digital, sendo registrada a imagem da impressão digital e suas minúcias. Essa tabela relaciona-se com as tabelas **classeid** e **dedo**, onde são informados o tipo de dedo e a classe que a impressão digital pertence. A tabela **municipio** está relacionada às tabelas **cidadao** e **crimes_cidadao** para que seja possível armazenar a naturalidade do cidadão e o local onde o crime foi cometido, respectivamente. Os Estados são identificados pela tabela **estado** relacionada com a tabela **municipio**.

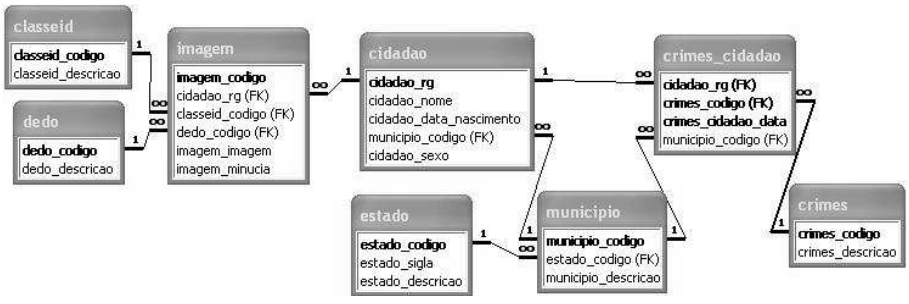


Figura 3. Modelagem do Banco de Dados

4. Desenvolvimento da Aplicação de Reconhecimento Automático de Impressão Digital

Foram desenvolvidos dois serviços de grade: um serviço para criação e gerenciamento dos recursos *Web Services*, denominado AfisDCCFactoryService; e um serviço que provê as funcionalidades do processo de reconhecimento da impressão digital, denominado AfisDCCService. Toda a implementação desses serviços foi desenvolvida na linguagem Java, uma das linguagens suportadas pelo GT4. O AfisDCCFactoryService é uma classe Java que implementa o *Web Service* que é utilizado como interface para integração com o recurso da *Factory* (responsável por gerenciar e instanciar os recursos). O AfisDCCService é uma classe Java que implementa a instância *Web Service* que disponibiliza os principais métodos da aplicação. As operações disponibilizadas pelo *Web Service* às aplicações clientes são:

- *SendFingerResponse sendFinger(SendFinger complexType)*: responsável pelo envio da impressão digital coletada, pelo sensor, através de um vetor de bytes. O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).
- *ClassifierResponse classifier(Classifier complexType)*: permite que a aplicação cliente solicite a classificação da impressão digital enviada pela operação sendFinger(). O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).
- *Minutia_extractorResponse minutia_extractor(Minutia_extractor complexType)*: permite que a aplicação cliente solicite a extração das minúcias da impressão digital enviada pela operação sendFinger(). O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).
- *SendFilterResponse sendFilters(sendFilters complexType)*: permite que a aplicação cliente informe ao recurso algum tipo de filtro a ser aplicado à busca das minúcias. A utilização do filtro é opcional e envolve o tipo do dedo (polegar, indicador, etc.) e o sexo do indivíduo. O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).
- *MatchingResponse matching(Matching complexType)*: responsável por fazer a comparação das minúcias extraídas da impressão digital capturada, com as minúcias coletadas da base de dados, referentes à classe detectada e aplicando os filtros, se definidos. O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).
- *GetSuspectCrimesResponse getSuspectCrimes(GetSuspectCrimes complexType)*: retorna à aplicação cliente as informações relativas aos crimes dos suspeitos identificados.

Além das classes supracitadas, foram criadas as seguintes classes, que servem de suporte à aplicação: (i) *AfisExecution*: classe utilizada para invocar os pacotes PCASYS, MINFTCT e BOZORTH3 do NFIS, responsáveis, respectivamente, pela classificação da impressão digital, extração das minúcias e comparação das mesmas com as agrupadas das bases de dados; (ii) *OgsaDaiClient*: classe Java utilizada para servir de cliente ao serviço OGSA-DAI, possibilitando que a busca às bases de dados sejam feitas de forma transparente; (iii) *SuspectModel, CrimeModel e FilterModel*: classes utilizadas para retornar informações a respeito dos suspeitos, crimes e filtros respectivamente.

Para execução através do *Web Service* intermediário, foram criadas duas classes que tratam com a regra de identificação da aplicação. Essas classes são *clsCrime* e *clsSuspeito*. Nessas, existem métodos para comunicação com o *Web Service* permitindo, assim, a comunicação remota com a aplicação da grade. Este *Web Service* é responsável por executar o cliente que se comunica e consome o *grid service* e por disponibilizar os serviços: (i) EnviarFiltro – para envio da imagem da impressão digital e filtros selecionados; (ii) BuscarSuspeitos – disponibiliza informações sobre os suspeitos; e (iii) BuscarCrimes – disponibiliza informações dos crimes cometidos pelos suspeitos identificados.

Adicionalmente, foi necessário armazenar temporariamente em arquivos texto as minúcias coletadas nas bases de dados para manter a compatibilidade com o processo de comparação de minúcias do NFIS. Uma vez que todos os pacotes PCASYS, MINDTCT e BOZORTH3 estão programados em C e assumem o uso de arquivos texto.

5. Avaliação da Arquitetura

A arquitetura proposta foi implementada e foram realizados alguns testes com o objetivo de avaliar o seu funcionamento e obter medidas preliminares de desempenho.

O ambiente de testes adotado consiste em cinco máquinas que compõem um *cluster* do tipo *beowulf*. As máquinas são AMD Opteron 2GHz, quatro com 2GB de RAM e uma com 4GB. As bases de dados estão distribuídas em quatro máquinas onde duas possuem o SGBD PostgreSQL e duas possuem o SGBD MySQL, além do GT4 e OGSA-DAI. Utilizamos SGBD's diferentes para testar a integração de bases distintas provida pelo OGSA-DAI. A quinta máquina é responsável por gerar os certificados, além de possuir o AfisDCCService, o GT4 e os pacotes do NFIS. Esta máquina foi utilizada, também, para testar a busca dos dados centralizados em uma única máquina. Nesta máquina, o SGBD instalado é o PostgreSQL.

O número total de imagens de impressão digital armazenadas, até o momento, é de 2700. Essas imagens foram disponibilizadas com o pacote NFIS. Para medir o tempo total de execução foram considerados os seguintes tempos: (i) tempo de acesso ao serviço (T_{AS}) – corresponde ao tempo necessário para instanciar o recurso da grade (métodos do *grid service* e criação do arquivo temporário para armazenamento da impressão digital; (ii) tempo de envio da impressão digital (T_{EnID}); (iii) tempo de classificação da impressão digital (T_{ClID}); (iv) tempo de extração das minúcias da impressão digital (T_{ExMin}); (v) tempo de envio dos filtros (T_{EF}); (vi) tempo do processo de comparação das impressões digitais (T_{CoID}); e (vii) tempo para obtenção dos suspeitos (T_{Sus}). O tempo total pode ser representado pela seguinte fórmula:

$$T_{total} = T_{AS} + T_{EnID} + T_{ClID} + T_{ExMin} + T_{EF} + T_{CoID} + T_{Sus} \quad (1)$$

5.1. Cenários de Testes

Foram utilizados quatro cenários de testes, envolvendo busca de dados centralizado em uma única máquina e busca de dados distribuídos entre as máquinas do *cluster*. Os testes foram efetuados através de interfaces *desktop* e interface de um PDA, sendo a submissão individual e simultânea entre o *desktop* e o PDA. Os filtros aplicados foram classe e por

classe e sexo. Os tempos de respostas foram medidos em segundos. Os testes envolveram busca por impressão digital de cada classe existente (arco, presilha externa, presilha interna e verticilo). A aplicação *desktop* foi executada via interconexão *Ethernet* conectado a 1000Mbps e a aplicação do PDA utilizou conexão *wireless*. O tamanho de cada base de dados e a quantidade de registros armazenados, em respectivo em cada máquina é: máquina 1 (15MB, 2700), máquina 2 (3.0MB, 675), máquina 3 (7920KB, 675), máquina 4 (2.9MB, 675), máquina 5 (8488KB, 675). A seguir serão apresentados os cenários utilizados nos testes.

- Cenário 1: os dados estão armazenados em uma única máquina, possuindo um total de 2700 registros. Foram efetuadas buscas pelo PDA e por um *desktop*. Inicialmente foram efetuadas buscas apenas pelo PDA e *desktop* e depois com os dois ao mesmo tempo sendo medidos seus tempos de respostas separadamente (identificados no gráfico por ”PDA simultâneo com *Desktop*”e ”*Desktop* simultâneo com PDA”, respectivamente). Utilizou-se o filtro pela classe da impressão digital.
- Cenário 2: o que difere este cenário do cenário 1 é que neste foi incluído, além da classe, filtro pelo sexo.
- Cenário 3: os dados estão armazenados em quatro máquinas com 675 registros cada. Foram efetuadas buscas pelo PDA e por um *desktop*. Inicialmente foram efetuadas buscas apenas pelo PDA e *desktop* e depois com os dois ao mesmo tempo sendo medidos seus tempos de respostas separadamente (identificados no gráfico por ”PDA simultâneo com *Desktop*”e ”*Desktop* simultâneo com PDA”, respectivamente). O filtro utilizado foi apenas pela classe da impressão digital.
- Cenário 4: o que difere este cenário do cenário 3 é que neste foi incluído, além da classe, filtro pelo sexo.

Os resultados dos testes podem ser apreciados nas Figuras 4 e 5.

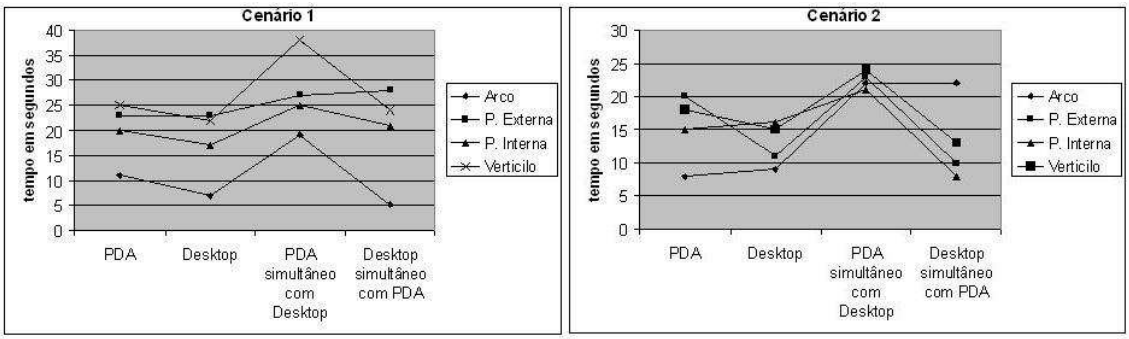


Figura 4. Resultados dos Testes dos Cenários 1 e 2

5.2. Avaliação dos Resultados

Ao analisar os quatro cenários de testes apresentados foi possível constatar que em geral: (i) o tempo de resposta da consulta com o acesso feito pelo *desktop* foi menor do que pelo PDA. Esse resultado já era esperado visto que o PDA precisa fazer acesso ao *Web Service* intermediário; (ii) em todos os cenários o tempo de resposta da consulta foi inferior quando executado somente pelo PDA e pelo *desktop* em momentos distintos em

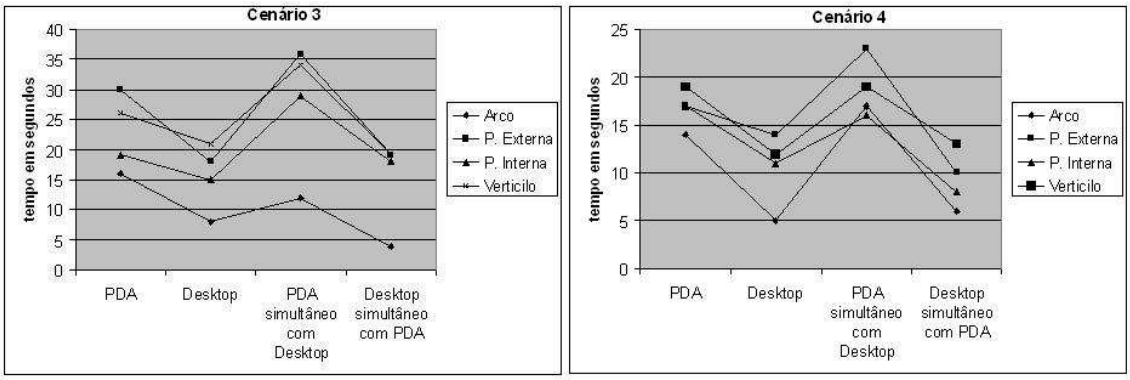


Figura 5. Resultados dos Testes dos Cenários 3 e 4

relação aos dois simultaneamente; (iii) o tempo de resposta da consulta foi menor quando aplicados filtros de classe e sexo em relação a utilização do filtro apenas por classe.

Entretanto, observou-se que o tempo de resposta da aplicação ainda não é satisfatório, visto que o ambiente, ao qual se propõe a arquitetura, possui bases de dados maiores do que as utilizadas e estarão distribuídas geograficamente, ao contrário dos testes realizados em uma rede local. Ao analisar separadamente cada tempo medido, especificado na Fórmula 1, identificamos que em determinados momentos o T_{AS} variou de 1 a 20 segundos, o que interfere consideravelmente no tempo total. Isso sugere a necessidade de executar um conjunto maior de testes.

Um dos requisitos da aplicação é a necessidade de agrupar os dados das minúcias em um sistema de arquivo. Pois, para cada registro compatível com a consulta aplicada, um arquivo de minúcias deve ser criado. Este procedimento é necessário visto que o sistema de comparação de minúcias, BOZORTH3, efetua o procedimento da comparação acessando esses arquivos. Contudo, como esperava-se, o tempo gasto na serialização das minúcias para arquivos texto é bastante significativo.

Na implementação inicial da arquitetura, o transporte dos dados agrupados eram enviados diretamente do OGSA-DAI ao GridFTP, porém o tempo de resposta encontrado não foi satisfatório para aplicação. Como alternativa, optamos por usar o agrupamento das consultas (*ResultSet*) que é encapsulado em mensagens SOAP. Só então ao receber as mensagens é que se cria os arquivos texto contendo as minúcias. Essa estratégia mais simples reduziu consideravelmente o tempo de resposta da aplicação em relação à utilização do GridFTP. Contudo mesmo os tempos nessa estratégia ainda não são satisfatórios.

6. Conclusão

Este artigo apresentou uma arquitetura para integração de dados de impressão digital, através da utilização de grades computacionais. Essa arquitetura serviu de base para criação de uma aplicação de identificação criminal, pelo reconhecimento de impressão digital. A aplicação foi desenvolvida e testes preliminares foram efetuados a fim de validar o seu funcionamento.

Os resultados indicam a necessidade de testes em maior escala e com maior nível de concorrência para ser possível observar como a infra-estrutura da grade computacional se comporta. No sentido de diminuir a sobrecarga de comunicação, pelo uso do OGSA-

DAI, é preciso avaliar por testes os diferentes métodos de entrega existentes para reconhecer uma melhor estratégia do que a usada atualmente. Em outro sentido, como o tempo de comparação de minúcias representa a maior parcela da computação, deve-se experimentar a distribuição do processamento do BOZORTH3, como sugere [Barbosa JR 2007], para, por exemplo, aproveitar a localidade dos dados já distribuídos e diminuir o tempo total.

Referências

Barbosa JR, A. A. (2007). Estudo e desenvolvimento de aplicação biométrica para ambiente em larga escala. Technical report, Universidade Federal da Bahia - UFBA, Salvador, Bahia. http://twiki.dcc.ufba.br/pub/Gaudi/ProjetoBiometrica/monografiaAmadeu.pdf.

Berry, D., Luniewski, A., and Antonioletti, M. (2007). Ogsa data architecture. Global Grid Forum.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture. W3C Web Services Architecture Working Group.

Cirne, W. and Santos, N. (2005). Grids computacionais: Da computação de alto desempenho a serviços sob demanda. Mini-curso no 23 Simpósio Brasileiro de Redes de Computadores.

Costa, S. M. F. (2001). Classificação e verificação de impressões digitais. Master's thesis, Universidade de São Paulo - USP.

Foster, I., Kesselman, C., Nick, J., and Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed systems integration. Global Grid Forum.

GT4 (2007). Welcome to globus. Disponível em http://www.globus.org/. Acesso em junho de 2007.

Kazienko, J. F. (2003). Assinatura digital de documentos eletrônicos através da impressão digital. Master's thesis, Universidade Federal de Santa Catarina, Florianópolis.

Maltoni, D., Maio, D., Jain, A. K., and Prabhakar, S. (2005). *Handbook of Fingerprint Recognition*. Springer Professional Computing. Springer, second edition.

Santos, L. A. S. (2006). Uma proposta de escalonamento descentralizado de tarefas para computação em grade. Master's thesis, Universidade Federal do Rio Grande do Sul - UFRGS.

Sotomayor, B. (2005). The globus toolkit 4 programmer's tutorial.

SSP (2005). Secretaria da segurança pública. Disponível em http://www.ssp.ba.gov.br/noticia.asp. Acesso em dezembro de 2007.

Venugopal, S., Buyya, R., and Ramamohanarao, K. (2006). A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1):3.

Watson, C. I., Garris, M. D., Tabassi, E., Wilson, C. L., Mccabe, R. M., and Janet, S. (2004). *User's Guide to NIST Fingerprint Image Software 2 (NFIS2)*. NIST - American National Standards, New York. Conteúdo sujeito a Lei de Controle de Exportação dos EUA.

A Grid Enabled Algorithm for the Multiple Resources Allocation Problem

Higor P. Vieira Neto , Lúcia M.A. Drummond , Vinicius T. Petrucci

[1]Department of Computer Science
Fluminense Federal University (UFF) – Brazil

{hneto,lucia,vpetrucci}@ic.uff.br

Abstract. *Typically, a Grid is composed by a collection of clusters, whose nodes are connected by high-speed dedicated links. Communication among nodes in distinct clusters is performed by much slower WANs. One would like to have scalable synchronization algorithms taking into account such hierarchical network topology. In this work we propose a token-based algorithm to solve the multiple resources allocation problem considering the usual hierarchical network topology of Grid environments. Few papers present a mutual exclusion algorithm specifically for Grids and usually they consider the sharing of a unique resource. However, it is typical in Grids to have several kinds of shared resources, including hardware such as RAM, disk space, communication channels, and software such as programs, files and data. We compared the proposed algorithm with a well known token-based algorithm for solving the multiple resource allocation problem, which does not consider the particularities of Grid environments. The reduction in the number of inter-cluster messages and the waiting time to enter a critical section makes the proposed algorithm very attractive in Grid applications.*

1. Introduction

Many distributed computations require that a resource is allocated to a single process at a time. Each process has code segments, called critical sections, in which the process access shared resources. The problem of coordinating the execution of critical sections is solved by providing mutual exclusive access in time to the shared resources. Algorithms to solve the mutual exclusion problem in distributed systems can be basically divided into two groups: permission-based (such as [Ricart and Agrawala 1981], [Raynal 1991], [Kakugawa et al. 1994] and [Raymond 1989]) and token-based (such as [DeMent and Srimani 1994], [Naimi et al. 1996], [Naimi 1993], [Bertier et al. 2006] and [Bouabdallah and Laforest 2000]) algorithms. In the first group, a process requires permissions from all other processes or from most of them in order to enter its critical section. In the second group, a unique token is shared among the processes and the possession of it guarantees the right to enter it. Token-based algorithms present an average lower message traffic with regard to the number of processes when compared to permissions-based ones, which present scalability problems.

Grids are computational environments containing a large number of nodes spread over large geographical areas. Usually, a Grid is composed by a collection of clusters, whose nodes are connected by high-speed dedicated links. Communication among nodes

in distinct clusters is performed by much slower WANs. One would like to have scalable synchronization algorithms taking into account such hierarchical network topology [Bertier et al. 2006].

In this work we propose a token-based algorithm to solve the M resources allocation problem considering the typical hierarchical network topology of Grid environments. Let $R = (r_1, k_1), (r_2, k_2), \ldots, (r_m, k_m)$ be the set of resources, such that for each resource r_i there are k_i instances of it, shared among a set of processes $P = P_1, P_2, \ldots, P_n$. Before entering in its critical section, a process P_i asks for a subset of the existing resources. P_i can not continue its execution before it has obtained all the resources it has requested, in order to prevent the well known deadlock problem. At the end of the critical section, the resources are released. In a future request, the process can ask for a different subset of resources.

There are few papers that propose a mutual exclusion algorithm specifically for Grids [Bertier et al. 2006] [Muhammad et al. 2005]. They usually consider the sharing of a single instance of a unique resource. However, it is typical in Grids to have several kinds of shared resources, including hardware such as RAM, disk space, communication channels, and software such as programs, files and data. See [CERN 2006] and [Han et al. 2006] for detailed examples of software and hardware sharing in Grids.

We compared the proposed algorithm with the token-based algorithm introduced by [Bouabdallah and Laforest 2000], that is a well known efficient algorithm for solving the M resource allocation problem, but does not consider the particularities of Grid environments.

The remaining of this paper is organized as follows. In Section 2, the related work is presented. In Section 3, we describe the proposed algorithm. Section 4 presents the strategies employed for measuring its performance. The experimental results are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

Several token-based algorithms have been proposed to solve the mutual exclusion problem in distributed environments. Some of them consider a hierarchical topology by gathering nodes into groups. For example, in [Housni and Trehel 2001] and [Chang et al. 1990], a based prioritized groups approach is presented, where nodes with the same priority are gathered into a same group. Both works propose a hybrid strategy, where the algorithm executed intra-group is different from the inter-group one. [Erciyes 2004] proposes an architecture based on rings of clusters, where each node in a ring represents a clusters of nodes. Despite of having a hierarchical approach, they do not consider different communication latencies. [Bertier et al. 2006] present a token based algorithm, inspired in the work proposed by [Naimi et al. 1996], that considers different latencies between different *sites*. That work is limited to a single shared resource. Our approach uses some of those hierarchical strategies proposed by [Bertier et al. 2006]. Moreover, our algorithm is extended to solve the problem of sharing of multiple resources with multiple instances. In that scenario, [Maddi 1997] presents algorithms based on the work introduced by [Raynal 1991], that considers the multiplicity of resources, but presents low performance in a large scale environment composed by many nodes. [Bouabdallah and Laforest 2000] also proposes a distributed algorithm for the dynamic

allocation problem, but does not consider latency disparities among spread nodes. Our proposed algorithm also employs some concepts introduced by them.

3. The Algorithm

3.1. General Idea of the Solution

The distributed environment considered consists of n processes each with a unique identifier, communicating asynchronously by exchanging messages. Messages are delivered in finite but unpredictable time.

A necessary condition for a process to execute a critical section is to have all the associated resource tokens. Moreover, the execution time of each critical section is finite but not bounded. The solution to this problem must guarantee the mutual exclusion property, at any moment a resource instance is used by at most one process, and the freedom from starvation property, any process requesting resources will obtain them in a finite time.

As we consider the multiple resources allocation problem, the proposed algorithm is based not only on tokens that grant resource accesses, but also on a unique Control Token that orders resources tokens grants, as proposed by [Bouabdallah and Laforest 2000]. The Control Token keeps a set of free tokens and a view, for each resource instance, of the process scheduled before the current owner of the Control Token. In order to a process execute its critical section it must request the Control Token and then, with the information about the necessary resources instances, forward requests to the processes which have or will have the corresponding required tokens.

The proposed algorithm aims at reducing inter-cluster messages and the average waiting time for resources. Concerning the Control Token messages, a leader process is elected in each cluster in order to concentrate initial local requests for the Control Token, that is eventually in another cluster. Moreover, local Control Token requests, i.e., requests issued by processes of the same cluster in which the Control Token is, are prioritized. Concerning the resource instance tokens, the algorithm also prioritizes the transmission of resource tokens among processes of the same cluster and defines priorities for processes with respect to each resource. A request for a resource will be forwarded to the process that has least frequently used it.

The algorithm satisfies the mutual exclusion requirement since a unique token is associated with each resource instance and only one process may have it at any moment. It is starvation-free, because each node requesting the Control Token will receive it in a finite time, and also because each node, needing tokens to execute its critical section, will obtain them in a finite time. The procedures of the proposed algorithm were inspired in the ones presented by [Bouabdallah and Laforest 2000] and [Bertier et al. 2006] and their analysis are also valid for our proposal.

The following subsections describe these procedures in detail.

3.2. Stage 1 - Obtaining the Control Token

When a process P_i needs a set of resources to execute its critical section, it first checks if the corresponding tokens are available locally. If so, it locks them and executes its critical section. Otherwise, it requests the Control Token.

Only one process at a time keeps the Control Token that consists of:

- a set A of free tokens;
- a list B of processes, each with the set of tokens that it has or will have;
- a list C of resources, each with a process that has the priority to access it and an associated value that represents its use frequency.

The process having the Control Token, takes all needed tokens that are in the set A, adds to A the unneeded tokens that it has and updates the list B to indicate that it has or will have all the tokens it needs. This last action will cause the sending of resource token requests. The C list is used to choose the most appropriate processes of list B to send such requests. Upon receiving all the corresponding answers, the Control Token can be released to a next process.

In each cluster a process is chosen as a leader to which is sent the first Control Token request of each process of that cluster. At the beginning, the Control Token is allocated to one of these leaders and requests from other leaders are forwarded to it. Each leader keeps the identification of the last local request to avoid inter-cluster transmissions. There are here two cases to be considered. In the first one, consider a cluster A, that does not have the Control Token and where a process P_i sends a Control Token request to its leader, L_a. If another local process P_j has previously asked for the token and L_a is aware of it, L_a redirects the P_i's request to P_j avoiding a transmission to the remote cluster. Note that the original request of P_j required that L_a forwarded the message to the cluster leader that has the Control Token. See the order in which the requests were sent in Figure 1. When the Control Token arrives at the cluster A, it follows the path shown in Figure 2.

In the second case, we have a similar scenario but with the Control Token at cluster A. Consider now that L_a receives a remote request originated from a process P_l of a cluster B. L_a forwards it to the last process of the cluster which has requested the Control Token, in our example P_i, but continues to consider P_i as the last process that has requested the Control Token. Upon the receiving of P_l's Control Token request, P_i will consider it as the next to receive the Control Token. However, if process P_i receives a Control Token request from a local process P_k, it may put the previous P_l request in a queue, considering P_k as the next candidate to receive the Control Token and sending the queue to P_k. Thus, P_k will consider P_l as the next to receive the Control Token. See in Figure 3 the path followed by the Control Token messages in this case. In order to avoid starvation, preemptions are limited to a fixed number of times that is an algorithm parameter.

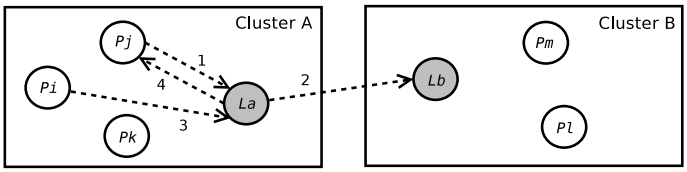


Figure 1. Control Token Requests

Remark that as the algorithm progresses, the next Control Token request of a process is sent to the other process that of its knowledge has or had the Control Token. In both cases previously described, a next request of P_j would be sent to P_i. More details of this procedure may be found in [Bertier et al. 2006].

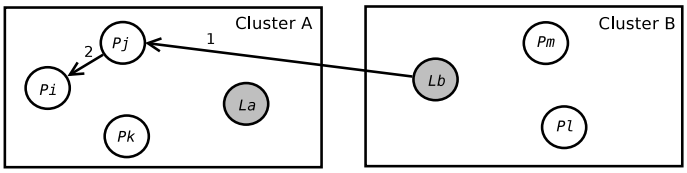


Figure 2. Control Token Path 1

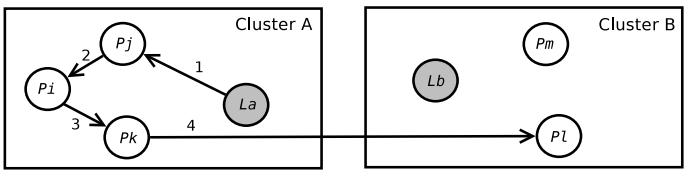


Figure 3. Control Token Path 2

3.3. Stage 2 - Obtaining Resource Tokens

During the execution of the algorithm, a token can be either in the Control Token or with some process. In this last case, the token can be either locked, when the process is using the corresponding resource instance, or free.

At first, a process tries to find all the needed tokens in the set A of the Control Token. In order to take the most suitable tokens of set A, a process can consider several criteria such as the physical locations of the resources that the tokens represent or characteristics of its execution. There are several works in literature that tackle the resource selection problem for Grid applications, see for example [Liu et al. 2002]. When a process does not find all the needed tokens in the set A, by using the list B, it determines the processes to which it has to request the unavailable tokens. In the proposed algorithm, a process chooses another one from list B to send a resource request obeying the following decreasing priority order: 1) a local process that has not frequently used the resource, 2) a local process that has frequently used the resource, 3) a remote process that has not frequently used the resource, and, finally, 4) a remote process that has frequently used such resource.

The Control Token keeps a list C of resources, each with a process that has the priority to access it and a corresponding counter. Every process keeps for each resource a counter that is increased each time it needs it. A process is considered a candidate to have priority over a resource when such counter reaches a predefined value, that is an algorithm parameter. Upon owning the Control Token, a candidate process may update the list C in one of the two cases: when there is not any other process with priority over that resource or when its local counter is greater than the corresponding value found in the Control Token.

Upon receiving a resource request, a process sends back immediately all the requested tokens that are free in a message called *ACK1*, that may be an empty message. If some of the requested tokens are locked or not yet available locally, a second message, called *ACK2* will be sent, containing the rest of the other requested tokens, when they become available.

When a process, having the Control Token, receives all *ACK1* messages, it then locks the received tokens and is able to send the Control Token to a next process. If

it received in *ACK1* messages all the tokens that it has requested, it enters its critical section. Otherwise, it waits for *ACK2* messages that will contain the rest of the requested tokens and enters its critical section.

When a process exits its critical section it may send *ACK2* messages to nodes that are waiting for tokens and sets locally that the remaining tokens are free.

4. Algorithm Analysis

In Grids, the wall clock time of two different executions of the same program may vary depending on the external load in the machines. Therefore, we used an additional measure of performance that does not rely on wall clock times.

In order to measure the waiting time for entering the critical section, we also adopted a logical clock as suggested in [Drummond and Barbosa 2003]. The employed logical clock is based on vector clocks [Fidge 1988] [Mattern 1994] with some changes to reflect the different latencies of links in Grid environments.

In our proposed algorithm, in order to access a critical section, a process, from the moment it requests the critical section, must wait for the Control Token and for the resource tokens. The total time required for a process to enter its critical section is obtaining by summing these waiting times. By the proposed algorithm, the messages for requesting the Control Token follow the same path defined by the Control Token messages themselves, possibly composed of links with different latencies. Then to monitor the waiting for the Control Token it must only to monitor the Control Token messages. Analogously, a resource token request may arrive at a process that does not have it yet, i.e., a process that is also waiting for *ACK2* messages to enter its critical section and then exit it and send the corresponding *ACK2* messages, forming a chain of *ACK2* messages.

Considering these facts, the proposed logical clock aims at capturing the longest chain of token messages with causal relation in the worst case. Such worst case happens when a process requests a token immediately after it releases this, having to wait eventually that several other processes receive it and release before the token reaches the original process again. The logical clock is constructed by letting each process P_i maintains an n elements vector, V_i, initialized with zeros, that is piggybacked along with the token messages. As two kinds of messages are monitored, Control Token and *ACKs*, two vectors are necessary, but the rules to update them are similar.

The rules to update the Control Token vector are presented next.

Algorithm 1 - Logical Clock - *Process* P_i

1: **(R.1) Upon receiving a Control Token message from process** P_j **(Control Token,** V_j**):**
2: $controltoken\_waiting\_time \leftarrow V_j[i]$;
3: **for** each entry k of V_i **do**
4: $V_i[k] \leftarrow$ max($V_i[k]$,$V_j[k]$);
5: $V_i[i] \leftarrow 0$;

1: **(R.2) Upon sending a Control Token message to process** P_j **on a link with latency** x**:**
2: **for** each entry k of V_i **do**
3: $V_i[k] \leftarrow (V_i[k] + x)$;
4: Send $\langle ControlToken, V_i \rangle$ to P_j

See Figure 4 for an example involving four messages exchanges by four processes

connected through links with different latencies, and the corresponding logical clocks.

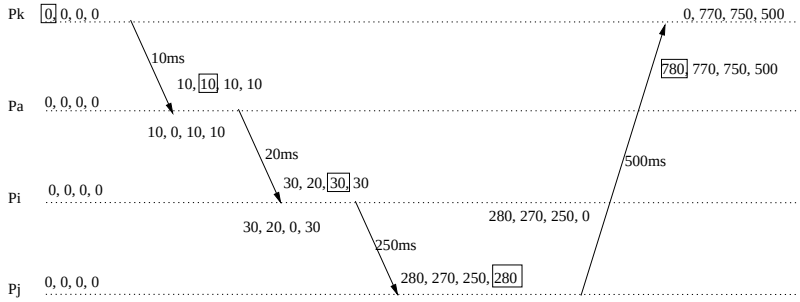


Figure 4. Logical Clocks

Although the algorithms for updating the Control Token and *ACKs* vectors are the same, the methods for calculating the waiting times for the resource tokens and the Control Token are not equal. In the case of resource token messages, one should consider the greatest waiting time among all *ACK2* and *ACK1* messages. Thus, line 2 of R1 is replaced by "$resourcetoken\_waiting\_time \leftarrow$ max($resourcetoken\_waiting\_time$,$V_j[i]$)".

As the waiting chains for the Control Token and the Resource Tokens occur sequentially, in the worst case the waiting time for a process P_i to execute its critical section is the sum of the $controltoken\_waiting\_time$ with the $resourcetoken\_waiting\_time$.

In our experiments using logical clock we consider that the same instances are requested in each critical section. We also measured the number of messages exchanges, divided into two categories: messages exchanged among processes in the same cluster (local messages) and messages among processes from different clusters (remote messages).

5. Experimental results

The distributed algorithms were implemented in Python, version 2.5, with PyMPI as a MPI wrapper, running under LAMMPI (version 7.1.4) for message-passing. Initially, we compared our algorithm with the proposed by Laforest et al. considering multiple resources each with a unique instance. For further analysis we adapted the Laforest et al.'s algorithm for multiple instances as proposed in their paper.

We emulated a Grid environment composed of 48 nodes divided into three clusters (C0, C1 e C2), each with 16 nodes. We considered that there was only one process scheduled at each node. The results are averages of five executions. The average standard deviation of the execution times was around 5.0 %. Each execution consists of the following steps repeated 10 times with intervals of 500 milliseconds: to enter its critical section, a process requests a set of resource instances randomly selected, executes its critical section for 500 milliseconds and exit it. We used as the preemption threshold the value 8, i.e., a process can not be put in a queue more than 8 times. We consider 10 types of resources, each of them with a number of instances varying from 1 to 10. In our approach we also consider that a process can obtain priority over a resource when it uses this one, successively, at least 6 times.

At first we evaluated the average logical times, as defined in Section 4; and the number of messages exchanged, divided in two categories: inter and intra cluster, called

remote and local, respectively. These experiments were executed on 24 computers, connected through a 1.0 Gbits/s Ethernet. Each computer has either two Intel Xeon processors or two AMD Opteron dual core processors, and 16 Gbytes RAM memory, making 60 processors in total, running GNU/Linux with kernel version 2.4.1.

Figure 5 presents, for the resources with a unique token, the average logical time for obtaining the Control Token (part a), for obtaining the tokens (part b), and the average sum of these previous times (part c), varying the communication latency among processes from different clusters from *1 ms* to *750 ms* and considering the local latency equal to 1 ms. This figure also presents the average of local, remote and total messages exchanged (part d). Our algorithm presents the smallest waiting times, as expected, since it concentrates local messages and reduces the causal chains of messages involving inter-cluster links. Our algorithm shows to be more efficient when the inter-cluster latencies increase. It is also observed that our algorithm reduces the number of remote messages and increases local messages, by concentrating local requests.

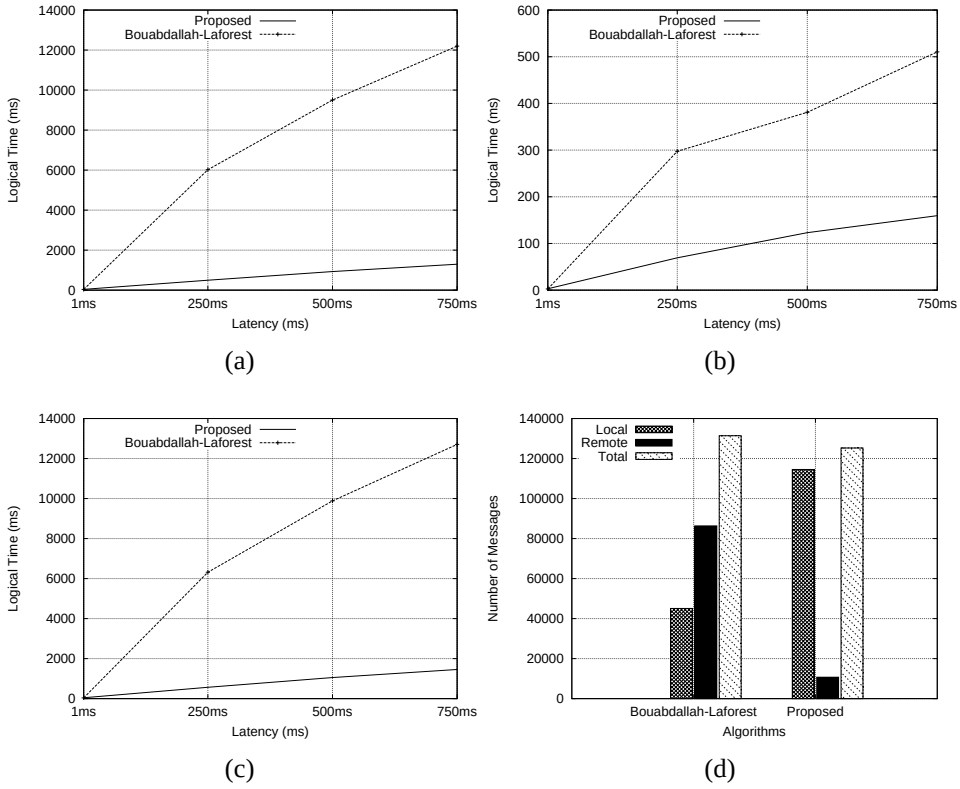


Figure 5. Logical Clock - Multiple resources, one instance: (a) Logical time to obtain the Control Token (b) Logical time to obtain the most distant token (c) Logical time to access the critical section (d) Number of messages

The next figure presents the obtaining time, considering wall-clock time. In order to simulate the different latencies, two strategies were employed. One of them uses the *sleep* function of the Python to delay the message delivering, simulating different link latencies. The other employs NetEm that is a tool that allows the simulation of network properties, in our experiments we used it to insert delays directly in the packets [Hemminger 2005]. The experiments using NetEm were executed on 24 Intel Pentium

IV processors dedicated exclusively to our application, connected through a 1.0 Gbits/s Ethernet, and a Grid environment composed of 24 nodes divided into three clusters, each with 8 nodes, was emulated on them.

In both cases, we observed that the algorithms presented a similar behavior of the previous tests. Remark that the wall clock times include not only the latencies but the times in the critical sections and the time intervals between the requests as well. So, we can observe a increasing of time in the tests with *sleep* function when compared with the tests with the logical clock that were executed in the same environment.

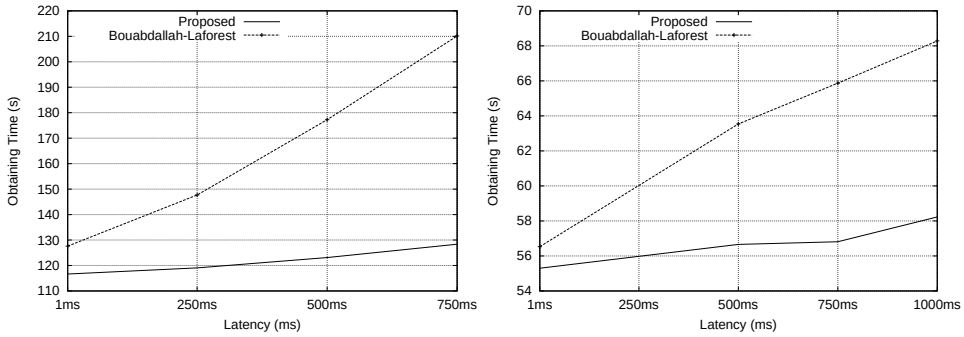


Figure 6. Wall Clock Time - Multiple resources, one instance: (a) Obtaining time using *sleep* (Python) (b) Obtaining time using NetEm

We repeated the same tests considering multiple instances for each resource. Considering the wall-clock time, in these tests only the *sleep* function was used, since the previous tests with NetEm presented similar results. In Figures 7 and 8, we can observe that the algorithms showed a similar behavior with a slight increase in the gap between the two algorithms.

The next experiments aims at evaluating the scalability of the proposed algorithm considering the following parameters: the numbers of nodes, resource types and clusters. For each test, we kept the same configuration of the previous experiments, only changing the analyzed parameter. The communication latency among clusters was fixed to 500 ms. The results shown in Figures 9(a) (Number of Nodes) and 9(c) (Number of Resource Types) present a growing of the obtaining time for the two algorithms and point out that our approach scales better than the Bouabdallah-Laforest algorithm. In Figure 9(b) (Number of Clusters) the proposed algorithm keeps a constant result while the other algorithm has a huge growing of the obtaining time. Remark that with the growing of the number of clusters, the number of remote links also increases. Because in the Bouabdallah-Laforest algorithm there is no distinction between remote and local nodes, more remote messages are exchanged so that a node enters its critical section, as the number of clusters increases.

6. Concluding Remarks

We presented an algorithm for solving the multiple resource allocation problem that considers the communication latency heterogeneity of a Grid environment. The proposed algorithm was compared with the one presented by [Bouabdallah and Laforest 2000]. In

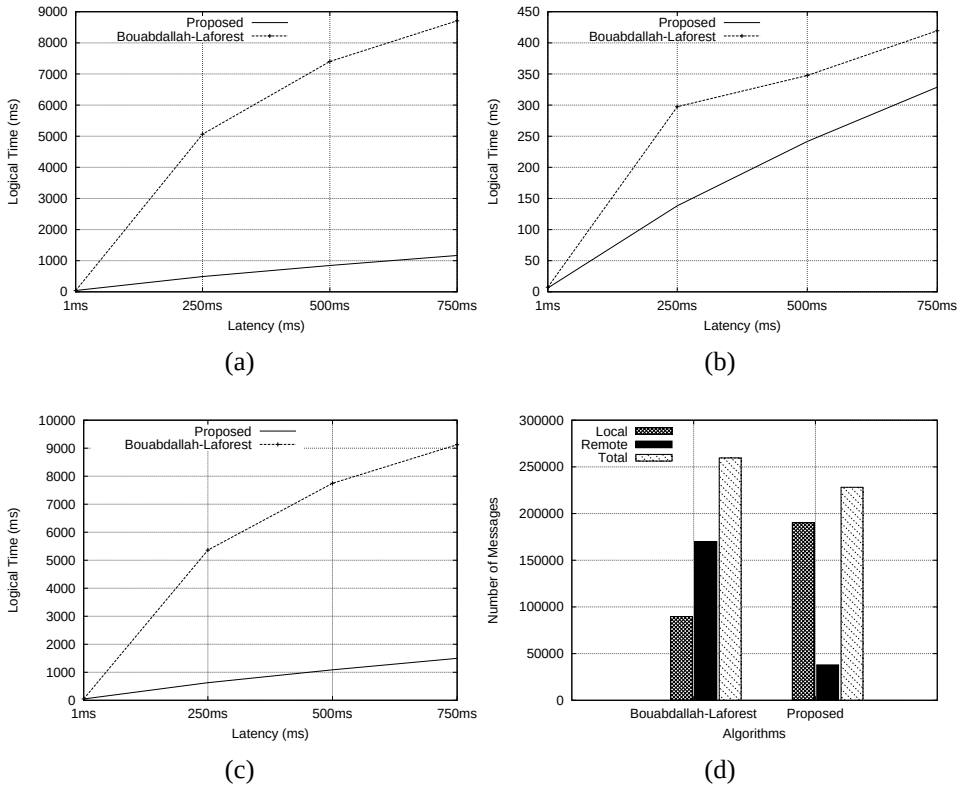


Figure 7. Logical Clock - Multiple resources, multiple instances: (a) Logical time to obtain the Control Token (b) Logical time to obtain the most distant token (c) Logical time to access the critical section (d) Number of messages

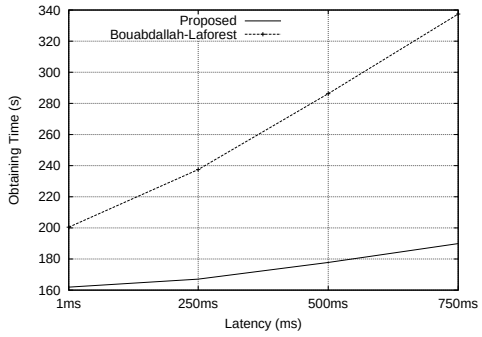


Figure 8. Wall Clock Time - Multiple resources, multiple instances - using *sleep* (Python)

order to evaluate these algorithms, we also proposed a logical clock to measure the waiting time for a process to obtain the Control Token and the resource tokens, and consequently enter its critical section. Such logical clock considers the different link latencies. We repeated the tests considering wall-clock times and adopting two different strategies for the emulation of a Grid. The results obtained by those different ways of evaluation were very close and testified that our algorithm is more efficient than the proposed by [Bouabdallah and Laforest 2000].

Summarizing, the reduction in the number of inter-cluster messages and the wait-

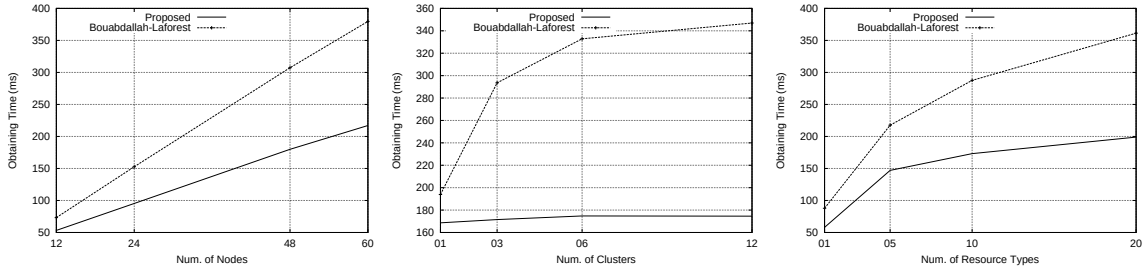


Figure 9. Wall Clock Time - Multiple resources, multiple instances: (a) Number of Nodes (b) Number of Clusters (c) Number of Resource Types

ing time to enter a critical section makes the proposed algorithm very attractive in Grid applications.

References

Bertier, M., Arantes, L., and Sens, P. (2006). Distributed mutual exclusion algorithms for grid applications: a hierarchical approach. *J. Parallel Distrib. Comput.*, 66(1):128–144.

Bouabdallah, A. and Laforest, C. (2000). A distributed token-based algorithm for the dynamic resource allocation problem. *SIGOPS Oper. Syst. Rev.*, 34(3):60–68.

CERN (2006). Large hadron collider, global grid service for lhc computing succeeds in gigabyte-per-second challenge. `http://press.web.cern.ch/Press/PressReleases/Releases2006/PR03.06E.html`. [Online; accessed 2007.11.07].

Chang, Y.-I., Singhal, M., and Liu, M. (31 Oct-2 Nov 1990). A hybrid approach to mutual exclusion for distributed systems. *Computer Software and Applications Conference, 1990. COMPSAC 90. Proceedings., Fourteenth Annual International*, pages 289–294.

DeMent, N. S. and Srimani, P. K. (1994). A new algorithm for mutual exclusions in distributed systems. *Journal of Systems and Software*, 26(2):179–191.

Drummond, L. M. A. and Barbosa, V. C. (2003). On reducing the complexity of matrix clocks. *Parallel Computing*, 29(7):895–905.

Erciyes, K. (2004). Distributed mutual exclusion algorithms on a ring of clusters. In Laganà, A., Gavrilova, M. L., Kumar, V., Mun, Y., Tan, C. J. K., and Gervasi, O., editors, *ICCSA (3)*, volume 3045 of *Lecture Notes in Computer Science*, pages 518–527. Springer.

Fidge, C. J. (1988). Timestamps in message-passing systems that preserve the partial ordering. In *Proc. 11th Australian Comp. Sci. Conf.*, pages 56–66.

Han, H., Jung, H., Yeom, H. Y., Kweon, H. S., and Lee, J. (2006). HVEM grid: Experiences in constructing an electron microscopy grid. In Zhou, X., Li, J., Shen, H. T., Kitsuregawa, M., and Zhang, Y., editors, *APWeb*, volume 3841 of *Lecture Notes in Computer Science*, pages 1159–1162. Springer.

Hemminger, S. (2005). Network emulation with netem. Technical report, Open Source Development lab. `http://devresources.linux-foundation.org/shemminger/netem/LCA2005_paper.pdf`, [Online; accessed 2008.03.19].

Housni, A. and Trehel, M. (2001). Distributed mutual exclusion token-permission based by prioritized groups. *Computer Systems and Applications, ACS/IEEE International Conference on. 2001*, pages 253–259.

Kakugawa, H., Fujita, S., Yamashita, M., and Ae, T. (1994). A distributed k-mutual exclusion algorithm using k-coterie. *Information Processing Letters*, 49(4):213–218.

Liu, C., Yang, L., Foster, I., and Angulo, D. (2002). Design and evaluation of a resource selection framework for grid applications. In *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, page 63, Washington, DC, USA. IEEE Computer Society.

Maddi, A. (1997). Token based solutions to M resources allocation problem. In *SAC*, pages 340–344.

Mattern (1994). Virtual time and global states of distributed systems. In *Zhonghua Yang and T. Anthony Marsland (Eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press*.

Muhammad, M., Cheema, A. S., and Gupta, I. (2005). Efficient mutual exclusion in peer-to-peer systems. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 296–299, Washington, DC, USA. IEEE Computer Society.

Naimi, M. (1993). Distributed algorithm for k-entries to critical section based on the directed graphs. *SIGOPS Oper. Syst. Rev.*, 27(4):67–75.

Naimi, M., Trehel, M., and Arnold, A. (1996). A log (n) distributed mutual exclusion algorithm based on path reversal. *J. Parallel Distrib. Comput.*, 34(1):1–13.

Raymond, K. (1989). A distributed algorithm for multiple entries to a critical section. *Inf. Process. Lett.*, 30(4):189–193.

Raynal, M. (1991). A distributed solution to the k-out of-M resources allocation problem. *Lecture Notes in Computer Science*, 497:599–609.

Ricart, G. and Agrawala, A. K. (1981). An optimal algorithm for mutual exclusion in computer networks. *Commun. ACM*, 24(1):9–17.

Interoperação de Grades Móveis Ad hoc com Grades Fixas

Bruno F. Bastos , Luciana S. Lima , Antônio Tadeu A. Gomes , Artur Ziviani

Laboratório Nacional de Computação Científica (LNCC)
Av. Getúlio Vargas, 333 – 25651-075 – Petrópolis, RJ

{bfbastos,lslima,atagomes,ziviani}@lncc.br

Abstract. *Mobile ad hoc grids are regarded as a promising approach for applications such as emergency response networks and research field systems. Despite being originally thought of as isolated processing environments, such grids may considerably increase their computing power whenever they approach a fixed network, due to the resources available in wired grids potentially accessible through such a network. The interoperation of mobile ad hoc grids and resources available in wired grids is, however, a problem still to be tackled in the literature. This paper proposes two different interoperation approaches, and provides a qualitative assessment of their implications. Two grid middlewares are used for such analysis. These are also employed to prototype the proposed approaches, so as to demonstrate their feasibility and provide a basis for the development of applications that can build on such interoperation.*

Resumo. *Grades móveis ad hoc têm sido apontadas como uma abordagem promissora para aplicações como redes de resposta a incidentes, sistemas de trabalho de campo em áreas isoladas, entre outras. Embora essas grades sejam vistas originalmente como agrupamentos de processamento isolados, a aproximação das mesmas a uma rede fixa permite expandir enormemente a sua capacidade computacional graças aos recursos disponíveis na rede fixa. A interoperação entre grades móveis ad hoc e recursos na rede fixa é, contudo, um problema ainda não abordado na literatura. O objetivo principal do presente trabalho é justamente explorar esse problema. Este artigo propõe duas abordagens distintas de interoperação, e avalia qualitativamente as implicações de cada uma delas. Dois middlewares de grade são usados como base para essa análise. Esses middlewares são empregados também na prototipação das abordagens propostas, a fim de demonstrar a sua exeqüibilidade e oferecer uma base para o desenvolvimento de aplicações que possam valer-se dessa interoperação.*

1. Introdução

Embora inicialmente pensadas para redes fixas, grades computacionais[1] têm recebido atenção crescente quanto à sua aplicação para redes sem fio [McKnight et al. 2004]. A principal justificativa para isso é que, apesar de dispositivos móveis (laptops, PDAs e celulares) terem capacidade computacional em geral comparativamente menor a de outros computadores, a disponibilidade e uso desses dispositivos têm crescido consideravelmente nos últimos anos. Isso cria um potencial enorme para o compartilhamento dos recursos desses dispositivos, cujo uso atual tem sido predominantemente pessoal.

[1]Neste artigo o termo genérico *grade* é usado para indicar um tipo particular de grade, que permite o compartilhamento de recursos como tempo de CPU e espaço de memória para execução distribuída de tarefas computacionalmente complexas. Outros tipos de grades – grades de dados, de instrumentos etc. – não são discutidos neste artigo.

Há duas abordagens principais de uso de dispositivos móveis em grades. Na primeira abordagem, dispositivos móveis podem participar como clientes ou mesmo como provedores (nós de processamento) no contexto de uma grade fixa pré-existente [Bruneo et al. 2003, Pinto et al. 2007, Silva et al. 2007]. Na segunda abordagem, esses dispositivos podem formar de maneira espontânea grades puramente sem fio – as chamadas 'grades móveis ad hoc' [Marinescu et al. 2003, Lima et al. 2005].

Grades móveis ad hoc oferecem uma abordagem bastante promissora para aplicações que demandam maior disponibilidade de poder computacional mas que, simultaneamente, têm como alvo condições em que a infra-estrutura de comunicação pode ser – ou se tornar repentinamente – indisponível, como redes de formação rápida (HFNs – *Hastily Formed Networks*) [Denning 2006] para resposta a incidentes, sistemas de trabalho de campo em áreas isoladas, entre outras. Contudo, o interesse recente de governos e empresas no desenvolvimento de tecnologias que permitam o estabelecimento de redes sem fio ad hoc de larga escala também em ambientes urbanos [BBN Technologies 2007] abre uma nova perspectiva na área de computação móvel. Isso porque, embora grades móveis possam a princípio ser pensadas como agrupamentos isolados de processamento, a aproximação de uma grade móvel a um ponto de acesso sem fio para uma rede fixa – comum em ambientes urbanos (p.ex. *hot spots* WiFi em aeroportos, centros comerciais e restaurantes) – permite expandir enormemente, mesmo que de forma temporária, a capacidade computacional da grade móvel graças aos recursos presentes na rede fixa. Esses cenários colocam um novo problema ainda não devidamente explorado tanto na literatura de computação móvel quanto na de grades computacionais: a *interoperação* de grades fixas com grades móveis ad hoc – problema este decorrente do fato de se tratarem de sistemas distribuídos distintos, com padrões e protocolos de controle bem diferentes. O objetivo principal do presente trabalho é justamente explorar esse problema, dando um primeiro passo para preencher essa lacuna atualmente presente na literatura.

Este artigo analisa os requisitos necessários à interoperação de grades móveis ad hoc com grades fixas. São considerados aspectos relacionados às fases de autenticação de usuários, de descoberta e seleção de recursos, e de submissão de tarefas. A partir dessa análise, são propostas duas abordagens distintas de interoperação. As implicações de cada uma das abordagens são avaliadas qualitativamente em termos do grau de transparência oferecido para a grade móvel, da eficiência no uso dos recursos dessa grade (em particular, recursos de comunicação) e da facilidade de implantação do proxy na infra-estrutura de rede fixa. Dois middlewares de grade são usados como base para a análise apresentada e para as abordagens propostas: o MoGrid [Lima 2007] para grades móveis ad hoc, e o Globus Toolkit [Foster e Kesselman 1997] para grades fixas. Esses middlewares são usados também na prototipação das abordagens de interoperação propostas, a fim de demonstrar a sua exeqüibilidade e oferecer uma base para o desenvolvimento de aplicações que possam valer-se dessa interoperação.

O restante do artigo está estruturado como se segue. A Seção 2 introduz os middlewares usados como base para a análise e as soluções de interoperação apresentadas nas Seções 3 e 4, respectivamente. A Seção 5 apresenta detalhes de implementação dos protótipos desenvolvidos como prova de conceito para essas soluções. Trabalhos relacionados são discutidos na Seção 6. A Seção 7 apresenta algumas considerações finais e levanta algumas perspectivas de trabalho futuro.

2. Conceitos Básicos

2.1. MoGrid

O middleware MoGrid permite a organização de dispositivos móveis como uma grade móvel puramente ad hoc. Esse middleware define dois papéis principais em uma grade móvel: iniciadores e colaboradores. Iniciadores são os dispositivos responsáveis pela submissão de tarefas, e colaboradores os responsáveis pela disponibilização de recursos computacionais para execução dessas tarefas.

O MoGrid utiliza um protocolo, denominado DICHOTOMY (*DIscovery and sCHeduling prOTOcol for MobilitY*) [Gomes et al. 2007], para selecionar, durante a fase de descoberta de recursos, os colaboradores mais aptos (isto é, que ofereçam os recursos lógicos necessários e que tenham mais recursos físicos disponíveis) a serem usados por um iniciador. O protocolo DICHOTOMY se baseia no acesso sob demanda às informações sobre os recursos dos colaboradores para efetuar essa seleção. Mais especificamente, não são usados anúncios periódicos dessas informações pelos colaboradores, como ocorre na maioria dos protocolos de descoberta de serviços [Lima et al. 2007a], devido à característica altamente dinâmica das mesmas. Assim, é necessário que um iniciador requisite explicitamente recursos da grade móvel ad hoc para execução de suas tarefas computacionais. Isso é feito por meio da difusão na rede sem fio de mensagens DICHOTOMY de descoberta de recursos. Essas requisições são respondidas pelos outros dispositivos da grade móvel ad hoc em função de sua aptidão como colaboradores.

A seleção dos colaboradores mais aptos é feita automaticamente, por meio de dois algoritmos [Lima 2007]: (i) o algoritmo DR (*Delayed Replies*), que faz com que as mensagens de resposta dos colaboradores mais aptos sejam enviadas primeiro na grade móvel ad hoc (colaboradores inaptos, ou seja, que não oferecem os recursos lógicos necessários, não respondem a requisições); e (ii) o algoritmo SbV (*Suppression by Vicinity*), que filtra mensagens de resposta dos colaboradores menos aptos ao longo dessa grade, conforme elas vão sendo encaminhadas em direção ao iniciador. Conforme demonstrado em [Gomes et al. 2007], a combinação dos dois algoritmos permite um balanceamento de carga entre os colaboradores, ao mesmo tempo que reduz a carga imposta à rede sem fio ad hoc devido ao encaminhamento das mensagens de resposta em direção ao iniciador.

A partir da descoberta e seleção dos recursos, o iniciador pode enviar tarefas para serem executadas pelos colaboradores. No MoGrid, assume-se a premissa que a formação de uma grade móvel ad hoc é baseada na colaboração entre os usuários dos dispositivos, não sendo adotado esquema algum de autenticação.[2] O MoGrid prevê, a princípio, o suporte a aplicações formadas somente por tarefas independentes (ao estilo *bag-of-tasks*). O envio dessas tarefas aos colaboradores é feita via um protocolo cliente-servidor simples denominado BoTDP (*Bag-of-Tasks Dispatcher Protocol*) [Lima 2007].

Atualmente, o MoGrid é quase todo implementado em Java. O único módulo desse middleware, implementado em C, que é dependente de plataforma é o monitor usado na coleta de informações sobre recursos físicos dos dispositivos. Esse monitor é adaptado da implementação disponível no middleware MoCA [Sacramento et al. 2004].

[2]Em [Lima 2007] são discutidos aspectos relacionados a essa premissa, como a necessidade de adoção de mecanismos de reputação que evitem situações similares às que ocorrem em redes P2P, como a presença de *free riders*.

2.2. Globus Toolkit

O middleware Globus Toolkit (GTK) permite que tarefas computacionais sejam submetidas a máquinas de uma grade, de acordo com os recursos disponíveis nessas máquinas e com as necessidades específicas das tarefas em questão. Para prover essas funcionalidades, o GTK disponibiliza vários serviços, sendo os mais importantes no contexto deste trabalho descritos resumidamente a seguir:

GSI (*Grid Security Infrastructure*). Provê segurança aliada à facilidade de uso, permitindo que o usuário se autentique globalmente na grade para submeter um determinado conjunto de tarefas para diferentes máquinas da mesma (*single sign-on*).

MDS (*Monitoring and Discovery System*). Disponibiliza informações sobre recursos da grade. O MDS provê um serviço central de informações, responsável por consolidar as informações recebidas de cada máquina da grade, e responder a requisições de descoberta por parte dos usuários da grade.

GRAM (*Grid Resource Allocation and Management*). Provê mecanismos de alocação de recursos e de criação e monitoração de tarefas. O GRAM depende de outros serviços, cuja implementação varia de acordo com a versão do GTK utilizada. Dentre esses serviços, destacam-se: (i) *file staging*, para transferência de arquivos entre as máquinas da grade (p.ex. via GridFTP); e (ii) *file streaming*, para transmissão de fluxos de dados entre tarefas (p.ex. *stdin*, *stdout* e *stderr*), permitindo assim a comunicação entre as mesmas.

O GTK oferece APIs para desenvolvimento de aplicações em grade em diversas linguagens de programação, incluindo Java, a qual foi escolhida para o desenvolvimento da solução proposta neste trabalho em função da facilidade de integração com o MoGrid. Neste trabalho, foi usada como base da implementação a versão 2.4 do GTK, que é a presente no ambiente de teste utilizado, conforme descrito na Seção 5.

3. Requisitos de Interoperação

Várias questões podem ser levantadas no que tange a interoperação de grades móveis ad hoc com grades fixas. Primeiro, grades móveis necessitam de padrões alternativos de controle, graças à natureza altamente dinâmica das redes sem fio ad hoc – por exemplo, dispositivos podem se mover, ou as propriedades de QoS (*Quality of Service*) do meio sem fio podem variar em curto espaço de tempo. É necessário, portanto, compatibilizar esses padrões de controle com aqueles tradicionalmente usados para grades fixas – que, por sua vez, assumem um ambiente muito mais estável. Além disso, é importante que a responsabilidade pela interoperação seja, o quanto possível, balanceada entre diferentes dispositivos móveis com capacidade de acesso à rede fixa, de modo a evitar a sobrecarga de um dispositivo específico devido ao encaminhamento intenso de mensagens dos outros dispositivos móveis interessados em ter acesso aos recursos da grade fixa. O alto dinamismo inerente às grades móveis torna ainda mais desafiador esse aspecto.

Nas soluções propostas neste artigo, os dispositivos de uma grade móvel ad hoc baseada no MoGrid podem também participar como clientes no contexto de uma ou mais grades fixas baseadas no GTK. A intermediação dessa comunicação é provida por proxies instalados na rede fixa, como se vê na Figura 1. Um proxy pode atender a uma grade específica (proxies P1 e P2), ou ainda a um conjunto de grades (proxy P3).

Um requisito fundamental para as soluções propostas é fazer com que um proxy

seja visto pela grade móvel ad hoc como um agrupamento de outros dispositivos que provêem recursos, de modo similar a qualquer outro colaborador na mesma. Essa transparência permite reduzir o impacto de alterações nos protocolos e na implementação atual do MoGrid, bem como facilita futuramente a interoperação do mesmo com outros middlewares de grade fixa [Goldchleger et al. 2004].

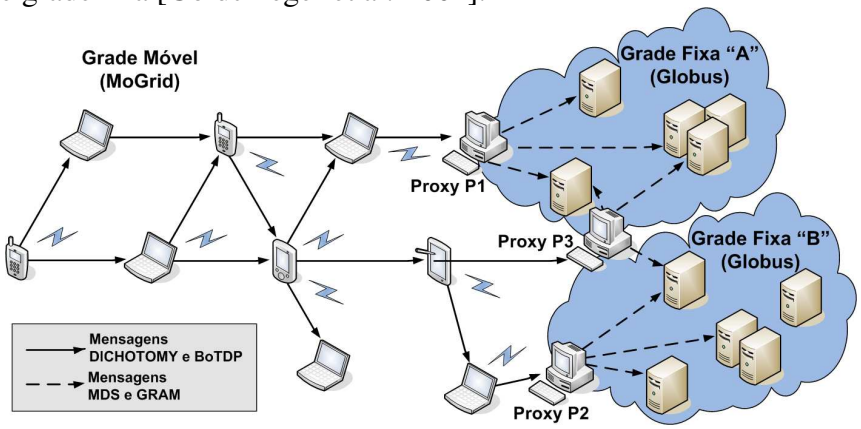


Figura 1. Interoperação de grades móveis e grades fixas com proxy.

As subseções que se seguem tratam de requisitos específicos relativos à interoperação dos middlewares MoGrid e GTK, e apresentam em linhas gerais as soluções concebidas neste trabalho para esses requisitos. Os detalhes sobre a implementação de cada uma dessas soluções são descritos na Seção 5.

3.1. Autenticação

No MoGrid, dispositivos móveis atuando como iniciadores não precisam se autenticar para submeter suas tarefas para outros dispositivos móveis na grade móvel ad hoc. Já o GTK trabalha com um sistema de segurança forte, baseado em certificados digitais, para autorizar a submissão de tarefas pelos usuários da grade. Um sistema de autenticação deve, portanto, ser criado no proxy, com a função de controlar as submissões na grade fixa ao qual esse proxy está associado, com base em alguma credencial fornecida pelo usuário do dispositivo móvel iniciador. Essa credencial tem por objetivo associar o usuário da grade móvel a um certificado de usuário válido na grade fixa. A autenticação do usuário deve a princípio ser local em cada grade, ou seja, caso o dispositivo móvel descubra diferentes proxies associados a diferentes grades, ele tem que se autenticar nesses proxies usando credenciais possivelmente distintas. Vale destacar que a implementação desses proxies deve considerar premissas de segurança similares a de qualquer outro sistema de autenticação, de modo a evitar o comprometimento desse proxy por um atacante, o que possibilitaria ao mesmo acesso aos certificados de usuário da grade fixa.

A autenticação proposta neste trabalho usa um processo de desafio-resposta [Kaufman et al. 1995] distribuído entre as fases de descoberta de recursos e submissão de tarefas no lado da grade móvel ad hoc. Quando um proxy recebe uma mensagem de requisição de descoberta dessa grade (via protocolo DICHOTOMY), ele retorna uma mensagem de resposta indicando ao iniciador a necessidade de autenticação. Essa indicação é acompanhada por um desafio. O iniciador deve retornar ao proxy a resposta ao desafio através da mensagem de requisição de submissão de tarefas do protocolo BoTDP. Essa resposta ao desafio corresponde às credenciais do usuário no proxy.

3.2. Descoberta e Seleção de Recursos

No MoGrid, o protocolo DICHOTOMY é responsável por manter uma base distribuída de informações sobre recursos físicos e lógicos presentes em cada dispositivo móvel. No GTK, informações sobre esses recursos são centralizadas no MDS.

A solução proposta neste trabalho é, conceitualmente, bastante simples: o proxy traduz uma mensagem DICHOTOMY de requisição em uma consulta ao MDS; da mesma forma, a resposta desse serviço é traduzida e repassada ao iniciador na grade móvel como uma mensagem DICHOTOMY de resposta. Porém, na grade fixa existe a possibilidade de um grande número de máquinas serem utilizadas como provedoras de recursos. O envio por parte do proxy de uma mensagem DICHOTOMY de resposta em separado para cada máquina disponível na grade fixa pode, portanto, ser extremamente ineficiente do ponto de vista dos recursos de comunicação utilizados na rede sem fio ad hoc. Por isso, foi feita uma alteração no protocolo DICHOTOMY para que este seja capaz de transportar, em uma única mensagem de resposta, informações contextuais consolidadas sobre múltiplas máquinas na grade fixa. Essas informações são vistas pelo iniciador como sendo providas por um mesmo colaborador, de modo que as requisições subseqüentes de submissão de tarefas nas máquinas da grade fixa, via protocolo BoTP, sejam endereçadas ao proxy, e não diretamente a essas máquinas, conforme descrito na seção que se segue.

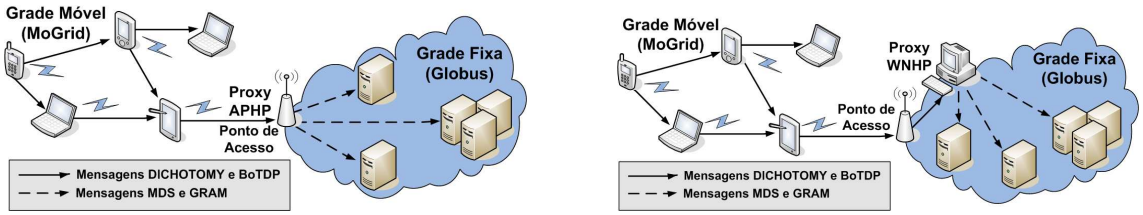
3.3. Submissão de Tarefas

No MoGrid as tarefas são descritas como objetos Java, sendo estes serializados e enviados através do protocolo BoTDP aos colaboradores selecionados durante a fase de descoberta. Esses objetos encapsulam informações sobre executáveis, argumentos e arquivos de entrada e saída. Da mesma forma, objetos Java são usados pelos colaboradores para devolver ao iniciador os resultados da execução das suas tarefas. O GTK, por sua vez, utiliza uma linguagem própria para descrição de tarefas, chamada RSL (*Resource Specification Language*), cujo escopo é mais abrangente que os objetos Java usados no MoGrid. Descrições em RSL (bem como eventuais arquivos necessários à execução das tarefas) são enviadas às máquinas da grade através do serviço GRAM.

Na solução proposta neste trabalho, ao receber do iniciador uma mensagem BoTDP de requisição de submissão, o proxy extrai as informações do objeto Java encapsulado na requisição e constrói com base nas mesmas uma RSL que é submetida, através do serviço GRAM, para as máquinas selecionadas na grade fixa. Durante o processo de submissão pelo proxy, são criadas nessas máquinas também (p.ex. via GridFTP) as instâncias de todos os arquivos necessários para a execução das tarefas descritas na RSL. Por fim, o proxy recebe o retorno das tarefas executadas na grade fixa e o converte em objetos Java que serão retornados para o iniciador via protocolo BoTDP. Vale destacar que as respostas a requisições na fase de descoberta são suficientemente rápidas a ponto de desconexões temporárias não afetarem, em condições de baixa mobilidade (até 2 m/s), a eficácia dessa fase, conforme demonstrado em [Lima 2007]. Já na fase de submissão, as respostas a requisições demoram o tempo necessário para a execução das tarefas correspondentes, tempo este que pode ser longo o bastante para desconexões temporárias ocorrerem. Nesse sentido, por centralizar a interoperação, o proxy pode ser implementado de forma a tratar essas desconexões (vide considerações na Seção 7).

4. Arquitetura do Proxy

Há duas abordagens principais para o estabelecimento de uma estrutura de interoperação baseada em proxy entre grades móveis ad hoc e grades fixas. Na primeira abordagem, o proxy fica hospedado em um ponto de acesso sem fio especificamente configurado para essa interoperação (*Access Point-Hosted Proxy* – APHP). Na segunda abordagem, o proxy fica hospedado em uma máquina convencional na rede fixa (*Wired Network-Hosted Proxy* – WNHP). A Figura 2 contrasta as duas abordagens propostas, que são descritas mais detalhadamente nas subseções que se seguem. São apresentadas também as implicações no uso de uma abordagem ou de outra em termos do grau de transparência oferecido para a grade móvel, da eficiência no uso dos recursos dessa grade (em particular, recursos de comunicação) e da facilidade de implantação do proxy na infra-estrutura de rede fixa.



(a) proxy hospedado no ponto de acesso (APHP).

(b) proxy hospedado na rede fixa (WNHP).

Figura 2. Alternativas de arquitetura do proxy.

4.1. Proxy Hospedado no Ponto de Acesso (APHP)

Na abordagem APHP (Figura 2(a)), uma máquina atua como um ponto de acesso sem fio à rede fixa especificamente configurado para a interoperação entre a grade móvel ad hoc e a grade fixa. Essa máquina deve ser equipada com duas interfaces de rede – uma para a rede sem fio e outra para a rede cabeada – e hospedar a funcionalidade de proxy.

Em um cenário típico, quando um dispositivo móvel entra no raio de alcance de um proxy APHP, este último pode passar a responder a requisições de descoberta difundidas pela grade móvel ad hoc, como se fosse também um dispositivo móvel. Assim, é provida uma maior transparência aos dispositivos da grade móvel ad hoc em relação à interoperação com uma grade fixa. Além disso, há uma economia maior de recursos de comunicação (e, conseqüentemente, de energia) nos dispositivos móveis, pois não é necessária uma fase prévia de sinalização entre esses dispositivos e o proxy. Em contrapartida, a necessidade de implantação de um ponto de acesso com funcionalidade específica pode limitar geograficamente os pontos em que as duas grades podem interoperar.

4.2. Proxy Hospedado na Rede Fixa (WNHP)

Na abordagem WNHP (Figura 2(b)), uma máquina hospeda a funcionalidade de proxy sem a necessidade de uma interface física com a rede sem fio. Essa abordagem tem como principal vantagem o fato de não demandar a implantação de um ponto de acesso com funcionalidade específica, de modo que qualquer ponto de acesso em potencial pode servir como ponte para a interoperação das duas grades. Contudo, como os proxies WNHP estão presentes na rede fixa, sem ligação com a grade móvel ad hoc, é necessário um mecanismo adicional (descrito abaixo) que permita a qualquer dispositivo da grade móvel ad hoc detectar automaticamente a possibilidade de comunicação com esses proxies. Para isso,

é necessário que esse dispositivo conheça a localização dos proxies das grades fixas às quais ele tem direito de acesso. Isso pode ser conseguido, por exemplo, através de uma lista de endereços armazenados no dispositivo, um para cada proxy, ou de um endereço de multicast específico associado a todos os proxies. Em ambos os casos, o mecanismo de detecção de proxies WNHP torna menos transparente para os dispositivos na grade móvel ad hoc a interoperação com grades fixas.

Em um cenário típico, quando um dispositivo móvel detecta a presença de um novo equipamento sem fio em seu raio de alcance, o dispositivo envia sondas (*probes*) a esse equipamento com o intuito de verificar se este último tem acesso a um ou mais proxies. Essas sondas podem ser implementadas com o próprio protocolo DICHOTOMY, por meio de mensagens de requisição de descoberta enviadas não por difusão (como descrito na Seção 2.1), mas diretamente para o equipamento recém-detectado, tendo como endereço de destino os proxies WNHP. Se o equipamento recém-detectado tratar-se de um ponto de acesso sem fio à rede fixa, este tentará encaminhar as mensagens de requisição até esses proxies e, se o encaminhamento for possível, esses proxies responderão a essa requisição. Se receber as respostas dos proxies, o dispositivo móvel pode então passar a atuar como representante (*surrogate*) desses proxies na grade móvel, encaminhando aos mesmos todas as mensagens de requisição de descoberta subseqüentes, bem como repassando à grade móvel as respostas provenientes desses proxies. É importante notar que diferentes dispositivos móveis podem atuar simultaneamente como *surrogates* de um proxy – nesse caso, o mecanismo de balanceamento de carga implementado no protocolo DICHOTOMY evita que um *surrogate* específico seja sobrecarregado com requisições.

Em decorrência do mecanismo de detecção de proxies WNHP descrito acima, há um uso menos eficiente dos recursos de comunicação/energia presentes na grade móvel ad hoc. Isso porque a detecção de um novo equipamento na vizinhança de um dispositivo móvel implica na transmissão, por parte deste último, de nova mensagens de requisição, com o intuito de sondar o novo equipamento e assim determinar se pode atuar ou não como *surrogate* desses proxies.

5. Implementação

Para avaliar a exeqüibilidade das soluções propostas, foi implementada uma versão de proxy para cada uma das abordagens descritas na Seção 4. Essas implementações foram testadas (Seção 5.3) no contexto do projeto Integridade,[3] que mantém uma grade fixa baseada na versão 2.4 do GTK. A API Java CoG Kit[4] foi utilizada no desenvolvimento das funcionalidades de acesso aos serviços da grade fixa. As subseções seguintes apresentam detalhes dessas implementações e do ambiente de teste utilizado.

5.1. Autenticação

O mecanismo de autenticação implementado é baseado em criptografia simétrica (vide considerações na Seção 7). Quando um proxy recebe uma requisição de descoberta da grade móvel (via DICHOTOMY), ele retorna uma resposta indicando ao iniciador a necessidade de autenticação. Essa indicação é acompanhada na mensagem por um número inteiro longo gerado aleatoriamente pelo proxy (o desafio). O iniciador deve cifrar esse número usando sua chave criptográfica, e retornar ao proxy essa cifra juntamente com

[3] `http://integridade.lncc.br/`
[4] `http://www.cogkit.org/`

a identificação do usuário no proxy (a resposta ao desafio), através de uma mensagem de requisição de submissão de tarefas (via BoTDP). Quando essa mensagem chega ao proxy, este decriptografa a cifra usando sua chave, e o número obtido é comparado com o número gerado anteriormente pelo proxy. Caso os números sejam iguais, o proxy poderá encaminhar a submissão da tarefa para as máquinas correspondentes na grade fixa, usando o certificado digital correspondente ao usuário identificado na resposta ao desafio.

Na implementação atual, os certificados digitais dos usuários da grade móvel ficam armazenados em um *keystore* no próprio proxy. Isso se deve à ausência, na grade do projeto Integridade, de um serviço de gerência de credenciais, como o disponibilizado pelo MyProxy [Basney et al. 2005]. Contudo, a adaptação da implementação para esse tipo de serviço é trivial (a API Java CoGKit, usada neste trabalho, já oferece acesso a servidores MyProxy) e transparente para os dispositivos da grade móvel.

5.2. Descoberta e Seleção de Recursos

Na versão 2.4 do GTK não é fornecido um serviço que permita a descoberta de recursos lógicos – o MDS oferece, no máximo, informação sobre sistema operacional e versão do mesmo. Por isso, foi necessário implementar no mecanismo de descoberta do proxy uma funcionalidade adicional, responsável por localizar recursos lógicos na grade. Essa funcionalidade é implementada por meio de scripts de teste que são submetidos para as máquinas da grade fixa descobertas pelo MDS, de modo similar a tarefas computacionais corriqueiras dessa grade. Esses scripts retornam para o proxy a localização de um determinado software em cada uma dessas máquinas, caso esse software esteja realmente instalado nas mesmas. Essa informação fica armazenada em um cache no proxy, sendo entregue aos iniciadores sob demanda, como parte das mensagens DICHOTOMY de resposta a requisições de descoberta.

5.3. Ambiente de Teste

Para testar os proxies desenvolvidos foi utilizada uma rede experimental. Essa rede foi composta de 4 PCs dotados de interfaces IEEE 802.11. Esses PCs atuavam somente como iniciadores, de modo que somente os proxies respondiam a requisições de descoberta e de submissão de tarefas. Na configuração com proxy APHP, um desses PCs foi configurado como o proxy, sendo dotado de uma interface adicional para a rede fixa que dava acesso à grade do projeto Integridade. Na configuração de proxy WNHP, a rede experimental foi acrescida de um ponto de acesso convencional. Esse ponto de acesso foi ligado à rede fixa, à qual foi ligado também um quinto PC atuando como o proxy.

Ambas as configurações de proxy foram testadas com duas aplicações *bag-of-tasks* desenvolvidas em Java – uma de multiplicação de matrizes e outra de simulação de redes –, usando as APIs fornecidas na implementação do MoGrid. Nessas aplicações, uma tarefa mestre era executada no iniciador, que distribuía tarefas escravas aos colaboradores selecionados – no caso, máquinas da grade fixa. Essas tarefas escravas retornavam seus resultados ao iniciador, que processava os mesmos gerando o resultado final. No caso da aplicação de simulação de redes, o iniciador enviava primeiramente aos colaboradores selecionados um módulo Java que implementava o controle das simulações, um script Tcl (linguagem usada pelo simulador ns-2,[5] que foi o usado nos testes) descrevendo o cenário de simulação, e os parâmetros de configuração da primeira rodada de

[5] `http://www.isi.edu/nsnam/ns`

simulação. Para as rodadas seguintes de um mesmo cenário, o iniciador enviava aos colaboradores somente os novos parâmetros de configuração. Ao final de uma rodada em um colaborador, o iniciador recebia do mesmo os arquivos de *trace* de pacotes gerados ao longo da rodada. Para essa aplicação, foi necessário o desenvolvimento de dois scripts de teste para os proxies, que verificassem nas máquinas da grade fixa a disponibilidade de: (i) uma máquina virtual Java de versão compatível com o módulo Java de controle das simulações; e (ii) uma versão do simulador ns-2 compatível com o script Tcl de descrição dos cenários de simulação. Essa aplicação foi efetivamente utilizada durante a avaliação de desempenho do protocolo DICHOTOMY, apresentada em [Lima et al. 2007b].

6. Trabalhos Relacionados

O uso de dispositivos móveis – enquanto unidades *isoladas* de processamento – em grades computacionais é um tema bastante explorado recentemente na literatura. Uma estratégia bastante comum é a utilização de aplicações Web como proxies entre dispositivos móveis e grades tradicionais. Por exemplo, em [Grabowski e Lewandowski 2005] é apresentado um sistema, desenvolvido no contexto do projeto GridLab,[6] que possibilita a dispositivos móveis que utilizem JME no perfil MIDP 1.0 (*Mobile Information Device Profile*) o acesso a diversos serviços de grade – autenticação, gerência de recursos, submissão e gerência de tarefas – implementados sobre o GTK versão 4. Esse acesso é provido por uma aplicação Web projetada especificamente para atender a requisições provenientes de dispositivos móveis, atua como um proxy que mapeia todos os protocolos de acesso aos serviços de grade para HTTP. Pelo fato de usar o perfil MIDP 1.0, essa abordagem apresenta como vantagem a possibilidade de uso a partir de dispositivos móveis com limitações mais estritas de recursos, como aparelhos celulares. Contudo, o uso desse perfil impede o uso de requisições seguras (p.ex. usando HTTPS).

Outra estratégia freqüentemente adotada – e similar em alguns aspectos à nossa proposta – é o uso de proxies que permitem que dispositivos móveis tenham acesso a uma grade fixa por meio de protocolos desenvolvidos especificamente para esse propósito. Em [Pinto et al. 2007], são propostos mecanismos de escalonamento adaptativo de tarefas no proxy. Esse trabalho inclui o desenvolvimento de uma aplicação JME para dispositivos móveis que permite gerenciar esse escalonamento. Já em [Silva et al. 2007] o foco é no tratamento de desconexões entre os dispositivos móveis que fazem uso dos serviços de grade implementados sobre o middleware Integrade [Goldchleger et al. 2004]. Esse trabalho se baseia em *caching* para que, no caso do dispositivo móvel apresentar conectividade intermitente, este possa receber o retorno da execução de suas tarefas em um momento futuro. Contudo, esse trabalho não contempla o tratamento de questões relacionadas à segurança em grades, como autenticação.

Todas as estratégias supracitadas se baseiam no padrão arquitetural cliente/proxy/servidor. O trabalho apresentado em [Bruneo et al. 2003] propõe uma abordagem diferente, baseada em agentes móveis. Nessa arquitetura, quando um dispositivo móvel se desloca entre duas áreas de cobertura relacionadas a dois pontos de acesso distintos (p.ex. em uma configuração de rede celular), um agente associado a esse dispositivo na rede fixa o acompanha, migrando do ponto de acesso anterior para o atual. Essa configuração é similar, em certos aspectos, à arquitetura de proxy proposta neste trabalho, com a diferença que, em nossa arquitetura, o agente hospedado no ponto de acesso

[6] `http://www.gridlab.org`

é fixo. Contudo, em [Bruneo et al. 2003] é apresentado somente um modelo analítico do problema, para fins de comparação com outros padrões arquiteturais, sem mencionar aspectos relacionados à implementação.

Por fim, vale reiterar que as abordagens discutidas acima focam no *uso* de dispositivos móveis em grades fixas – tipicamente, a partir de redes sem fio infra-estruturadas. Ao contrário da abordagem proposta neste artigo, nenhum desses trabalhos dá suporte explícito à *interoperação* de grades móveis ad hoc com grades fixas.

7. Conclusões

Através de um proxy de acesso a uma grade fixa, um dispositivo móvel se depara com uma quantidade de recursos muito maior e muito mais estável, quando comparada à sua própria, viabilizando a execução de aplicações que exigem maior capacidade computacional. Em contraste com as demais abordagens encontradas na literatura, tipicamente focadas na integração de dispositivos móveis a grades fixas através de redes infra-estruturadas, este artigo propôs o uso de protocolos e mecanismos especificamente desenvolvidos para grades móveis ad hoc no acesso de dispositivos móveis a grades fixas. Vários desafios na interoperação de grades móveis com grades fixas relacionados à segurança, à descoberta e seleção de recursos, e à submissão de tarefas foram levantados e discutidos, e algumas propostas para abordar esses desafios foram apresentadas.

Há alguns pontos deste trabalho que pretende-se explorar futuramente. Um deles é investigar a possibilidade da grade móvel ad hoc executar tarefas para a grade fixa, o que traz implicações quanto à QoS fornecida por esses dispositivos e à confidencialidade dos dados submetidos para processamento nos mesmos. Ligada a esse ponto está a adoção de criptografia simétrica em nossa implementação atual. Essa decisão de implementação deveu-se ao seu menor custo computacional para os dispositivos móveis, mas traz outros problemas relacionados à gerência distribuída de chaves, os quais pretende-se também investigar. Outro ponto é possibilitar a submissão de tarefas com interdependências (*workflows*) a partir dos dispositivos móveis. No contexto das grades móveis, tais tarefas são extremamente complexas de serem geridas por causa do alto dinamismo dessas grades; contudo, no caso de proxies WNHP, um iniciador toma conhecimento da existência dos mesmos, podendo então usá-los para submeter tarefas com interdependências especificamente para os nós da grade fixa. Para tanto, são necessárias extensões ao protocolo BoTDP, como o uso de objetos Java representando grafos de dependência entre tarefas que pudessem ser interpretados pelos proxies. Por fim, pretende-se implementar nesses proxies mecanismos de tolerância a falhas que evitem a perda de resultados de execuções de tarefas devido a desconexões temporárias na grade móvel ad hoc.

Agradecimentos

Agradecemos ao pesquisador Bruno Schulze pelo acesso à grade do projeto Integridade. Este trabalho foi financiado pela FAPERJ, CAPES e CNPq, e pelo MCT.

Referências

Basney, J., Humphrey, M., e Welch, V. (2005). The MyProxy online credential repository. *Software: Practice and Experience*, 35(9):801–816.

BBN Technologies (2007). Press release: BBN technologies awarded 10.8 million in defense funding to design and develop huge, scalable, adaptable, wireless network.

Bruneo, D., Scarpa, M., Zaia, A., e Puliafito, A. (2003). Communication paradigms for mobile grid users. In *Proc. 3rd CCGrid*, Washington, DC, USA. IEEE CompSoc.

Denning, P. J. (2006). Hastily formed networks. *Comm. ACM*, 49(4):15–20.

Foster, I. e Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *Int'l Journal of Supercomputer Appl. and High Performance Computing*, 11(2):115–128.

Goldchleger, A., Kon, F., Goldman, A., Finger, M., e Bezerra, G. C. (2004). Integrade: Object-oriented grid middleware leveraging idle computing power of desktop machines. *Concurrency and Computation: Practice and Experience*, 16(5):449–459.

Gomes, A. T. A., Ziviani, A., Lima, L. S., e Endler, M. (2007). DICHOTOMY: A resource discovery and scheduling protocol for multihop ad hoc mobile grids. In *Proc. 1st WCAMG*, págs. 719–724, Washington, DC, USA. IEEE CompSoc.

Grabowski, P. e Lewandowski, B. (2005). Access from J2ME-enabled mobile devices to grid services. Technical report, The GridLab Project.

Kaufman, C., Perlman, R., e Speciner, M. (1995). *Network security: private communication in a public world*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Lima, L. S. (2007). *Um Protocolo para Descoberta e Seleção de Recursos em Grades Móveis Ad hoc*. Tese de Doutorado, PUC-Rio.

Lima, L. S., Gomes, A. T. A., Ziviani, A., e Endler, M. (2007a). Descoberta de serviços em redes de computadores. In Abelém, A. J. G., Costa, J. C. W. A., Rodriguez, N. R., e Reis, R. Q., eds. *Minicursos XXV SBRC*, Cap. 3, págs. 113–162. UFPA, Belém, PA.

Lima, L. S., Gomes, A. T. A., Ziviani, A., Endler, M., Soares, L. F. G., e Schulze, B. R. (2005). Peer-to-peer resource discovery in mobile grids. In *Proc. 3rd MGC*, Washington, DC, USA. IEEE CompSoc.

Lima, L. S., Gomes, A. T. A., Ziviani, A., França, P. A., Bastos, B. F., e Endler, M. (2007b). Reduzindo a implosão de respostas em protocolos de descoberta de serviços para redes ad hoc de saltos múltiplos. In *Anais XXV SBRC*, Porto Alegre, RS. SBC.

Marinescu, D., Marinescu, G. M., Yongchang, J., Boloni, L., e Siegel, H. J. (2003). Ad hoc grids: communication and computing in a power constrained environment. In *Proc. 6th IPCCC*, págs. 113–122, Washington, DC, USA. IEEE CompSoc.

McKnight, L. W., Howison, J., e Bradner, S. (2004). Guest editorial: Wireless grids—distribute resource sharing by mobile, nomadic, and fixed devices. *IEEE Internet Computing*, 8(4):24–31.

Pinto, A. R., Caetano, M., Dantas, M. A. R., e Bordin, J. L. (2007). Uma abordagem para integração de dispositivos móveis com agregados de computadores. In *Proc. 1st WPUC*, Porto Alegre, RS. SBC.

Sacramento, V., Endler, M., Rubinsztejn, H. K., Lima, L. S., Goncalves, K., Nascimento, F. N., e Bueno, G. A. (2004). MoCA: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10).

Silva, F. J. S., Gomes, D. S., e Endler, M. (2007). Integrando dispositivos móveis ao middleware Integrade. In *Proc. 1st WPUC*, Porto Alegre, RS. SBC.

Agentes Móveis: Uma Abordagem para a Execução de Aplicações Longas em Ambientes Oportunistas

Vinicius Pinheiro[1], Alfredo Goldman[1] e Francisco José da Silva e Silva[2]

[1]Depto. de Ciência da Computação, Universidade de São Paulo, Brasil (USP)

[2]Depto. de Ciência da Computação, Universidade Federal do Maranhão, Brasil

{vinicius,gold}@ime.usp.br, fssilva@deinf.ufma.br

Abstract. *The mobile agent paradigm has emerged as a promising alternative to overcome the construction challenges of opportunistic Grid environments. MAG (Mobile Agents for Grid Computing Environment) explores this powerful paradigm by dynamically loading grid applications into mobile agents. These MAG agents can be dynamically reallocated to Grid nodes through a transparent migration mechanism as a way to provide load balancing and support for non-dedicated nodes. MAG also includes retrying, replication and checkpoint as fault-tolerance techniques. These mechanisms can be applied in a flexible manner, and be combined in order to meet different scenarios of resource availability. In this paper we describe the MAG architecture and what it can do in a volunteer computing environment. We also present a description of these mechanisms and a performance evaluation in a real world environment.*

Resumo. *O paradigma de agentes móveis vem sendo abordado como uma alternativa promissora para superar os desafios impostos na construção de ambientes oportunistas. O* middleware *de grade MAG (Mobile Agents for Grid Computing Environment) explora esse poderoso paradigma através do encapsulamento das aplicações em agentes móveis, que assim são submetidas ao ambiente de execução. Esses agentes podem ser realocados dinamicamente entre os nós da grade através de um mecanismo de migração transparente, adequado ao balanceamento de carga entre nós não dedicados. O MAG também inclui técnicas de tolerância a falhas tais como replicação,* checkpoint *e reenvio de tarefas. Esses mecanismos podem ser utilizados isoladamente ou em conjunto de forma a se adequar a diferentes cenários de disponibilidade de recursos. Este trabalho mostra a arquitetura do MAG e o que esse* middleware *pode fazer em ambientes de grades oportunistas. Inclui também a descrição dos mecanismos de tolerância a falhas e avaliações de desempenho em um ambiente real.*

1. Introdução

As grades computacionais têm atraído bastante atenção das comunidades acadêmica e industrial. Esses ambientes são alternativas atraentes para a execução de aplicações paralelas ou distribuídas que demandam alto poder computacional, tais como mineração de dados, previsão do tempo, biologia computacional, física de partículas, processamento de imagens médicas, entre outras. O funcionamento de uma grade computacional é determinado pelo seu *middleware*. O *middleware* de grade é responsável por esconder toda a complexidade relacionada à distribuição e a heterogeneidade dos recursos, fornecendo uma

visão transparente dos serviços oferecidos aos usuários. Essa tarefa não é trivial, já que envolve o gerenciamento e alocação de recursos distribuídos, escalonamento dinâmico de tarefas, tolerância a falhas, suporte a alta escalabilidade e grande heterogeneidade dos componentes de software e hardware, e conformidade com requisitos de segurança. A tecnologia de agentes móveis se mostra adequada para lidar com esses desafios devido às suas caraterísticas intrínsecas tais como:

1. *Cooperação*: Agentes possuem a capacidade de interagir e cooperar com outros agentes. Isto pode ser explorado para o desenvolvimento de complexos mecanismos de comunicação entre os nós da grade;
2. *Autonomia*: Agentes são entidades autônomas, de forma que a sua execução flui sem ou com pouca intervenção do cliente que a iniciou. Esse modelo é adequado para a submissão e execução de aplicações de grade;
3. *Heterogeneidade*: Diversas plataformas de agentes móveis foram projetadas para ambientes heterogêneos. Esta característica propicia uma integração mais transparente dos recursos computacionais dispersos na infra-estrutura multi-institucional da grade;
4. *Reatividade*: Agentes podem reagir a eventos externos (e.g. variações na disponibilidade dos recursos);
5. *Mobilidade*: Agentes móveis podem migrar de uma máquina para outra, carregando consigo o seu estado de execução atual. Esse mecanismo pode ser utilizado para prover balanceamento de carga entre os nós da grade;
6. *Proteção e Segurança*: Diversas plataformas de agentes oferecem mecanismos de proteção e segurança, como autenticação, criptografia e controle de acesso.

Desde 2004, nosso grupo de pesquisa tem trabalhado na tarefa de aplicar o paradigma de agentes móveis para o desenvolvimento de *middlewares* de grade [Barbosa and Goldman 2004, Lopes et al. 2005]. Esses *middlewares* seguem um modelo oportunista, no qual o poder computacional ocioso das estações de trabalho é utilizado para executar aplicações paralelas de computação intensiva.

Este trabalho descreve melhorias realizadas no *middleware* MAG para dar suporte ao alto dinamismo dos ambientes oportunistas, provendo um gerenciamento efetivo de aplicações seqüenciais longas e alocação de recursos necessários para sua execução. Na próxima seção são apresentados alguns trabalhos relacionados. Em seguida, na seção 3, é apresentada a arquitetura do MAG e do Integrade. Na seção 4, são apresentadas as mudanças necessárias para prover mecanismos eficientes de tolerância a falhas. São fornecidos alguns resultados experimentais na seção 5 e, na última seção, são apresentadas as conclusões e os trabalhos futuros.

2. Trabalhos Correlatos

Existem diversos trabalhos relacionados ao apresentado neste artigo. O trabalho mais conhecido na área é o projeto SETI (Search for Extra Terrestrial Inteligence - `http://setiathome.ssl.berkeley.edu`), relacionado à pesquisa por vida extraterrestre. Este projeto foi implementado com ênfase em aspectos de segurança e na confiabilidade dos resultados. Mais recentemente, o projeto BOINC (`http://boinc.berkeley.edu/`) propôs uma infra-estrutura que permite a execução de diferentes programas, os quais podem ser carregados em máquinas de voluntários espalhadas pela internet. Existem projetos similares que compartilham algoritmos fixos como Mersenne

(http://www.mersenne.org), no qual diferentes algoritmos ou desafios podem ser programados (http://www.distributed.net). Contudo, nesses projetos, o suporte para aplicações seqüenciais longas se restringe à realização de *checkpoints* locais (com algumas exceções como climate - http://www.climateprediction.net), ou ao uso de replicação para garantir o progresso de aplicações invidivuais. Outra solução que dá suporte às aplicações do tipo paramétricas é fornecida pelo projeto OurGrid [Cirne et al. 2006], contudo o objetivo principal deste projeto é lidar com a infraestrutura do *middleware* e não com aplicações individuais seqüenciais.

Diversos trabalhos utilizam técnicas de *checkpoint* para garantir o progresso de aplicações seqüenciais longas. Um desses trabalhos, descrito em [Hwang and Kesselman 2003], está diretamente relacionado com a nossa pesquisa. Nesse trabalho, os autores estudaram diversas abordagens para lidar com falhas nas máquinas da rede: reenvio, *checkpoint*, replicação, e replicação com *checkpoint*. Eles concluíram que em ambientes de grade com altos períodos de indisponibilidade (e.g. ambientes oportunistas), a replicação com *checkpoint* se sobressai sobre as outras abordagens, usando como métrica de comparação o menor tempo de execução.

No contexto dos agentes móveis, vários trabalhos sobressaem-se. Alguns utilizam uma abordagem oportunista [Fukuda et al. 2003], mas a maior parte deles apresenta características mais relacionadas ao *middleware* do que às aplicações [Cao et al. 2001, Cao et al. 2002, Loke 2003, Martino and Rana 2004]. O projeto UWAgents [Fukuda and Smith 2006] se concentra no desenvolvimento de uma nova infra-estrutura para a execução de agentes móveis. Apesar de oferecer os serviços de *checkpointing* e migração, essa plataforma só permite a retomada de execução a partir do início de uma função especificada diretamente no código. Além disso, recursos como arquivos de entrada e saída e linhas de execução internas não são migrados. Outros projetos, como Anthill [Bagaoglu et al. 2002] e Organic Grid [Chakravarti et al. 2005], se inspiram em abordagens sociais e biológicas para a implementação de redes ponto a ponto. Contudo, são *middlewares* de propósitos gerais e não possuem uma política de tolerância a falhas definida ou voltada para fins específicos.

Alguns dos trabalhos sobre agentes móveis foram realizados dentro do nosso projeto Integrade [Goldchleger et al. 2004]. As primeiras abordagens na utilização de agentes móveis em grades oportunistas são vistas em [Barbosa and Goldman 2004] onde uma arquitetura baseada em Aglets (http://www.trl.ibm.com/aglets/) é inicialmente apresentada, e depois avaliada com o uso de diversas réplicas em [Barbosa et al. 2005]. Mais recentemente, um trabalho baseado na plataforma de agentes JADE (http://www.jade.tilab.com) foi realizado [Lopes et al. 2005, Lopes and da Silva 2006]. Nesse trabalho, os agentes móveis emprestam seus serviços para a implementação de um mecanismo transparente de tolerância a falhas baseado em *checkpoints*. Isso é realizado através da técnica de instrumentação dos binários das aplicações. Nosso trabalho dá continuidade aos esforços desse último e, até o momento, pelo que pudemos observar, é o primeiro que especificamente utiliza agentes móveis em conjunto com técnicas de replicação e *checkpointing*, em um *middleware* de grade, para dar suporte à execução de aplicações seqüenciais longas em ambientes oportunistas. Além disso, realizamos experimentos em um ambiente real com o propósito de validar a eficácia desses mecanismos.

3. Arquitetura

O projeto Integrade envolve o desenvolvimento de um *middleware* de grade que aproveita o poder computacional ocioso das estações de trabalho. Este projeto é mantido pelo Instituto de Matemática e Estatística da Universidade São Paulo, em conjunto com outras instituições. O *middleware* Integrade é baseado em CORBA, um padrão para sistemas de objetos distribuídos. Os serviços do Integrade (i.e. nomeação, transação, persistência) são exportados como interfaces CORBA IDL sendo acessíveis por uma grande variedade de linguagens de programação e sistemas operacionais.

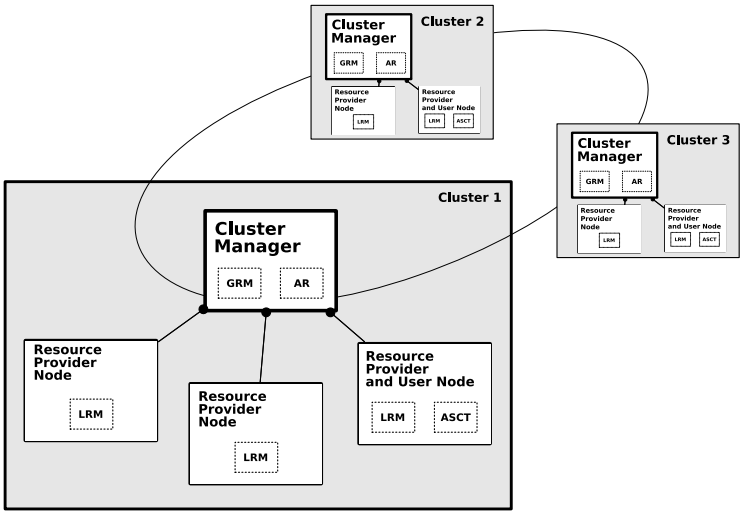


Figura 1. Arquitetura do Integrade

A arquitetura do Integrade segue uma hierarquia na qual cada nó pode assumir diferentes papéis. O *Cluster Manager* é o nó responsável por gerenciar o aglomerado e realizar a comunicação com outros aglomerados. Um nó do tipo *Resource Provider* exporta parte dos seus recursos, deixando-os disponíveis para os usuários da grade. Um nó do tipo *User Node* é aquele que pertence a um usuário da grade que submete aplicações ao ambiente de execução. Como pode ser visto na figura 1, a arquitetura do Integrade segue uma hierarquia de duas camadas no nível interno ao aglomerado, e, no nível externo, a comunicação é feita através de uma rede ponto a ponto que interliga os aglomerados.

O projeto MAG foi desenvolvido pelo Departamento de Ciência da Computação da Universidade Federal do Maranhão e introduz a tecnologia de agentes móveis como uma nova forma de executar aplicações seqüenciais e paramétricas no Integrade [Lopes and da Silva 2006]. Através do MAG, o usuário da grade pode submeter aplicações Java, o que não é permitido pelo *middleware* nativo do Integrade. Isso é realizado através do encapsulamento dessas aplicações dentro de agentes móveis. O MAG utiliza a plataforma de agentes JADE (*Java Agent Development Framework*) para prover serviços de agentes como comunicação e monitoramento do ciclo de vida. JADE provê uma fila privada de mensagens para cada um dos agentes, permitindo que eles possam trocar mensagens especificando os tópicos e seus destinatários. JADE é portátil, visto que é implementado em Java, e está de acordo com a especificação FIPA (Foundation for Intelligent Physical Agents - `http://www.fipa.org`).

O desenvolvimento do MAG reaproveitou diversos componentes do Integrade,

evitando-se o esforço desnecessário de reimplementá-los. São eles: o *Global Resource Manager (GRM)*, o *Local Resource Manager (LRM)*, o *Application Repository (AR)* e o *Application Submission and Control Tool (ASCT)*. O *GRM* é o componente principal da grade e é executado em nós do tipo Cluster Manager. Esse mantém uma lista sobre os *LRM*s ativos e pode escalonar aplicações entre eles. O *LRM* é executado em cada nó do tipo *Resource Provider* e carrega todo o ambiente necessário para execução das aplicações. O *AR* provê um repositório centralizado para o armazenamento dos binários das aplicações submetidos à grade. Por fim, o *ASCT* é executado nos nós dos usuários (tipo *User Node*) e fornece uma interface pela qual o usuário pode submeter aplicações à grade. O usuário pode monitorar a execução e visualizar os resultados finais. Além desses, em breve, será incorporado o componente *LUPA (Local Usage Pattern Analyzer)* que será executado junto ao LRM para coletar informações sobre utilização de memória, CPU e disco. A arquitetura do MAG incorpora ao Integrade outros componentes que adicionam funcionalidades de agentes móveis e mecanismos de tolerância a falhas:

1. *ExecutionManagementAgent (EMA)*: Esse componente armazena informações sobre as execuções atuais e passadas, como o estado atual de execução (*accepted, running, finished*), parâmetros de entrada e máquinas utilizadas. Essas informações podem ser consultadas posteriormente para restaurar a execução das aplicações a partir do ponto em que elas se encontravam antes da falha;
2. *AgentHandler*: Esse componente é executado em cada um dos *LRM*s. O *AgentHandler* funciona como um proxy para a plataforma de agentes JADE, instanciando os *MAGAgent*s para cada execução pedida e hospendado-os;
3. *ClusterReplicationManagerAgent (CRM) e ExecutionReplicationManagerAgent (ERM)*: Quando um *GRM* recebe uma requisição de execução com réplicas ele a delega para o *CRM*. Esse componente processa informações para cada réplica e cria um agente *ERM* para lidar com a requisição. O *ERM* faz contato com os *LRM*s das máquinas escolhidas pelo escalonador do *GRM*, com o objetivo de executar as réplicas, uma em cada máquina
4. *StableStorage*: É o componente que recebe o *checkpoint* em formato compactado, armazena-o no sistema de arquivos, e o recupera assim que recebe um pedido para tal. Esse agente é executado nos nós do tipo Cluster Manager;
5. *MAGAgent*: É o principal componente do *middleware* MAG. O *MAGAgent* encapsula e instancia a aplicação, além de tratar as exceções que podem ser lançadas;
6. *AgentRecover*: Esse componente é criado sob demanda para recuperar a execução de um agente na ocorrência de falhas.

Todos esses componentes acima descritos são implementados como agentes e podem ser monitorados através de uma interface gráfica nativa da plataforma JADE. Nessa, é possível visualizar o *Main-Container* e os agentes que ele hospeda: *EMA*, *StableStorage*, *CRM* e *ERM*, além de outros agentes que fazem parte da infra-estrutura da plataforma JADE. O *Main-Container* é executado junto ao *GRM*. Por essa interface podemos também visualizar as tarefas que estão sendo executadas em cada *AgentHandler* e sua máquina.

4. Tolerância a falhas no MAG

Nesta seção, serão apresentados os mecanismos de tolerância a falhas que o MAG dispõe. Esses mecanismos podem ser combinados para se adequar a diferentes cenários

de execução que surgem quando da variação na disponibilidade dos recursos, resultando em 4 diferentes estratégias: reenvio (a aplicação é submetida novamente em caso de falha); replicação (várias réplicas da aplicação são submetidas para execução ao mesmo tempo); *checkpointing* (a aplicação salva o seu estado de execução periodicamente no *StableStorage*) e *checkpointing* com replicação (cada réplica salva o seu estado de execução periodicamente em um repositório estável). Na presença de falhas, o reenvio e a retomada de execução a partir do último *checkpoint* são aplicados para cada réplica.

O MAG permite a submissão de aplicações Java. Para executá-las, é necessário fazer uma extensão da classe `MagApplication`. Isso é preciso para que, no momento da execução, o código da aplicação seja encapsulado no agente móvel e esteja apto para executar na plataforma de agentes. Segue uma descrição do que ocorre quando é feito uma submissão com réplicas no MAG (vide figura 2):

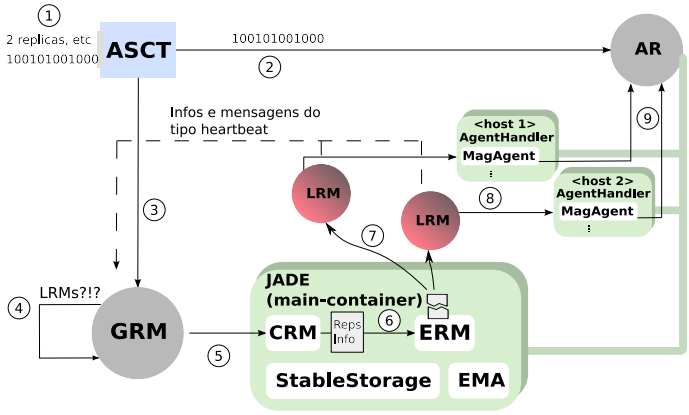


Figura 2. Submissão de aplicações no MAG

O usuário submete a aplicação através da interface do *ASCT*, junto com informações sobre a execução (1): parâmetros de entrada, número de réplicas, arquivos de entrada e saída. O binário da aplicação é armazenado no *AR* (2). O pedido de execução é enviado ao GRM (3). Após a submissão, o *GRM* verifica se existem recursos suficientes (e.g. número de réplicas não pode exceder o número de nós *LRM*)(4). Caso positivo, o *GRM* delega a execução para o *CRM* (5). Esse componente gera as informações das réplicas (com um *ID* único para cada réplica) e cria um agente *ERM* para gerenciar a requisição (6). O *ERM* prossegue com a execução repassando para cada *LRM* escolhido as informações de execução de uma das réplicas (7). Então, cada *LRM* delega a execução para o *AgentHandler*, que por sua vez cria um *MAGAgent* para encapsular a aplicação (8). O *MAGAgent* é responsável por fazer o download do binário junto ao *AR* (9), instanciar a aplicação, e notificar o *AgentHandler* quando a execução estiver terminada. Todas as informações sobre execução (e.g. tempo de execução, número de réplicas, máquinas que foram usadas, etc) são colocadas em um banco de dados relacional pelo *EMA* e podem ser consultadas posteriormente.

Em um ambiente oportunista, muitos problemas podem surgir enquanto a aplicação está executando (e.g. máquinas sendo desligadas, perda de mensagens, falhas de memória, particionamento da rede, etc). Essa situação se torna ainda mais crítica quando consideramos a execução de aplicações longas, já que elas ficam expostas a esses problemas durante um longo período de tempo. Existem diversas formas de se classificar

as falhas que podem ocorrer em sistemas distribuídos. Em nosso trabalho, adotamos a classificação proposta por Veríssimo em [Veríssimo and Rodrigues 2001] que estabelece os seguintes tipos: falhas de colapso, omissão, temporização, sintática e semântica. Atualmente, o nosso *middleware* detecta somente falhas de colapso nos nós. Apesar disso, é válido ressaltar que os mecanismos de reenvio e de replicação de tarefas procuram amenizar o prejuízo causado pelas falhas não detectadas. Em nossa grade, as falhas de colapso ocorrem geralmente quando a aplicação é encerrada de forma abrupta e inesperada. Quanto isso acontece, uma *RuntimeException* é lançada e o fluxo de execução é redirecionado para o método *uncaughtException* da classe `MagAgent` (que é uma sobrecarga do método presente na classe `ThreadGroup`). Esse método instancia um *AgentRecover* local que recebe informações do *GRM* contendo uma referência para o *AgentHandler* de um outro nó disponível. Finalmente, o *AgentRecover* solicita a esse *AgentHandler* remoto que a execução seja retomada.

No MAG, o *checkpoint* é alcançado através da instrumentação do binário da aplicação, não sendo necessário qualquer intervenção do programador no código-fonte. Este processo é realizado através do arcabouço *MAG/Brakes*. Esse arcabouço é uma versão modificada do arcabouço Brakes [Truyen et al. 2000], e foi desenvolvida no Laboratório de Sistemas Distribuídos da Universidade Federal do Maranhão. O *MAG/Brakes* realiza a captura do estado de execução de *threads* Java, possibilitando a retomada da execução em uma outra máquina. Essa captura é realizada automaticamente e pode ocorrer sempre após a invocação de um método da aplicação, com a condição de que um tempo mínimo tenha se passado desde o último *checkpoint* realizado. Atualmente, esse intervalo entre os *checkpoints* fica definido no código da aplicação. O *MAG/Brakes* também é o núcleo de um poderoso mecanismo de migração, já que a execução pode ser interrompida a qualquer momento e, posteriormente, ser retomada sem perda da computação já realizada. Atualmente, o *MAG/Brakes* realiza apenas instrumentação de binários Java compilados para a versão 1.4 (ou anteriores) da máquina virtual.

Quando uma aplicação instrumentada é executada no MAG, o método *setCompressedCheckpoint* da classe `MagAgent` é periodicamente invocado. Através desse método, o agente interage com o componente *StableStorage* que, assim, fica responsável por recolher todo o contexto de execução devidamente compactado e armazená-lo em um arquivo local. Quando a execução da aplicação precisa ser restaurada (i.e. alguma falha ocorreu), o método *getCompressedCheckpoint* é invocado para preencher o contexto da aplicação com o *checkpoint* recuperado. Após isso, a execução da aplicação é retomada.

5. Avaliação de Desempenho

Diferente da maioria dos sistemas distribuídos, o MAG oferece múltiplas técnicas de tolerância a falhas. Nesta seção, será apresentada uma avaliação de desempenho realizada em um ambiente do mundo real, demonstrando assim a necessidade de se fornecer suporte a múltiplos mecanismos de tolerância a falhas para a execução de aplicações longas em ambientes oportunistas. Todos os testes foram realizados em 6 máquinas do LCPD (Laboratório de Computação Paralela e Distribuída) do nosso instituto. Essas máquinas estão conectadas por uma rede Ethernet local de 100Mbps. Durante os testes, as máquinas permaneciam ligadas e eram utilizadas por estudantes. Contudo, os testes foram realizados em períodos de pouca atividade, já que o mais importante era provocar impacto no desempenho através das falhas geradas artificialmente. Seguem as configurações:

Máquina	Processador	Memória	Swap	OS	Versão
bauru	AMD 2.0 GHz	1 GB	-	Linux i686	2.6.20.6
ilhabela	AMD 2.0 GHz	1 GB	1.5 GB	Linux i686	2.6.22.14-generic
taubate	AMD 2.0 GHz	3 GB	768 MB	Linux x86_64	2.6.22.14-generic
giga	Intel 3.0 GHz	2 GB	2 GB	Linux i686	2.6.22.14-generic
orlandia	AMD 2.0 GHz	1 GB	640 MB	Linux i686	2.6.22.14-generic
motuca	AMD 2.2 GHz	1.5 GB	2 GB	Linux x86_64	2.6.10

5.1. Metodologia

Para propósito de avaliação, nós usamos uma aplicação Java que executa o mesmo método em um loop de 200000 iterações. Esse método realiza a concatenação de cadeias de caracteres, incrementando sucessivamente o valor da cadeia obtida na chamada anterior. Pelo o que foi visto na seção 4 sobre o mecanismo de *checkpointing*, essa aplicação, por fazer tantas chamadas a métodos e aliada a um intervalo aproximado de 5 segundos entre os *checkpoints*, é um exemplo de aplicação com um alto sobrecusto. Além disso, essa aplicação faz uso intenso de memória, explorando, assim, a característica mais discrepante entre as máquinas utilizadas. Isso faz com que o tempo de execução da aplicação sofra uma grande variação, dependendo da máquina. Para obter uma estimativa do tempo de execução livre de falhas, executamos essa aplicação uma vez em cada máquina, sem o uso de *checkpoints*. Os resultados podem ser vistos na tabela a seguir:

Máquina	Tempo de Execução	Máquina	Tempo de Execução
bauru	8 minutos e 28 segundos	ilhabela	8 minutos e 18 segundos
taubate	2 minutos e 17 segundos	giga	5 minutos e 34 segundos
orlandia	8 minutos e 51 segundos	motuca	3 minutos e 11 segundos

Esses resultados foram obtidos a partir da execução da aplicação em um ambiente ideal, sem falhas, e fora do ambiente do MAG (i.e. apenas a máquina virtual Java foi utilizada). Esses valores servem, portanto, como uma medida de referência aos resultados dos testes. Nossa avaliação foi realizada variando-se dois parâmetros: tempo médio entre falhas (TMEF) e número de réplicas. O primeiro é a média do tempo que leva para uma falha ocorrer no ambiente de execução e o segundo é o número de réplicas de uma aplicação submetida para execução. Nós usamos 0, 1, 3, e 5 réplicas durante nossa avaliação, e para cada número de réplicas nós fixamos o TMEF para 0,5 (30 segundos) e 1,0 (1 minuto). Esses valores de TMEF podem ser considerados severos de acordo com os resultados da simulações feitas em [Sallem et al. 2007]. Esses valores foram escolhidos, portanto, para medir efetivamente a eficácia dos mecanismos de tolerância a falhas.

O *ERM* foi modificado para gerar as falhas de acordo com o valor de TMEF. Ele invoca periodicamente uma função que devolve *verdadeiro* ou *falso*. Se *verdadeiro*, a falha é gerada e o nó entra em colapso, caso contrário, nada ocorre. Essa função é invocada a cada 10 segundos de maneira que, a cada vez que isso é feito, a probabilidade de que o valor devolvido seja *verdadeiro* é de 1/3 para TMEF = 0,5 e de 1/6 para TMEF = 1,0. A cada erro gerado pelo *ERM*, um *AgentHandler* é escolhido randomicamente, e todos os seus agentes são abruptamente encerrados. A partir daí, o mecanismo de tolerância a falhas entra em ação de acordo com o exposto na seção 4. Devido a restrições de tempo no acesso às máquinas, adotamos a seguinte convenção temporal: 1 minuto equivale a 1 hora para todos os valores adotados em nossos experimentos. Se fosse considerado o

tempo real, os experimentos levariam meses para serem executados. Dessa forma, foi possível realizar mais execuções em menos tempo para obter resultados estatisticamente relevantes.

5.2. Resultados

Baseado nesses valores, nós utilizamos duas estratégias para medir o tempo total de execução da aplicação: Replicação e Replicação com *Checkpointing*. Para cada número de réplicas, nós realizamos 30 execuções e medimos a média aritmética e o desvio padrão dos tempos de execução. Quando usamos réplicas, nós assumimos que o tempo de execução a ser considerado é o tempo de execução da réplica que encerrou primeiro. Os resultados podem ser vistos nas figuras 3 e 4.

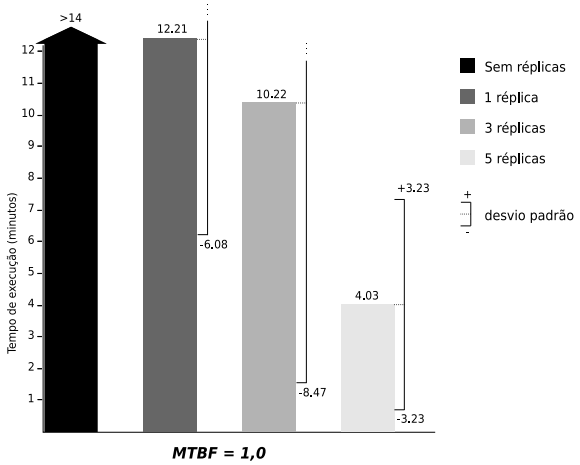


Figura 3. Replicação

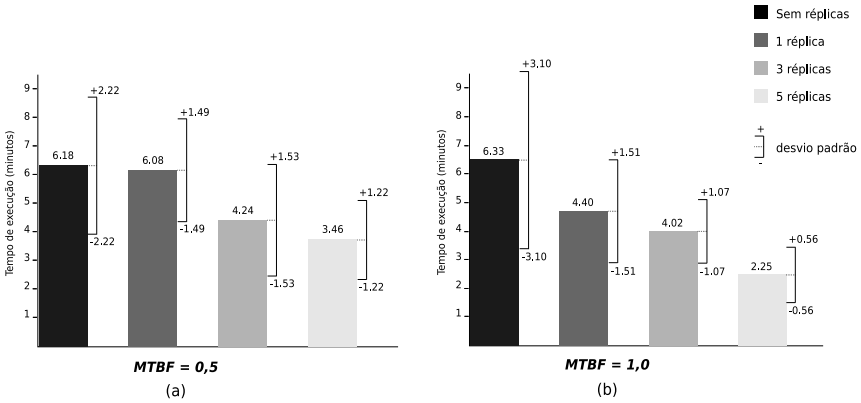


Figura 4. Replicação com Checkpointing

Analisando os resultados da figura 4(a), nós podemos perceber um considerável ganho no tempo total de execução quando o número de réplicas é aumentado. Por exemplo, nós podemos ver uma clara vantagem em usar 3 réplicas ao invés de somente uma, dado que o tempo médio de execução cai de 6 minutos e 8 segundos para 4 minutos e 24 segundos. Podemos ainda perceber que, quando utilizamos 5 réplicas, o tempo total de execução (3 minutos e 46 segundos) decai para quase a metade do tempo de execução sem

réplicas (6 minutos e 18 segundos). Em 4(b), com o TMEF igual a 1 minuto, essa vantagem na utilização de 5 réplicas é ainda maior, com o tempo médio de execução caindo de 6 minutos e 33 segundos para 2 minutos e 25 segundos. Sem o uso de *checkpoint* (figura 3), percebe-se uma redução de 61% no tempo de execução entre os experimentos com 3 e 5 réplicas.

Em nossos experimentos, os ganhos com a utilização de *checkpoints* foram evidentes: mesmo quando 5 réplicas são utilizadas os experimentos com replicação apresentaram um tempo de execução 44% mais alto (4 minutos e 3 segundos) do que os experimentos com replicação e *checkpointing* (2 minutos e 25 segundos), para TMEF igual a 1 minuto. Essa diferença ocorre por dois motivos: (1) os valores de TMEF são baixos e, assim, as réplicas são constantemente interrompidas. Sem *checkpointing*, as réplicas precisam reiniciar sua execução desde o início; (2) como a aplicação não processa uma grande quantidade de dados, os *checkpoints* são pequenos e todo o processo de recuperação e migração não consome mais do que 2 segundos. Mas, a despeito dessas vantagens, experimentos com outro tipo de aplicação foram realizados em [Lopes 2006] e revelaram uma sobrecarga de 8,93% em relação ao tempo normal de execução. Vale observar, portanto, que a sobrecarga depende do tipo de aplicação e, em ambientes mais estáveis (i.e. sem falhas ou com TMEF maiores), o uso de *checkpoints* pode ser questionado ou otimizado, mas isso será validado em futuros experimentos.

Fazendo-se uma análise comparativa entre 4(a) e 4(b), nota-se uma clara discrepância quando somente uma réplica é utilizada. Quando o TMEF é igual a 0,5, não há praticamente diferença alguma entre utilizar somente uma réplica ou nenhuma réplica. Mas, ao dobrar o valor do TMEF, já percebe-se uma clara vantagem obtida com o uso de uma réplica. Nesse exemplo, fica evidente que, para valores distintos de TMEF, números diferentes de réplicas devem ser utilizados se o objetivo for manter o tempo total de execução abaixo de um valor desejado.

Os experimentos apresentam valores de desvio padrão proporcionalmente altos. Na figura 4(b), a proporção entre desvio padrão e tempo médio de execução sem réplicas é de 49%. Nos outros tempos médios de execução extraídos, essa proporção varia entre 25% (4(b) para 5 réplicas) e 36% (4(a) para execução sem réplicas). Quando somente a replicação é utilizada (figura 3), a proporção aumenta mais ainda, chegando a 83% nos experimentos com 3 réplicas. Isso se deve a 3 fatores: número de réplicas, heterogeneidade das máquinas utilizadas nos experimentos e grau de ociosidade dos recursos. Quando nenhuma réplica é utilizada, a probabilidade de que a aplicação permaneça na máquina mais lenta ou na mais rápida é a mesma. Dessa forma, os tempos de execução extraídos possuem diferenças mais acentuadas. À medida que mais réplicas são utilizadas, esse cenário se altera já que aumenta-se a probabilidade de que uma das réplicas esteja sendo executada na máquina mais rápida do conjunto. Como o tempo de execução da réplica mais rápida é o que representa o tempo de execução de uma submissão, os tempos de execução são mais curtos e mais próximos entre si. Isso também explica a relação inversa entre o número de réplicas utilizado e o tamanho proporcional do desvio padrão. Mais réplicas em execução significam tempos de execução mais próximos entre si e, portanto, desvios padrão proporcionalmente menores. Os outros dois fatores, heterogeneidade e ociosidade, também contribuem na variação dos tempos de execução: como as réplicas são executadas em máquinas com configurações distintas, algumas réplicas evoluem mais

rapidamente do que outras. Além disso, por serem máquinas de uso compartilhado, o percentual de ociosidade das mesmas está sujeito a variações ao longo do tempo.

6. Conclusão e trabalhos futuros

Neste trabalho, nós argumentamos que o paradigma de agentes móveis se mostra bastante adequado para lidar com a complexidade inerente à infra-estrutura das grades e que isso se deve às características intrínsecas desse paradigma tais como cooperação, autonomia, heterogeneidade, reatividade e mobilidade. Foi apresentado o *middleware* MAG, que combina agentes móveis com replicação e técnicas de *checkpointing* para dar suporte à execução de aplicações seqüenciais longas em ambientes oportunistas. Nós argumentamos que essas técnicas podem ser combinadas de forma flexível para atender a diversas situações que envolvam variações na disponibilidade dos recursos. Nós demonstramos através de nossos experimentos que, em ambientes de computação oportunista, é essencial dar suporte a múltiplos mecanismos de tolerância a falhas para que se alcance alta desempenho mesmo na presença de falhas.

Nossos próximos passos incluem o ajuste dinâmico de alguns parâmetros de tolerância a falhas, como o número de réplicas e o intervalo de tempo entre *checkpoints*. O objetivo será fazer com que o mecanismo de tolerância a falhas do MAG reaja às mudanças no ambiente de execução e modifique o conteúdo dessas variáveis para valores aproximadamente ótimos, durante a execução das aplicações. Outro passo será o de utilizar a migração de agentes como estratégia de otimização de aplicações regulares e paramétricas. A execução de aplicações paramétricas também pode ser encarada como um caso semelhante ao da execução com réplicas, em que os argumentos de entrada para cada réplica são distintos, e o resultado final só é obtido a partir da junção de todos os resultados parciais devolvidos pelas réplicas. Assim, é necessário aguardar o término de todas elas, onde o tempo total de execução a ser considerado é o tempo de execução da réplica que se encerra por último. A partir desse cenário, o objetivo é detectar as réplicas mais lentas e, com os dados coletados pelo *LUPA*, avaliar se a sua migração para outra máquina pode reduzir o tempo total de execução da aplicação. Essa mesma técnica poderia também ser aplicada ao cenário onde duas ou mais réplicas (sejam da mesma aplicação ou de aplicações distintas) competem pelos recursos do mesmo nó. Por fim, outras intervenções futuras incluem a modificação do *AR* e do *StableStorage* para que explorem uma abordagem distribuída.

Referências

Bagaoglu, O., Meling, H., and Montresor, A. (2002). Anthill: A framework for the development of agent-based peer-to-peer systems. In *Proceedings of the 22nd ICDCS*, pages 15–22. IEEE CS Press: Vienna (A).

Barbosa, R. M. and Goldman, A. (2004). Framework for mobile agents on computer grid environments. In *In First International Workshop on MATA*, pages 147–157.

Barbosa, R. M., Goldman, A., and Kon, F. (2005). A study of mobile agents liveness properties on mobigrid. In *In 2nd International Workshop on MATA*.

Cao, J., Jarvis, S. A., Saini, S., Kerbyson, D. J., and Nudd, G. R. (2002). Arms: an agent-based resource management system for grid computing. *Scientific Programming (Special Issue on Grid Computing)*, 10(2):135–48.

Cao, J., Kerbyson, D. J., and Nudd, G. R. (2001). High performance service discovery in large-scale multi-agent and mobile-agent systems. In *International Journal of Software Engineering and Knowledge Engineering, Special Issue on Multi-Agent Systems and Mobile Agents.*, number 11 in 5, pages 621–641. World Scientific Publishing.

Chakravarti, A., Baumgartner, G., and Lauria, M. (May 2005). The organic grid: self-organizing computation on a peer-to-peer network. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(3):373–384.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L. B., Andrade, A., Novaes, R., and Mowbray, M. (2006). Labs of the world, unite!!! *Journal of Grid Computing*, 4(3):225–246.

Fukuda, M. and Smith, D. (2006). Uwagents: A mobile agent system optimized for grid computing. In *2006 International Conference on Grid and Applications - CGA'06*, pages 107–113, Las Vegas, NV.

Fukuda, M., Tanaka, Y., Suzuki, N., Bic, L. F., and Kobayashi, S. (2003). A mobile-agent-based pc grid. *Autonomic Computing Workshop, 2003*, pages 142–150.

Goldchleger, A., Kon, F., Goldman, A., Finger, M., and Bezerra, G. C. (2004). Integrade: object-oriented grid middleware leveraging the idle computing power of desktop machines. *Concurrency - Practice and Experience*, 16(5):449–459.

Hwang, S. and Kesselman, C. (2003). A flexible framework for fault tolerance in the grid. volume 1, pages 251–272.

Loke, S. W. (2003). Towards data-parallel skeletons for grid computing: An itinerant mobile agent approach. In *Proceedings of the CCGrid'03*, pages 651–652.

Lopes, R. F. (2006). Mag: uma grade computacional baseada em agentes móveis. Master's thesis, Universidade Federal do Maranhão, São Luís, MA, Brasil.

Lopes, R. F. and da Silva, F. J. (2006). Fault tolerance in a mobile agent based computational grid. In *4th International Workshop on Agent Based Grid Computing. CCGrid'06*, Singapore. IEEE Computer Society.

Lopes, R. F., da Silva e Silva, F. J., and Souza, B. B. (2005). Mag: A mobile agent based computational grid platform. In *Proceedings of CCGrid'05*, LNCS Series, Beijing. Springer-Verlag.

Martino, B. D. and Rana, O. F. (2004). Grid performance and resource management using mobile agents. In *Performance analysis and grid computing*, pages 251–263, Norwell, MA, USA. Kluwer Academic Publishers.

Sallem, M. A. S., de Sousa, S. A., and da Silva e Silva, F. J. (2007). Autogrid: Towards an autonomic grid middleware. In *16th IEEE International WETICE*.

Truyen, E., Robben, B., Vanhaute, B., Coninx, T., Joosen, W., and Verbaeten, P. (2000). Portable support for transparent thread migration in java. In *ASA/MA2000*, LNCS Series, Berlin. Springer-Verlag.

Veríssimo, P. and Rodrigues, L. (2001). *Distributed Systems for System Architects*. Springer, 1th edition.

Pré–escalonamento com QoS em *Grids* Computacionais utilizando Economia de Créditos e Acordos em Nível de Serviço

Matheus Bandini[1,2], Antonio R. Mury[1], Bruno Schulze[1], Ronaldo Salles[2]

[1]Computação Científica Distribuída (ComCiDis)
Laboratório Nacional de Computação Científica (LNCC)
Av. Getúlio Vargas, 333 – 25651–075 – Petrópolis – RJ – Brazil

[2]Departamento de Engenharia de Computação
Instituto Militar de Engenharia (IME)
Rio de Janeiro – RJ – Brazil

{mbandini,aroberto,schulze}@lncc.br, salles@de9.ime.eb.br

Abstract. *This paper proposes an architecture for the supply of Quality of Service on Grid computing (Grid–QoS) making use of a credit based Economy for the definition of Service Level Agreements (SLAs). Users are allowed to interact with the System on a transparent way, no matter their knowledge about the usage of a grid infrastructure. The System will use the information about the plan of service that has been chosen by the users to define the resources that will be used, the priority of each process and scheduling, as well as the charge each user will have to pay for the services.*

Resumo. *Este artigo apresenta uma proposta para o fornecimento de Qualidade de Serviço em Grids Computacionais (Grid–QoS) utilizando Economia de créditos para a definição de Acordos em Nível de Serviço (SLAs). Usuários são capazes de interagir com o Sistema de forma transparente, independente do conhecimento técnico sobre grids. O Sistema utilizará as informações sobre o plano de serviço escolhido pelo usuário para a definição dos recursos a serem utilizados, a prioridade de cada processo e escalonamento, bem como a tarifação a ser cobrada pelos serviços diferenciados de cada usuário.*

1. Introdução

O desenvolvimento tecnológico desencadeou um aumento considerável na capacidade computacional dos hardwares, que por sua vez, permite que desenvolvedores e pesquisadores imaginem novas formas de explorar ao máximo toda a capacidade computacional disponível [Foster et al. 2001]. No entanto, os elevados preços de computadores de última geração podem tornar inviável uma atualização constante dos *hardwares* de uma instituição [Foster et al. 2002]. Uma alternativa para solucionar o problema de custo é a utilização de *grids* computacionais, utilizando um determinado número de recursos (computadores e dispositivos) conectados entre si através de redes locais ou pela Internet, permitindo que o processamento de um ou mais recursos sejam utilizados de forma cooperativa para realizar uma tarefa em comum.

Os *grids* computacionais permitem o uso de recursos computacionais geograficamente distribuídos, como processamento e memória, através de esquemas de segurança

que envolvem a utilização de certificados que identificam os usuários aptos a utilizar um *grid* computacional [Foster et al. 2001].

Em sua forma nativa, um *grid* funciona basicamente da seguinte forma: um usuário define o número de recursos que ele deseja para executar a sua aplicação e envia a mesma, junto com os arquivos e parâmetros necessários para a execução. Em seguida, o *grid middleware* utilizado se encarrega de escalonar a tarefa para o número de recursos especificado pelo usuário, sem se preocupar com necessidades especiais requeridas pela aplicação ou pelo próprio usuário.

A utilização de *grids* em sua forma natural sempre atendeu as necessidades da comunidade científica, entretanto, devido ao crescimento da utilização de *grids* comerciais, foi observada a necessidade de aplicar meios que permitam a utilização de QoS em *grids* computacionais com o objetivo de otimizar o desempenho do ambiente utilizado, além de melhor atender às necessidades dos usuários que utilizam este ambiente [Sahai et al. 2003].

Baseado nisto, este trabalho visa aplicar Qualidade de Serviço em um ambiente de *grid* computacional, otimizando a utilização de recursos e fazendo com que o processo de escolha da QoS necessária seja realizada de forma transparente por um usuário, uma vez que é possível que usuários que não possuam conhecimento sobre computação também utilizem o ambiente de *grid*.

O objetivo deste trabalho é oferecer melhorias em relação aos serviços oferecidos aos usuários, melhorando a forma de interação com o Sistema, otimizar a utilização dos recursos do *grid* e realizar tarifição diferenciada para serviços distintos, baseado nos planos de serviço previamente escolhidos pelo usuário.

O artigo está organizado da seguinte forma: na seção 2, são apresentados alguns trabalhos existentes que abordam Grid–QoS de alguma forma e que foram utilizados como referência para a realização deste trabalho. Na seção 3, é traçado um paralelo entre Qualidade de Serviço em redes e *Grid–QoS*, para as duas abordagens, é feita uma análise dos parâmetros que define as definem. Na seção 4, é apresentado o modelo de Economia utilizado no trabalho e é proposta a inclusão de *Grid–QoS* baseando–se neste modelo. A seção 5 apresenta a implementação que vem sendo realizada com o objetivo de validar a proposta, enquanto que na seção 6, são apresentadas as considerações finais obtidas a partir do trabalho realizado.

2. Trabalhos Relacionados

Na literatura, existem diversas propostas e sistemas que tem por objetivo prover Qualidade de Serviço em *grids* computacionais (Grid-QoS). Cada uma delas, com características próprias que são vistas como referência a ser seguida, ou como fatores que devem ser modificados. Dentre eles, os que mais se relacionam com este trabalho e merecem destaque são: GARA [Roy 2001], G-QoSm [Al-Ali et al. 2004], AppLeS [EPCC 2003], Nimrod/G [Nimrod/G 2005]

2.1. GARA

O GARA [Roy 2001] surgiu como primeira ferramenta capaz de prover acesso à QoS fim-a-fim aos desenvolvedores, além de prover reserva antecipada e tratamento uniforme dos

recursos (rede, disco e CPU). A reserva de recursos do GARA garante que a aplicação irá receber os recursos reservados através do gerenciador de recursos [Roy 2001].

O fato de ter sido o primeiro sistema provedor de Grid–QoS faz com que o GARA possua algumas limitações relacionadas a tecnologia e usabilidade.

Por não ter sido desenvolvido utilizando *Web Services*, o GARA não é compatível com sistemas que utilizam esta tecnologia, caso do *Globus Toolkit* 4 [Al-Ali et al. 2004]. Além disso, o usuário é questionado sobre informações técnicas sobre recursos. Também não oferece atualizações dinâmicas sobre recursos que entram e saem do *grid*. Estas atualizações ocorrem apenas no início de cada sessão do usuário.

2.2. Grid QoS Management

O sistema G–QoSm (*Grid QoS Management*) [Al-Ali et al. 2004] foi desenvolvida com o objetivo de solucionar algumas funcionalidades não satisfatórias encontradas no GARA. Este sistema também oferece a possibilidade de reservar recursos antecipadamente, assim como gerenciar estas reservas, além de oferecer atualizações dinâmicas sobre os recursos que aderem e/ou deixam o *grid* computacional.

Quanto à tecnologia utilizada, o G–QoSm foi o primeiro sistema de *Grid–QoS* desenvolvido utilizando o conceito de *Web Services*, que permite maior integração com novas versões, caso do *Globus Toolkit 4*, e com aplicações que também utilizam esta tecnologia [Al-Ali et al. 2004].

A grande limitação encontrada no G–QoSm é o fato de que, assim como o GARA, é necessário que os usuários possuam conhecimentos técnicos sobre a utilização de um ambiente *grid* para que sejam capazes de otimizar o uso do sistema.

2.3. AppLeS

O AppLeS (*Application-Level Scheduling*) [EPCC 2003] é um sistema de escalonamento adaptativo a nível de aplicação que pode ser aplicado a um *grid* computacional [Li and Baker 2005]. É possível aplicar uma instância diferente do AppLeS a cada aplicação submetida no *grid*. O AppLeS tem como enfoque principal o fato de que os aspectos de desempenho do sistema e utilização são experimentados pela perspectiva da aplicação usando o sistema [Li and Baker 2005]. Para alcançar um maior desempenho da aplicação, o AppLeS mede o desempenho da aplicação em um determinado recurso e utiliza esta informação para tomar futuras decisões em relação ao escalonamento de novos recursos.

Embora o AppLeS utiliza escalonamento adaptativo, não possui características simples de um ambiente de *Grid–QoS* como reserva antecipada de recursos, além de também não abstrair informações técnicas para os usuários.

2.4. Nimrod/G

O Nimrod/G é uma ferramenta criada com o intuito de auxiliar o desenvolvimento de modelos de simulações de aplicações paramétricas utilizando *grids* computacionais. É utilizado para simular o comportamento de experimentos complexos em diversas áreas como engenharia, economia e ciências de forma geral [Nimrod/G 2005].

O Nimrod/G utiliza algumas características oferecidas pelo *Globus Toolkit* [Globus] como descoberta de recursos além de utilizar o conceito de economia computacional [Nimrod/G 2005].

No que diz respeito à qualidade de serviço em *grids* computacionais, o Nimrod/G permite que o usuário escolha os recursos descobertos através do *Globus Toolkit*, informando os tipos de recursos necessários para satisfazer as exigências da aplicação.

O principal problema do Nimrod/G é a falta de uma interface simples e transparente, o que causa um grave problema de usabilidade do sistema.

3. Qualidade de Serviço

A definição de Qualidade de Serviço em *Grids* Computacionais (*Grid-QoS*) é, de certa forma, uma extensão do que é definido por Qualidade de Serviço em redes. Uma vez que os objetivos principais em ambos os casos é oferecer serviços de melhor qualidade, oferecendo a opção de escolha do serviço que mais se adequa a cada caso específico. Preocupa–se também em realizar a cobrança destes serviços de forma justa e que reflita de fato a utilização dos recursos e serviços pelos usuários [Sahai et al. 2003].

Embora Qualidade de Serviço possua conceitos semelhantes tanto em redes quanto em *grids*, os fatores que caracterizam a oferta de um serviço de qualidade são diferentes e englobam características próprias.

A idéia de Qualidade de Serviço em Redes busca oferecer melhoria nos serviços oferecidos pela Internet que, ainda nos dias de hoje, oferece serviços baseados no "melhor esforço", uma herança da Arpanet que originou a grande rede na década de 70.

Para oferecer melhor QoS em redes, são analisados parâmetros de desempenho como o atraso de uma rede (*delay*), a variação em que este atraso ocorre (*jitter*), a taxa de transmissão de uma rede (*throughput*) e a taxa com que pacotes são perdidos em uma rede (*packet–loss rate*) [Katchabaw et al. 1998]).

Em QoS em redes, o nível de serviço, segundo [Gomes 1999], "expressa o grau de certeza de que a QoS será mantida pelo fornecedor", o que é diretamente dependente das políticas de provisão de QoS definidas pelo servidor.

Existem diversos exemplos de níveis de serviço, dentro dos quais, podemos citar os seguintes como sendo os mais comuns [ISO 1995]:

- *Melhor esforço (best effort)*: O sistema fornecedor de serviços não apresenta nenhuma garantia de que a qualidade de serviço será mantida dentro dos limites pré–estabelecidos;
- *Compulsório*: O sistema fornecedor de serviços monitora os recursos e pode atender às necessidades das aplicações de forma alternativa, caso os recursos requeridos não se encontrem disponíveis naquele momento, e até mesmo interrompendo o serviço, caso a qualidade de serviço não possa ser mantida dentro dos limites;
- *Garantido*: O sistema fornecedor de serviço aceita o usuário quando os fatores de QoS forem capazes de atender às necessidades das aplicações, conforme pré–estabelecido pelo nível de serviço associado ao usuário.

Da mesma forma com que é necessário medir fatores para prover QoS em redes, também é preciso fazer medições de parâmetros para que seja possível fornecer *Grid–*

QoS. Esses fatores são a capacidade de processamento e memória total e disponível dos recursos, fila de processos nos recursos, capacidade de armazenamento em disco e análise dos *links* que conectam os respectivos recursos, análise esta, feita utilizando os fatores de medição para QoS em redes [Al-Ali et al. 2004] [Katchabaw et al. 1998].

Naturalmente, a oferta de melhores serviços gera a necessidade de realizar tarifação diferenciada, uma vez que, se serviços diferenciados são oferecidos por tarifas iguais, a tendência é que os melhores recursos sempre sejam utilizados e acabem ficando sobrecarregados, fazendo com que a qualidade do serviço fique comprometida [Sahai et al. 2003]. Na próxima seção será apresentada uma proposta de tarifação de serviços com *Grid–QoS* baseada em créditos.

4. Economia Baseada em Créditos

Para viabilizar um sistema que seja capaz de fornecer Qualidade de Serviço, seja ela em nível de redes ou em uma infraestrutura de *grid*, é necessário que haja uma forma de tarifação para evidenciar a diferenciaçào dos serviços oferecidos. No entanto, o objetivo não é especificar em alguma unidade financeira quanto irá custar um serviço *grid* que possua determinada qualidade. Por isso, será utilizada uma espécie de Economia baseada em créditos, tomando como exemplo e base para a estipulação dos Acordos em Nível de Serviço (SLAs), a denominada Economia *Grid* Baseada em Créditos do Projeto VCG (*Virtual Community Grid*) [Schulze 2006], onde é realizada a adesão voluntária de novos usuários a projetos de uma infraestrutura *grid*. A inclusão de novos recursos e a utilização destes acarretam na geração de créditos a serem somados ao projeto ao qual o usuário faz parte.

Um dos objetivos de prover *Grid–QoS* utilizando Economia baseada em créditos e SLAs é conseguir abstrair informações técnicas, que nem sempre são conhecidas pelos usuários de um *grid*. Para alcançar este objetivo, são estabelecidos padrões de usuários no sentido de definir os planos de serviço onde serão embutidas informações como processamento, memória, prioridade requerida para a aplicação, etc. Desta forma, é possível permitir que usuários que detenham pouco ou nenhum conhecimento sobre *grid* sejam capazes de utilizar esta infraestrutura com Qualidade de Serviço e transparência.

Para prover Qualidade se Serviço em *grids* computacionais, são definidos planos de serviço baseado no que os usários dos projetos recebem de forma garantida ao assinarem aquele serviço (Tabela 1). Ou seja, para cada plano, os projetos obtém garantia dos respectivos recursos (CPU, memória, disco, prioridade de processos e rede) e são cobrados seguindo a taxa de cobrança de cada plano de serviço.

A tabela 1 indica os planos de serviço inicialmente propostos e as respectivas garantias dos parâmetros de *Grid–QoS* e acréscimo ao custo para cada SLA.

O plano de serviço Básico, a princípio, é considerado o plano padrão e adota características de “melhor esforço” e não gera acréscimo ao custo de utilização de recursos. No caso deste plano de serviço, as tarefas serão alocadas de forma aleatória, independente das necessidades da aplicação e dos recusrsos disponíveis.

Com relação aos planos de serviço Master e Premium, os recursos são pré-selecionados baseando-se nas informações de requisitos de cada plano e comparando com os recursos registrados na infraestrutura *grid*. Os recursos que obtiverem resultados satis-

fatórios são listadas como opções para a submissão de tarefas.

No caso de não haver recursos que atendam aos requisitos especificados nos planos de serviço Master e Premium, o usuário recebe uma notificação de que não há recursos disponíveis naquele momento e é oferecida a opção de utilizar outros recursos que podem não atender todas as especificações do plano de serviço, ou a opção de aguardar até que existam recursos disponíveis.

A definição dos parâmetros utilizados para a definição dos planos de serviço é feita de forma a evidenciar a importância dos mesmos no que diz respeito a uma infraestrutura de *grid*. Não é possível formular uma estratégia para qualidade de serviço em *grid* sem levar estes fatores em consideração. Da mesma forma, se dá a classificação destes fatores dentro dos planos de serviço especificados acima com o objetivo de atender aplicações de forma a oferecer maior performance para cada caso.

Tabela 1. Tabela de planos de serviço (SLAs).

Tipo SLA	CPU (% disponível)	Memória (% disponível)	Disco (% disponível)	Prioridade	Rede (atraso)	Taxa (custo + %)
Básico	Flutuante	Flutuante	10	Flutuante	Flutuante	+ 0 %
Master	50	40	15	Média	< 200 ms	+ 25 %
Premium	80	70	25	Alta	< 100 ms	+ 50 %

Os valores dos planos indicados pela tabela são valores arbitrários utilizados para testes do processo de pré–escalonamento e não são baseados em estatísticas de utilização de recursos do *grid* por determinados tipos de aplicação. A inserção de novos planos de serviço que sejam baseados em valores relevantes para tipos de aplicações específicas pode ser realizada independentemente do processo de pré–escalonamento, utilizando ferramentas de *benchmark* e aplicações de exemplo para avaliar o seu desempenho ideal em determinado recurso, tomando por base o trabalho de [Ye et al. 2006] que verifica o desempenho de um recurso computacional de acordo com uma aplicação específica, o que permite a realização futura desta atividade. A taxa de utilização de novos planos de serviço será definido a partir da utilização dos recursos para aquela aplicação.

É necessário destacar que a submissão de uma tarefa no *grid* será cobrada de acordo com o que é efetivamente utilizado de um recurso. Isto significa que, se uma tarefa é submetida segundo o especificado pelo plano Premium, mas, por alguma razão, não haja nenhum recurso que atenda a respectiva especificação, e recursos considerados inferiores no que diz respeito ao desempenho forem selecionados para a execução desta tarefa, o valor cobrado será recalculado de forma a refletir o recurso que, de fato, foi utilizado.

Os planos de serviço Master e Premium, onde existem valores mínimos de dempenho a serem cumpridos, também irão garantir que tarefas submetidas sob estes planos terão os respectivos valores reservados até o fim de sua execução. O plano de serviço Básico, por sua vez, possui valores flutuantes e não oferece garantia de que a capacidade de processamento inicial será mantida até o final de uma execução.

A classificação dos tipos de serviços é uma forma de garantir que aplicações detenham os recursos necessários para a sua execução de forma plena, no entanto, não é suficiente para que a cobrança possa ser feita de forma justa e que reflita a real utilização do ambiente. Como o ambiente *grid* é um ambiente heterogêneo, recursos distintos tanbém

devem ser classificados de forma a valorizar recursos com maior capacidade de processamento, memória e disco. A classificação de recursos foi realizada baseando–se nos recursos disponíveis no ambiente grid do laboratório de Computação Científica Distribuída (ComCiDis) do Laboratório Nacional de Computação Científica (LNCC) e se dá conforme a tabela 2.

Tabela 2. Tabela de classificação de recursos.

Tipo Recurso	Qtd núcleos CPU	CPU Total	Memória Total	Fator
A	2	Acima 2.0 GHz	2048 MB	3,0
B	2	Entre 1.5GHz e 2.0 GHz	2048 MB	2,8
C	2	Acima 2.0 GHz	1536 MB	2,6
D	2	Entre 1.5GHz e 2.0 GHz	1536 MB	2,4
E	2	Acima 2.0 GHz	1024 MB	2,2
F	2	Entre 1.5GHz e 2.0 GHz	1024MB	2,0
G	1	Entre 1.5GHz e 2.0 GHz	2048 MB	1,8
H	1	Entre 1.5GHz e 2.0 GHz	1536 MB	1,6
I	1	Entre 1.5GHz e 2.0 GHz	1024 MB	1,4
J	1	Entre 1.0GHz e 1.5 GHz	2048 MB	1,2
K	1	Entre 1.0GHz e 1.5 GHz	1536 MB	1,0
L	1	Entre 1.0GHz e 1.5 GHz	1024 MB	0,8
M	1	Até 1.0GHz	2048 MB	0,6
N	1	Até 1.0GHz	1536 MB	0,4
O	1	Até 1.0GHz	1024 MB	0,2

A utilização da tabela 2 permite que a qualidade de serviço seja diferenciada tanto pelo serviço em si, dependendo da alocação dos recursos, como também pelo tipo de recurso utilizado. Para isso, são utilizados os valores do campo fator como multiplicador que determina o valor daquele recurso. O fator de multiplicação é simplesmente um classificador de recursos de acordo com a quantidade de núcleos de CPU, Capacidade de processamento e memória total. Este fator permite também que novas classes de recursos sejam inseridas ou removidas, conforme a necessidade.

Para que o cálculo do custo total de créditos em uma submissão esteja de acordo com o estipulado, tanto para planos de serviço quanto para recursos diferenciados, e obedeça aos critérios de justiça de cobrança de qualidade de serviço, foi necessário o desenvolvimento de uma fórmula baseada nos parâmetros definidos nas tabelas 1 e 2:

$$C = S + (S.t) + (S/2.f) \tag{1}$$

onde *C* é o custo total da submissão no recurso selecionado, *S* é a quantidade de créditos utilizados em uma submissão, independente do tipo do plano de serviço e dos recursos utilizados, *t* é a taxa que representa o acréscimo relativo ao plano de serviço (tabela 1) e *f* é o fator de utilização do recurso em questão (tabela 2). O cálculo do custo será feito separadamente em cada um dos recursos utilizados para a submissão.

O surgimento de aplicações que demandam diferentes níveis de serviço faz com que seja preciso elaborar novos planos que, não necessariamente, atendam o que é definido em [ISO 1995], tanto em relação à forma com que o fornecedor garante a qualidade de serviço, quanto à variação dos parâmetros de *Grid–QoS*.

Além da classificação de parâmetros poder ser realizada de acordo com o nível de serviço desejado pelo usuário, ela também o podem ser classificada de acordo com a categoria de serviço. Assim sendo, "uma categoria de serviço relaciona um conjunto de parâmetros a um determinado tipo de ambiente" [Gomes 1999].

5. Implementação

A implementação deste trabalho está em desenvolvimento e está planejada em três etapas. A primeira parte, que é o foco principal deste artigo, aborda o pré–escalonamento de recursos baseado nas informações sobre os planos de serviços escolhidos por cada usuário. É necessário dizer que, no contexto do *Virtual Community Grid* [Schulze 2006], um ou vários usuário fazem parte de um projeto, isso significa que o projeto é que estará associado a um plano de serviços. Uma segunda etapa a ser realizada é a implementação de uma fila de processos, podendo ser feita no nível de escalonador do *middleware*, ou em uma camada de nível superior, evitando assim, desestabilizar versões estáveis e problemas de incompatibilidade com novas versões do *middleware*. Por fim, a terceira etapa será transformar o serviço de provisão de *Grid–QoS* em um *Web Service*, com o objetivo de facilitar a integração com os serviços do *Virtual Community Grid* já existentes. A figura 1 ilustra uma pré–visualização do sistema após o término da implementação e a inclusão do *Web Service*.

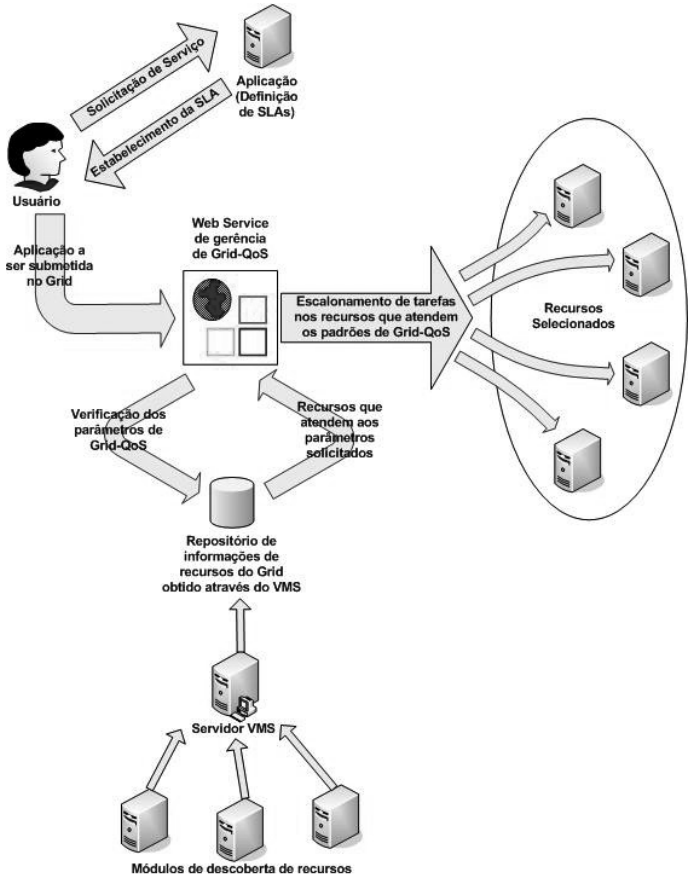


Figura 1. Estrutura prevista do módulo de pré–escalonamento transformado em *Web Service*.

Para que um sistema de Grid–QoS possa selecionar recursos de acordo com os planos de serviço especificados pelos usuários, é necessário que haja um repositório com estas informações. Este repositório pode ser alimentado tanto pelos serviços de monitoramento e descoberta de recursos dos *middlewares*, como por outras aplicações independentes desenvolvidas para este fim. No caso deste trabalho, a seleção de recursos para o

pré–escalonamento será feita através das informações obtidas pelo sistema de monitoramento utilizado e desenvolvido pelo Projeto VCG, denominado VMS (*Virtual Management System*).

O pré–escalonamento de recursos será feito levando em consideração a ordem de importância dos parâmetros. Em seu trabalho, [Al-Ali et al. 2004] considera a disponibilidade de CPU e a disponibilidade de memória nos recursos como os principais fatores de definição de *Grid–QoS*, logo, também classificamos estes dois parâmetros como os de maior importância, mas incluímos outros que também devem ser levados em conta. Desta forma, a ordem de relevância dos parâmetros está definida como se segue:

1. CPU disponível;
2. Memória disponível;
3. Espaço em disco disponível;
4. Capacidade da rede.

No que diz respeito à capacidade da rede, com o objetivo de não saturar as conexões entre os recursos do *grid* e levando em consideração que, atualmente, poucas conexões possuem suporte a QoS, a capacidade da rede será definida utilizando-se apenas do RTT (*Round Trip Time*) entre o Portal VCG e o respectivo recurso.

A associação entre o projeto e o respectivo plano de serviço será identificado através de um relacionamento entre tabelas do banco de dados incorporado ao Portal VCG, onde cada projeto estará relacionado a apenas um plano de serviço. Ao iniciar o processo de submmissão, o usuário do projeto irá receber informações relativas aos recursos que atendam aos requisitos de *Grid–QoS* do projeto ao qual ele está vinculado. A figura 2 ilustra o relacionamento entre usuários, projetos, serviços e recursos.

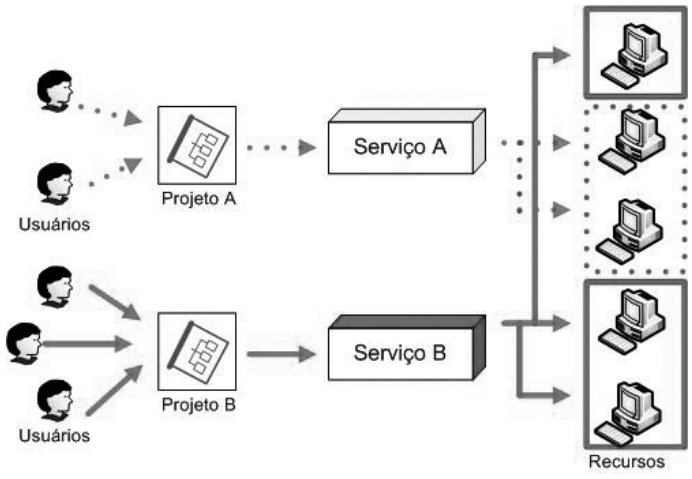


Figura 2. Relacionamento usuários/projetos/serviços/recursos.

Após o pré–escalonamento ser realizado baseado em informações de CPU, memória e disco, é realizado o teste de rede, onde acontece a medição entre os recursos previamente selecionados pelo sistema pré–escalonador e o servidor do Portal VCG. Apenas após esta última etapa, serão listados os recursos para que, enfim, o usuário possa escolher os que deseja utilizar, conforme a figura 3.

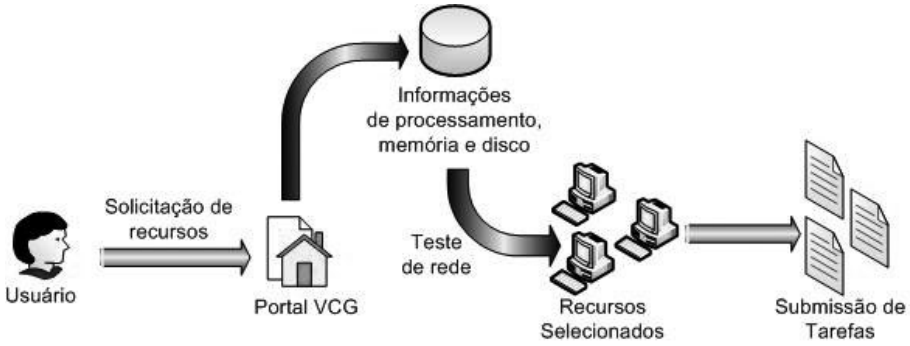


Figura 3. Fluxograma de seleção de recursos.

Com o objetivo de permitir que um usuário não fique sem opções caso não hajam recursos disponíveis que atendam aos requisitos de *Grid–QoS* especificados pelo respectivo plano de serviço, os valores não encontrados são gradativamente reduzidos a fim de fornecer uma configuração que, embora reduzida em termos de desempenho, atenda às necessidades do usuário/projeto o mais próximo possível do que está especificado no plano de serviço.

Para que o sistema de Grid–QoS seja consistente no sentido de realmente oferecer garantias de atendimento ao serviço selecionado, o sistema deve permitir que os usuários sejam capazes de selecionar e reservar os recursos de forma antecipada. da mesma forma, também deve ser possível que que os usuários sejam capazes de modificar ou desfazer uma reserva. Isto é feito com uma associação simples no banco de dados entre as tabelas de usuários e de reservas, onde esta última possui identificadores da reserva e do usuário que a realizou, os recursos selecionados e a data e hora em que a reserva deve ser submetida. Um exemplo de registros do repositório de dados que indica a associação usuários/reservas de recursos se encontra na tabela 3.

Tabela 3. Exemplos de registros da tabela de reserva de recursos.

id_reserva	id_usuario	recursos	data_hora
1	2	grade01.lncc.br- grade03.lncc.br	2008-05-02 22:30:00
2	5	grade04.lncc.br- grade05.lncc.br	2008-05-03 07:00:00

6. Considerações Finais

O desenvolvimento de técnicas de provisão de qualidade de serviço, tanto em redes quanto em *grids* computacionais vêm proporcionando uma melhor utilização de recursos e serviços, assim a definição de planos de serviços em SLAs permite que a cobrança por serviços diferenciados seja feita de forma justa e personalizada.

Este artigo visou introduzir fundamentos de *Grid–QoS* e apresentar a fase inicial do trabalho que vem sendo desenvolvido com o objetivo de incluir a provisão de qualidade de serviço em um ambiente de *grid* computacional com a característica de adesão voluntária de usuários e inclusão dinâmica de novos recursos. O que foi exposto no artigo é apenas a etapa que envolve a definição de planos de serviço que possam servir de base para o pré–escalonamento de recursos, uma vez que a redução da quantidade de recursos, a adequação dos mesmos de acordo com as necessidades dos usuários do *grid* e a

forma de tarifação são aspectos primordiais para se considerar um sistema de provisão de *Grid–QoS*.

O fato deste artigo abordar um trabalho em sua fase inicial de desenvolvimento faz com que ainda não haja um serviço para utlização em grande escala, no entanto, está previsto para as etapas posteriores do trabalho, o desenvolvimento de planos de serviço personalizados voltados para aplicações específicas, além da implantação de um serviço *web* que, não apenas facilitará comunicação entre o portal *web*, os middlewares e a base de dados, como também permitirá a ampla utilização do ambiente *grid* com qualidade de serviço, permitindo que novas informações sejam obtidas, promovendo assim o estudo das características do ambiente e uma conseqüente melhoria no fornecimento dos serviços.

7. Agradecimentos

Gostaríamos de agradecer ao Laboratório ComCiDis (Computação Científica Distribuída), situado ao LNCC (Laboratório Nacional de Computação Científica) e à RNP (Rede Nacional de Pesquisa e Ensino) por fornecer acesso ao ambiente *grid* e ao sistema de monitoramento de recursos VMS (*Virtual Management System*) do Projeto *Virtual Community Grid*. Agradecemos também ao Departamento de Engenharia de Computação do Instituto Militar de Engenharia na elaboração e realização deste trabalho.

Referências

Al-Ali, R., von Laszewski, G., Amin, K., Amin, K., Hategan, M., Rana, O., Walker, D., and Zaluzec, N. (2004). Qos support for high-performance scientific grid applications. In *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, pages 134–143, Washington, DC, USA. IEEE Computer Society.

EPCC (2003). Apples: Application–level scheduling system.

Foster, I., Kesselman, C., Nick, J., and Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed systems integration.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–25.

Globus, T. G. A. The globus toolkit.

Gomes, A. T. A. (1999). Um framework para provisão de qos em ambientes genéricos de processamento e comunicação.

ISO (1995). International organization for standardization/international eletrotechnical commitee. "qos: Basic framework". draft international standard (dis) 9309.

Katchabaw, M. J., Lutfiyya, H. L., and Bauer, M. A. (1998). A quality of service management testbed. In *SMW '98: Proceedings of the IEEE Third International Workshop on Systems Management*, page 57, Washington, DC, USA. IEEE Computer Society.

Li, M. and Baker, M. (2005). *The grid core technologies*. John Wiley & Sons.

Nimrod/G, T. N. P. (2005). Nimrod: Tools for distributed parametric modelling.

Roy, A. J. (2001). *End-to-end quality of service for high-end applications*. PhD thesis. Adviser-Ian Foster.

Sahai, A., Graupner, S., Machiraju, V., and van Moorsel, A. (2003). Specifying and monitoring guarantees in commercial grids through sla.

Schulze, B. (2006). Workgroup proposal: (vcg - virtual community grid).

Ye, D., Ray, J., Harle, C., and Kaeli, D. R. (2006). Performance characterization of spec cpu2006 integer benchmarks on x86-64 architecture. In *IISWC*, pages 120–127.

Compartilhamento Eficiente de Dados Ambientais em um Ambiente Distribuído

José Flávio M. V. Júnior, Ricardo A. Santos
Eliane Araújo, Lauro Beltrão Costa, Francisco Brasileiro

[1]Universidade Federal de Campina Grande (UFCG)
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Campina Grande – PB – Brasil

{zflavio, ricardo}@lsd.ufcg.edu.br

{eliane, lauro, fubica}@dsc.ufcg.edu.br

Resumo. *O compartilhamento eficiente de dados ambientais tem sido um grande desafio para pesquisadores e profissionais da área. A produção desses dados é interdisciplinar e inclui agências governamentais e de pesquisa e desenvolvimento. Por isso, os dados gerados são dispersos geograficamente e não possuem um mesmo formato, o que dificulta o compartilhamento. Este artigo apresenta o MetaFinder, uma solução aberta baseada em uma grade de dados (data grid), que viabiliza o compartilhamento de dados ambientais de forma escalável e tolerante a falhas. O sistema usa uma solução distribuída para transferência eficiente das informações e princípios de folksonomia, para equacionar problemas de incompatibilidades de vocabulário.*

Abstract. *An efficient environmental data sharing is a big challenge to researchers and professionals of this area. The work needed to produce this kind of data is interdisciplinary and is made by governmental agencies, research and development centres. So, the data generated are spread geographically and do not have a common format, what makes it difficult to share. This paper presents MetaFinder, an open solution based in a data grid, which makes environmental data sharing possible in an efficient and fault-tolerant way. The system uses a distributed solution for efficient data transfer. Moreover, it uses collaborative tagging methods, based on folksonomy principles, to solve problems related to vocabulary incompatibility.*

1. Introdução

A necessidade do estudo do meio ambiente, recursos naturais e suas interações com os seres humanos não é recente. Todavia, a relevância destes estudos, nos dias atuais, ganhou enormes proporções. Tornou-se freqüente a ocorrência de eventos extremos tais como furacões, enchentes, tempestades e grandes secas, nas mais diversas partes do planeta. O aquecimento global, hoje pauta das principais discussões entre líderes mundiais, acentua a necessidade de mais pesquisa na área de recursos naturais a fim de traçar metas e estratégias que permitam o desenvolvimento sustentável e a sobrevivência do planeta.

O trabalho com recursos naturais é interdisciplinar, o que pressupõe a colaboração entre pessoas das mais diferentes áreas de atuação: engenheiros, agrônomos, meteorologistas, gestores públicos e pesquisadores. Estes profissionais geralmente são lotados em

instituições diferentes e, muitas vezes, distantes umas das outras. Os dados ambientais produzidos por estas instituições são volumosos e, como exemplo da previsão de tempo, necessita de processamento de alto desempenho o que eleva o seu custo. A necessidade de uma estrutura computacional específica faz com que sejam poucos os centros produtores deste tipo de dado. Sendo assim, o compartilhamento de dados é vital para a comunidade que trabalha com recursos naturais. A dispersão geográfica dos dados e o fato de eles não serem produzidos de acordo com um mesmo formato dificultam o compartilhamento. Além disso, por lidar com times de profissionais diferentes, há dificuldades também na comunicação, já que o jargão utilizado por determinado grupo para caracterizar os dados é diferente do usado por outro grupo.

Esses desafios motivaram o desenvolvimento do *MetaFinder*: uma solução para o compartilhamento de dados ambientais que atendesse às necessidades da comunidade de recursos naturais. Os seus principais requisitos funcionais são: (i) oferecer uma interface de busca para dados ambientais, (ii) buscar dados em diversos servidores simultaneamente, (iii) lidar com as incompatibilidades no vocabulário dos envolvidos no processo e (iv) aperfeiçoar a recuperação dos dados, melhorando o processo de recuperação da informação. Além destes, alguns requisitos não-funcionais são desejáveis: (i) a solução deve ser escalável, (ii) tolerante a falhas, (iii) com controle descentralizado e (iv) lidar de forma transparente com diferentes formatos de dados.

A solução possibilita a criação de uma grade de dados (*data grid*) que viabiliza o compartilhamento e garante escalabilidade. Existem soluções que compartilham dos mesmos objetivos como Earth System Grid (ESG) [Bernholdt et al. 2005] e NERC DataGrid [Lawrence et al. 2004]. Diferente dessas soluções que usam um esquema fixo de metadados para descrever os dados, o MetaFinder utiliza técnicas de classificação colaborativa para construção de metadados assim como para equacionar problemas relativos à incompatibilidade de vocabulários. ESG utiliza o protocolo GridFTP para transferência de dados, enquanto que o MetaFinder propõe o uso de uma solução que permite fazer recorte sobre eles, evitando o tráfego desnecessário na rede. A grade de dados NERC vai além dos propósitos do MetaFinder incluindo uma camada de segurança, que restringe o acesso a dados e metadados. O MetaFinder assume que os dados são públicos.

Além disso o MetaFinder utiliza um sistema de informação de grades (*GIS - Grid Information Service*) entre-pares (*peer-to-peer*) para a descoberta de dados geograficamente dispersos e uma interface de busca regida pelos princípios da *folksonomia*, que permite aos usuários do sistema, quer sejam eles provedores ou consumidores de dados, participar ativamente do processo de classificação. O MetaFinder está disponível para acesso no endereço http://opendapmetafinder.lsd.ufcg.edu.br/, em carater experimental, para acesso dos interessados e pesquisadores da área.

Neste artigo, serão apresentadas a arquitetura e a implementação do MetaFinder e será descrito como os requisitos da solução foram atendidos. O artigo está estruturado da seguinte forma. A seção 2 apresenta os principais conceitos envolvidos na classificação colaborativa de dados, a seção 3 apresenta a arquitetura da solução, a seção 4 aborda as escolhas de implementação que foram tomadas enfatizando as suas razões técnicas. Serão apresentadas as nossas considerações finais na seção 5, seguida das referências utilizadas na condução deste trabalho de pesquisa e desenvolvimento.

2. Classificação colaborativa de dados

Diariamente, grandes quantidades de dados e informações, nos mais diversos formatos, são disponibilizadas na Web. De modo geral, organizá-los de forma a facilitar a sua localização não tem sido uma tarefa trivial. Isso por que as formas até então existentes para organizar e classificar dados têm se tornado inadequadas por ter semântica limitada e muitas vezes complicada de se manter e usar [Xu et al. 2006]. O uso de vocabulários controlados oferece grande potencial de interoperabilidade e escalabilidade. Apesar disso, esta não é uma forma popular de realizar descoberta de conteúdos na Web, principalmente devido ao alto custo de manutenção e a amplitude de assuntos em algumas áreas. Nestes casos, o vocabulário controlado não consegue atender às necessidades eficazmente [Macgregor and Mcculloch 2006]. Estudos recentes têm mostrado que ontologias funcionam bem apenas quando a quantidade de itens que se deseja classificar está restrita a um domínio pequeno e específico [Shirky 2005].

Padrões de metadados, como DIF[1] e Dublin Core[2], também têm sido uma abordagem proposta neste sentido. O DIF é um formato padronizado e descritivo para troca de informações sobre conjuntos de dados científicos provendo uma rica taxonomia com aproximadamente 12.000 termos usados para definir que informações estão contidas no conjunto de dados sendo descrito. Nós utilizamos o DIF de forma experimental para descrever os dados que gostaríamos de compartilhar. Essa experiência mostrou que os usuários sentem dificuldades em relacionar o verdadeiro significado das variáveis que compõem os dados com os termos disponíveis, chegando a não conseguir encontrar os termos adequados para classificá-los. Além disso, a expansão da taxonomia é difícil, uma vez que a adição de um novo termo é um processo extremamente burocrático. Como dito em [Mathes 2004], construir metadados pode ser uma tarefa difícil, dispendiosa e não escalável, quando uma grande quantidade de dados é produzida e usada, além da desvantagem de manter o processo de criação da taxonomia alheia aos usuários.

A classificação colaborativa de conteúdo na Web é uma nova abordagem que tem se popularizado através de sistemas como o Delicious[3], Flickr[4] e CiteULike[5]. Esses sistemas permitem que os usuários associem rótulos (*tags*) livremente a um determinado conteúdo disponível na Web, a fim de que sejam facilmente encontrados e compartilhados com outros usuários. Nesse caso, a evolução do vocabulário é regida pelo princípio da *folksonomia*. A palavra *folksonomia* é uma combinação de "folks" (que significa pessoas) e "taxonomia", e se refere à forma livre como os conteúdos são classificados na Web. Esta abordagem tem sido utilizada quando a quantidade de conteúdo a ser classificado é muito grande e dispersa, para que uma só entidade controladora possa fazê-lo, ou mesmo quando tal entidade não existe para definir as regras de classificação.

Diferente de taxonomias, que utilizam um modelo hierárquico e exclusivo para classificar e organizar as informações, os sistemas de classificação colaborativa utilizam um modelo plano (*flat*) e livre, onde os itens classificados podem pertencer a várias classes ao mesmo tempo. Os rótulos estabelecem relações entre os dados uma vez que um

[1]Directory Interchange Format (DIF) - http://www.citeulike.org/.
[2]Dublin Core Metadata Initiative - http://dublincore.org/.
[3]Delicious - http://del.icio.us.org/.
[4]Flickr - http://flickr.com/.
[5]CiteULike - http://www.citeulike.org/.

mesmo rótulo pode ser utilizado várias vezes para classificar diferentes itens. Isso permite melhorar significativamente o resultado das consultas já que itens relacionados podem ser facilmente recuperados. Uma comparação detalhada sobre o modelo hierárquico das taxonomias e o modelo plano dos sistemas de classificação colaborativa é apresentada em [Golder and Huberman 2005].

O caráter aberto da folksonomia permite a participação direta dos usuários no processo de criação da terminologia através da associação de rótulos aos conteúdos. Dessa forma, a terminologia utilizada para classificar os dados tende a ser a mais próxima da utilizada pelos usuários. Essa metodologia permite que grandes quantidades de dados sejam classificadas com um menor esforço. A inexistência de regras ou qualquer entidade centralizadora ou controladora permite a livre expansão do vocabulário. Além disso, a evolução desse vocabulário que está sendo criado dinamicamente é auxiliada pela interface do sistema de classificação, através de mecanismos de destaque e sugestão de rótulos.

A eficiência e popularização dos mecanismos de classificação colaborativa para descrição de conteúdos distribuídos nos têm feito acreditar que ele pode ser aplicado em outros campos, como na descrição/classificação de dados ambientais, uma vez que: (i) minimiza o problema de terminologias conflitantes, pois os termos comuns conhecidos pelos usuários serão utilizados para descrever os dados e suas variáveis; (ii) independe de qualquer formato ou padrão de metadado; (iii) não sofre com alterações hierárquicas ou estruturais da terminologia; (iv) atualizações são automaticamente incorporadas à terminologia sem qualquer restrição ou avaliação prévia; (v) o sistema de classificação é auto-gerenciável, com os rótulos mais populares ou mais recomendados sendo automaticamente sugeridos a fim de melhor classificar os dados e (vi) permite uma melhoria gradativa na eficiência das consultas, à medida que se adicionam rótulos que fortalecem o significado dos dados.

Contudo, essa solução exige uma rica interface com o usuário, a fim de fornecer o suporte para classificação adequada dos dados, bem como para a manutenção da terminologia sendo criada. Além disso, por ser um vocabulário não controlado, a *folksonomia* possui algumas limitações que devem ser trabalhadas de acordo com as restrições do sistema que a utiliza, como o problema da ambigüidade, rótulos compostos e explosão de rótulos, descritos em [Golder and Huberman 2005, Xu et al. 2006, Mathes 2004].

3. Arquitetura do MetaFinder

Os elementos que compõem a arquitetura do MetaFinder são: o catálogo de informações, o servidor de dados, o agente publicador e um portal para consulta e classificação.

O MetaFinder usa um catálogo entre-pares para descoberta de dados, permitindo a localização eficiente de dados geograficamente dispersos, utilizando técnicas de roteamento e balanceamento de carga nas consultas. O acesso aos dados é feito independente de sua localização e de forma eficiente, possibilitando a recuperação apenas de um subconjunto dos dados e minimizando a sobrecarga de transferência na rede. A independência de formatos de armazenamento também é garantida através da transferência de dados em um formato intermediário. Isso facilita a sua utilização, já que não é necessário conhecer detalhes sobre como e em qual formato os dados são armazenados.

Além da abordagem entre-pares para descoberta e acesso a dados distribuídos, o MetaFinder utiliza a abordagem de classificação colaborativa para descrição dos da-

dos que são originalmente caracterizados com diferentes terminologias. Regida pelos princípios de *folksonomia*, a classificação colaborativa vai permitir que os usuários participem diretamente da descrição dos dados através da definição de rótulos, melhorando sua compreensão e facilitando a localização. Essa abordagem dispensa o uso de qualquer taxonomia ou vocabulário controlado, uma vez que a terminologia é definida pelos próprios usuários, resolvendo o problema da incompatibilidade de vocabulários das pessoas envolvidas no processo de produção dos dados. Além disso, ela melhora a comunicação, uma vez que a terminologia utilizada para classificar os dados tende a ser cada vez mais próxima da habitualmente utilizada pela comunidade.

Um portal Web auxilia os usuários na associação de rótulos aos dados e provê os recursos necessários para que a terminologia comum surja e evolua de acordo com a utilização do sistema. O portal também disponibiliza um mecanismo de busca que permite a realização de consultas por faixas de valores de atributos além de permitir o uso de rótulos escolhidos pelo usuário ou sugeridos pelo sistema para uma recuperação mais eficiente dos dados.

A arquitetura proposta para o MetaFinder permite que os sites produtores de dados publiquem seus dados na comunidade de forma que qualquer usuário possa encontrá-los e acessá-los diretamente, independente de onde os dados estão armazenados. A Figura 1 exibe os quatro principais componentes da arquitetura do MetaFinder: o portal de busca e classificação, o catálogo de informações, o servidor de dados e o agente publicador.

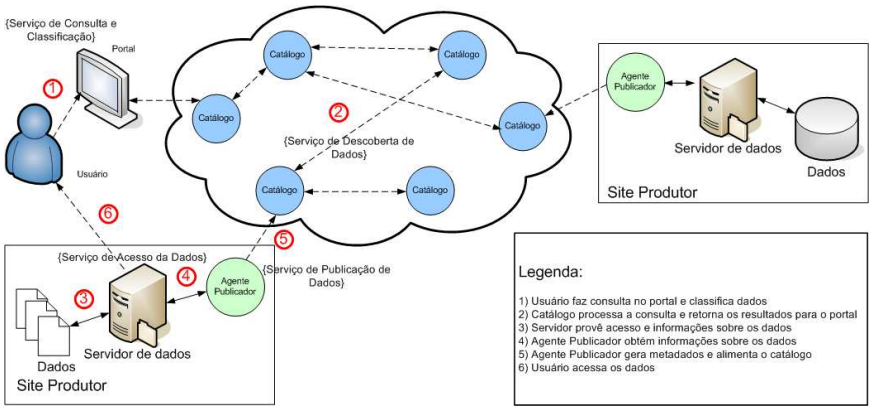


Figura 1. Arquitetura do MetaFinder

O Site produtor, representa um centro produtor de dados. Ele armazena dados produzidos em diferentes formatos. O componente Servidor de dados permite o acesso aos dados independente de formato de armazenamento. Os dados liberados para acesso, são publicados em um catálogo por um agente que reside do lado do site provedor. O catálogo mantém as informações sobre os dados disponíveis de forma que os usuários possam localizá-los através de submissão de consultas. Uma interface de busca e classificação é oferecida para que os usuários possam localizar e classificar os dados disponíveis. Essa interface é provida através de um portal Web. O usuário submete sua consulta através do portal, que se comunica com o catálogo. O catálogo localiza os dados disponíveis fazendo o roteamento das consultas por diversas instâncias do catálogo, distribuídas na rede, devolvendo o resultado em seguida para o portal. O portal lista para o usuário as informações sobre os dados localizados, bem como o endereço remoto através do qual o

usuário pode ter acesso aos dados. De posse desse endereço remoto, o usuário pode ter acesso às partes do dado que lhe interessam, através de uma aplicação cliente que sabe trocar informações com o Servidor de dados. O funcionamento de cada componente da arquitetura do MetaFinder é descrito em mais detalhes a seguir.

3.1. Portal

O portal provê a interação entre o usuário final e o sistema MetaFinder. Ele oculta do usuário todos os detalhes de como as informações são localizadas, fornecendo uma interface amigável para localização e classificação dos dados. A primeira função do portal é possibilitar a localização dos dados que estão sendo compartilhados. Para isso, o usuário deve fornecer valores para os atributos de cobertura espacial e temporal (e.g., latitude, longitude, data inicial e data final) e rótulos que descrevam os dados (e.g., climatologia, Nordeste, UFCG), além das variáveis que o usuário deseja que estejam contidas nos dados (e.g., temperatura, vento, precipitação). Uma sentença de busca é criada e enviada ao catálogo de informações, que retornará uma lista de resultados com a localização dos dados que atendem aos requisitos da consulta especificada.

A outra função do portal é possibilitar a classificação dos dados usando marcações por rótulos. Isso pode ser feito tanto pelo usuário como pelo administrador dos dados. Inicialmente, a classificação é feita pelo administrador dos dados, quando os dados são publicados no sistema (o processo de publicação é descrito na Seção 3.4). Isso permite que os dados recém publicados possam ser localizados logo em seguida, usando os rótulos fornecidos pelo administrador. A participação do usuário no processo de classificação se dá através do fornecimento de mais rótulos, reforçando o uso de rótulos anteriormente utilizadas por outros usuários ou adicionando novos. Para realizar a classificação, o usuário precisa localizar algum dado através do sistema de busca. Uma vez localizados os dados desejados, o usuário pode visualizar os rótulos já fornecidos e assim adicionar ou remover seus próprios rótulos para aquele dado. Os rótulos fornecidos pelos usuários são armazenados no catálogo e passarão a compor a terminologia que vai sendo criada gradativamente, à medida que novos rótulos vão sendo adicionados no sistema.

3.2. Catálogo de Informações

O catálogo de informações é um serviço descentralizado para a descoberta de dados. Ele armazenará meta-informações sobre os dados compartilhados como localização, por exemplo, balanceando a carga entre suas instâncias de forma a contribuir para a escala do sistema. Assim, consultas serão submetidas a qualquer dessas instâncias e roteadas para aquelas que contêm as meta-informações procuradas que, por sua vez, alimentarão o portal com os resultados.

A falha de uma instância do catálogo não deixará o serviço de localização indisponível, já que as outras instâncias realizam a mesma função. A pena sob a falha de uma instância do catálogo é a indisponibilidade de localização de uma pequena parte dos dados cujas informações ele armazenava. Esse problema pode ser mitigado usando uma estratégia de atualização de estado fraca (*soft state*). Nessa estratégia as informações armazenadas no catálogo têm um prazo de validade e desaparecem após transcorrido esse prazo. Os serviços que atualizam o catálogo devem, de tempos em tempos, republicar as informações que ainda são válidas e cujo prazo de validade expirou. Esse mecanismo provê uma maneira simples de “limpar” o catálogo, ao mesmo tempo que permite que as

informações perdidas por conta da falha de uma instância do catálogo sejam naturalmente re-inseridas no mesmo.

3.3. Servidor de Dados

O Servidor de Dados é o componente responsável por viabilizar o acesso remoto aos dados locais armazenados em um site produtor. Ele provê uma camada lógica que fornece informações sobre os dados que estão sendo compartilhados, bem como otimiza a transferência dos dados via rede. Para satisfazer os requisitos da nossa arquitetura, o servidor de dados deve prover independência de formatos de armazenamento, podendo lidar com vários formatos de arquivos bem como formas diferentes de armazenamento, como sistema de arquivos e banco de dados relacional.

Manter os dados no próprio centro produtor possibilita-nos alcançar escalabilidade de gerenciamento, uma vez que cada site é responsável por manter seu Servidor de Dados ativo. Outra vantagem é que qualquer decisão sobre que dados podem ser compartilhados ou até quando, é resolvida pelo próprio produtor dos dados, sem que seja necessária a intervenção de terceiros.

O Servidor de Dados deve prover também um mecanismo que permita fazer "recorte de dados", especificando-se as coordenadas que delimitam a área de interesse e/ou o intervalo de tempo de interesse, e a transferência de apenas a parte relevante dos dados deve ser feita, aumentando a eficiência na obtenção dos mesmos.

3.4. Agente Publicador

O agente publicador é um componente que reside no lado do provedor e é responsável por anunciar no catálogo de informações os dados que estão sendo compartilhados. Ele obtém informações sobre os dados através do servidor de dados. A partir dessas informações, são gerados metadados e publicados no catálogo, juntamente com o endereço remoto de cada dado. Esse endereço é utilizado pelo usuário para ter acesso aos dados que foram encontrados através do portal.

4. Detalhes de Implementação

Diante da arquitetura proposta do MetaFinder, nós definimos quais tecnologias deveriam ser implementadas e quais tecnologias existentes poderiam ser utilizadas, visando aproveitar os benefícios fornecidos por tecnologias já consolidadas em diferentes áreas da computação. Como o MetaFinder foi proposto para ser uma solução aberta, o uso de tecnologias abertas constituiu-se um requisito fundamental. Isso permite que alterações na implementação do MetaFinder para atender a requisitos particulares dos usuários possam ser feitas sem maiores restrições.

4.1. Serviço de Acesso a Dados

A implementação do Serviço de Acesso a Dados (Servidor de dados) foi possível com o uso de uma solução aberta para acesso a dados ambientais chamada OPeNDAP [Cornillon et al. 2003]. O OPeNDAP (Open-source Project for a Network Data Access Protocol) é um framework que facilita o acesso remoto a dados locais, com as vantagens de fornecer independência de formato e a possibilidade de fazer recorte de dados com parâmetros desejados. A tecnologia OPeNDAP usa um modelo cliente-servidor.

Informações como endereço e nome dos recursos e parâmetros associados ao refinamento dos dados são representados numa URL e o protocolo HTTP é usado para o transporte das requisições e respostas. Consegue-se uma grande redução na quantidade de dados transmitidos pela rede com o uso de recorte de dados. Considere um site produtor de dados de previsão climática, abrangendo a área do país, e um consumidor que deseja analisar dados referentes à Paraíba. Uma vez que o consumidor obtenha a URL dos dados necessários à sua análise, ele a fornece ao seu cliente OPeNDAP e esse emite uma requisição e recebe os dados referentes somente à área da Paraíba. Como os dados são transferidos em um formato intermediário, o cliente também não precisa conhecer o formato do dado que está sendo acessado ou de que forma está sendo armazenado (e.g., sistema de arquivos ou banco de dados relacional). Existem diversos clientes OPeNDAP disponíveis, como também o próprio usuário pode estender sua aplicação com o suporte fornecido pela comunidade OPeNDAP [Araújo et al. 2006].

4.2. Serviço de Publicação de Dados

Com o uso do OPeNDAP, o site produtor tem seus dados mapeados em URLs. Essas URLs precisam ser publicadas no catálogo juntamente com meta-informações sobre os dados que estão sendo compartilhados. O *Crawler* é a implementação do Agente Publicador, responsável por essa tarefa. Ele cria um índice com as URLs dos dados disponíveis no servidor OPeNDAP. A partir desse índice, ele obtém informações de cobertura temporal e espacial dos dados e gera um metadado que é publicado no catálogo distribuído. A varredura no servidor OPeNDAP para detectar alterações nos dados é feita de tempos em tempos, dessa forma, garante que o catálogo tenha sempre informações atualizadas sobre os dados, adicionando novos metadados e removendo-os quando os dados não estiverem mais disponíveis. Como o catálogo distribuído está sujeito a falhas que podem culminar na perda de parte das informações, o Crawler republica os metadados em intervalos regulares, mais freqüentes que a varredura feita no servidor OPeNDAP. O Crawler foi totalmente escrito em Java, de forma a garantir a independência de plataforma.

4.3. Serviço de Descoberta de Dados

O NodeWiz [Basu et al. 2005] atua como catálogo de informações e é utilizado como serviço de descoberta de dados. Ele é um GIS organizado de maneira distribuída em uma arquitetura entre-pares (P2P) para a descoberta de quaisquer tipos de recursos. Suas principais vantagens sobre outros sistemas de GIS P2P são: (i) atender a requisições ricas e.g. consultas por faixas (*range queries*); e, (ii) prover uma distribuição eficiente da informação pela rede P2P de maneira que a carga fique balanceada, o que é essencial para a escalabilidade de sistemas como grades de dados. Cada provedor publica os seus recursos especificando um conjunto de atributos que os descrevem e os seus endereços.

Cada recurso tem atributos que podem assumir vários valores. Por exemplo, $rainFall = true, longitudeMax = 37^o14', logitudeMin = 20^o19', produtor = \text{“}LSD''$. A responsabilidade que cada *peer* tem sobre os dados é definida através do espaço de valores dos atributos. Caso um *peer* não seja o responsável pela faixa de atributos especificada pela busca ou pelos atributos de um novo recurso a ser publicado, a consulta ou a publicação é roteada pela rede P2P até o *peer* responsável por esse espaço de atributos. Essa característica é fundamental para os dados ambientais onde consultas por faixas é comum para especificar limites espaciais (latitude, longitude) e temporais (dias, meses) dentre outros.

Para tolerar falhas, os *peers* do NodeWiz mantêm um mecanismo distribuído de detecção de falhas. Quando uma falha ocorre, a rede P2P detecta e um outro *peer* que já faz parte do sistema assume a responsabilidade sobre o espaço de atributos do *peer* falho. As publicações dos recursos mantidas pelos *peers* falhos são perdidas, mas são novamente armazenadas quando o serviço de publicação de dados republica a informação. A descrição detalhada do mecanismo de tolerância a falhas do NodeWiz pode ser encontrada em [Brasileiro et al. 2006].

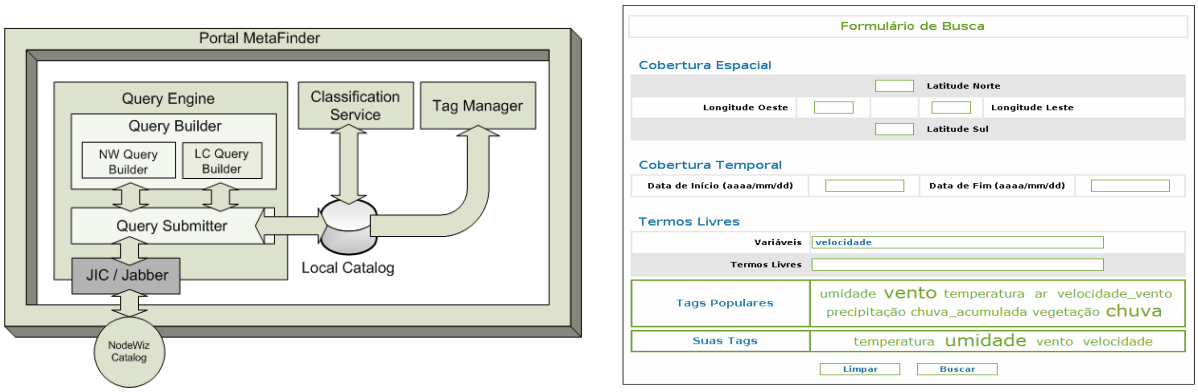
4.4. Serviço de Busca e Classificação de Dados

Para prover o serviço de busca e classificação colaborativa de dados, foi implementado o portal MetaFinder. Em termos arquiteturais, como mostra a Figura 2(a), o portal consiste em um mecanismo de consulta, classificação e gerenciamento de rótulos, além da interface gráfica que fornece o ambiente para utilização desses mecanismos. O mecanismo de consulta (*Query Engine*) recebe parâmetros fornecidos pelo usuário na interface e gera consultas, que são encaminhadas para o catálogo de informações. Como o NodeWiz, atualmente, não fornece suporte a consultas por rótulos, um catálogo local (*Local Catalog*) implementado usando uma base de dados relacional provê tal funcionalidade. Dessa forma, a consulta é realizada em duas fases: uma primeira consulta é gerada contendo faixas de valores referentes aos campos fixos dos dados e é encaminhada para o NodeWiz; uma segunda consulta é elaborada a partir dos resultados obtidos pelo NodeWiz de forma a filtrá-los usando os rótulos. Essa segunda consulta é enviada para o catálogo local que filtrará apenas os endereços dos dados que foram classificados com os rótulos informados. Futuras melhorias estão previstas para serem implementadas no NodeWiz de forma a permitir a realização de consultas mais ricas, como consultas por rótulos, o que dispensará o uso do catálogo local.

A comunicação do portal com o NodeWiz se dá através de troca de mensagens assíncronas usando JIC (Java Internet Communication) [Lima et al. 2006] que utiliza o protocolo Jabber (XMPP). O mecanismo de classificação de dados (*Classification Service*) trata das rotinas relacionadas com a classificação dos dados usando rótulos. Ele recebe a URL do dado que está sendo classificado e os rótulos fornecidos. Os rótulos são normalizados para que se possa eliminar repetições. Os rótulos são então associados à URL do dado e ao usuário que está classificando e são armazenadas no catálogo local. O gerenciador de rótulos (*Tag Manager*) trata das rotinas que estão diretamente relacionadas com a obtenção de rótulos no catálogo. Ele é responsável por obter rótulos populares, rótulos de usuários, calcular os índices de popularidade dos rótulos em relação a outros, montar nuvens de rótulos entre outras funções relacionadas. A comunicação desses mecanismos com o catálogo local é feita através do framework *Hibernate*[6].

A localização dos dados é feita através de um formulário de busca. Visando aumentar o número de dados relevantes encontrados, o formulário de busca (Figura 2(b)) foi desenvolvido utilizando uma abordagem híbrida. Ele é composto por campos fixos, onde os usuário podem especificar valores para atributos de cobertura espacial e temporal dos dados. Além de campos para informar rótulos que determinam variáveis e características dos dados. Segundo [Sinclair and Cardew-Hall], formulários devem ser utilizados quando os atributos são particulares o suficiente. Já a utilização de rótulos

[6]Hibernate - http://www.hibernate.org/.



(a) Arquitetura do portal MetaFinder (b) Formulário de busca do portal MetaFinder

Figura 2. Mecanismos implementados pelo portal

contribui positivamente quando os resultados não são tão específicos. No caso da comunidade de recursos naturais, encontrar dados que cubram determinadas áreas em determinado período é indispensável. Por isso, não poderíamos nos limitar a fornecer consultas somente por rótulos, como é comum observar em sites de classificação colaborativa de conteúdos. O portal ainda auxilia o usuário nas suas consultas através de sugestões de rótulos. Ele sugere os rótulos que têm sido mais utilizados pela comunidade usuária para classificar os dados, dessa forma, aumentando a eficiência das consultas e endereçando o problema de vocabulários conflitantes no processo de produção dos dados. Tais sugestões são exibidas através de "nuvens de rótulos", um artifício visual para o conceito de listas ponderadas, ressaltando os mais populares. As nuvens de rótulos também têm por função guiar o vocabulário da *folksonomia*.

Os usuários registrados podem classificar os dados resultantes de suas consultas, adicionando rótulos para classificar variáveis ou o dado inteiro. Para guiar a escolha dos rótulos, também são dadas sugestões com os rótulos mais usados tanto globalmente, quanto pelo usuário até o momento. Uma vez que o usuário classifique um dado ou variável, seu rótulo passa a interferir nas sugestões dadas a outros usuários. Além disso, os usuários registrados também possuem uma área pessoal (*User Library*) onde é possível salvar uma lista de dados de interesse. Nessa área, o usuário pode visualizar informações sobre estes dados e, se desejar, classificá-los novamente.

4.5. Lidando com os problemas da folksonomia

Contudo, por ser um vocabulário não controlado, o conjunto de rótulos da *folksonomia* possui algumas limitações que devem ser tratadas de acordo com as restrições dos sistemas que a utilizam, como o problema de ambigüidade, rótulos compostos e aumento irregular na popularidade de determinado rótulo. No MetaFinder, nós desconsideramos o problema de ambigüidade de rótulos. Segundo [Golder and Huberman 2005], os usuários utilizam os sistemas de rótulos da maneira mais correta possível, para o próprio benefício, e acabam contribuindo automaticamente para a comunidade. Dessa forma, nós acreditamos que a medida que os usuários classificam os dados de seu interesse de maneira correta, sem utilizar jargões ou palavras ambíguas, a terminologia tende a evoluir de forma

correta e os rótulos mais adequadas passarão a ser sugeridos automaticamente.

Para simplificar a implementação do mecanismo de classificação por rótulos nós não permitimos que termos compostos sejam entendidos como um único rótulo. Nós nos baseamos na regra de "termos separados por espaços" para identificar rótulos. Dessa forma, simplificamos também o processo de inferência do mecanismo de busca.

O problema de "explosão de rótulos" é o aumento irregular na popularidade de um rótulo, que surge quando o usuário administrador classifica seus dados. Como o processo de classificação feito pelo administrador envolve muitos dados de uma só vez, os rótulos usados por ele têm uma popularidade maior que a dos usuários comuns, pois eles são multiplicadas pelo número de dados sendo classificados. Assim as nuvens de rótulos tendem a sugerir mais os rótulos de usuários administradores. Em [Xu et al. 2006] o problema é resolvido atribuindo pesos aos rótulos, de acordo com a pontuação de autoridade (*authority score*) do usuário, de forma que usuários que seguem a maioria ganham pontos positivos e seus rótulos têm um impacto maior na comunidade. No MetaFinder o problema é resolvido ajustando a contabilidade da popularidade dos rótulos balanceando com o número de usuários diferentes que usam um mesmo rótulo, sem que seja necessário atribuir pontuações aos usuários.

5. Conclusões e Trabalhos Futuros

Neste trabalho, nós buscamos viabilizar a realização de pesquisas na área das ciências ambientais fornecendo um mecanismo para localização e acesso eficiente aos dados que são produzidos diariamente por diversos centros operacionais, visto que o compartilhamento eficiente desses dados tem sido um grande desafio para os pesquisadores e profissionais da área.

Para realizar o compartilhamento de dados ambientais é necessário saber onde estão os provedores, permitir a realização de consultas sobre seus conjuntos de dados e viabilizar a recuperação dos dados de maneira que eles possam ser utilizados. O MetaFinder satisfaz todos esses requisitos além de garantir escalabilidade, usando um catálogo entrepares, e tolerância a falhas, uma vez que a falha em algum de seus componentes tornaria indisponível apenas uma parte dos dados compartilhados.

O MetaFinder dispensa a utilização de vocabulário controlado e opta pela abordagem da classificação colaborativa dos dados. A participação dos usuários no processo de classificação é de fundamental importância para a melhoria da eficiência das consultas, fornecendo rótulos relevantes que dão significado aos dados, fazendo com que resultados mais precisos sejam retornados. Outro ponto é a facilidade na geração dos metadados, que se resume ao fornecimento de rótulos para descrever os dados e as suas variáveis. Essa forma de descrição de dados faz com que a terminologia utilizada dentro do sistema seja a mais próxima possível da terminologia conhecida pelos usuários.

O MetaFinder já está em funcionamento em caráter experimental e seu portal está disponível no endereço http://opendapmetafinder.lsd.ufcg.edu.br/. Pretende-se incorporá-lo ao projeto SegHidro (http://seghidro.lsd.ufcg.edu.br/), que tem demandado uma solução aberta para compartilhamento de dados ambientais. Como trabalhos futuros, pretendemos avaliar o comportamento do sistema medindo o tempo de resposta das consultas e analisando a evolução da terminologia usada pelos usuários para descrever os dados. Mediante os resultados dessa avaliação, poderemos propor melhorias ao sistema de

descoberta de dados e de sugestão de rótulos. Além disso, pretendemos realizar melhorias no catálogo de forma a dar suporte a rótulos para classificação de recursos, assim, dispensando o uso de uma base de dados relacional.

Agradecimentos

Este trabalho foi desenvolvido em colaboração com a HP Brasil P&D.

Referências

Araújo, E., Carvalho, M., Medeiros, A. C., Cirne, W., de Oliveira Galvão, C., de Souza, E. P., and Brasileiro, F. V. (2006). Usando Grades Computacionais na Melhoria da Gestão de Recursos Hídricos. In *Primeiro Workshop sobre Clusters e Grids Aplicados a Governança Eletrônica - ParGov 2006*.

Basu, S., Banerjee, S., Sharma, P., and Lee, S.-J. (2005). Nodewiz: Peer-to-peer resource discovery for grids. In *Proceedings of* 5^{th} *International Workshop on Global and Peer-to-Peer Computing*, pages 213–220. IEEE Computer Society Press.

Bernholdt, D., Bharathi, S., Brown, D., Chanchio, K., Chen, M., Chervenak, A., Cinquini, L., Drach, B., Foster, I., et al. (2005). The Earth System Grid: supporting the next generation of climate modeling research. *Proceedings of the IEEE*, 93(3):485–495.

Brasileiro, F., Costa, L. B., Andrade, A., Cirne, W., Basu, S., and Banerjee, S. (2006). A large scale fault-tolerant grid information service. In *MCG '06: Proceedings of the* 4^{th} *international workshop on Middleware for grid computing*, page 14, New York, NY, USA. ACM Press.

Cornillon, P., Gallagher, J., and Sgouros, T. (2003). OPeNDAP: Accessing data in a distributed, heterogeneous environment. *Data Science Journal*, 2:164–174.

Golder, S. A. and Huberman, B. A. (2005). The Structure of Collaborative Tagging Systems. *Technical report, In-formation Dynamics Lab, HP Labs*.

Lawrence, B., Cramer, R., Gutierrez, M., van Dam, K., Kondapalli, S., Latham, S., Lowry, R., O'Neill, K., and Woolf, A. (2004). The NERC DataGrid: "GOOGLING"secure data. *Proceedings of the UK e-Science All Hands Meeting*.

Lima, A., Cirne, W., Brasileiro, F., and Fireman, D. (2006). A case for event-driven distributed objects. In 8^{th} *International Symposium on Distributed Objects and Applications (DOA)*.

Macgregor, G. and Mcculloch, E. (2006). Collaborative Tagging as a Knowledge Organisation and Resource Discovery Tool. *Library Review*, 55:291–300.

Mathes, A. (2004). Folksonomies-Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication*.

Shirky, C. (2005). Ontology is Overrated: Categories, Links and Tags.

Sinclair, J. and Cardew-Hall, M. The folksonomy tag cloud: When is it useful? *Journal of Information Science*, 34(1):15–29.

Xu, Z., Fu, Y., Mao, J., and Su, D. (2006). Towards the semantic web: Collaborative tag suggestions. *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland,May*.

Um modelo para sistemas de backup baseado em serviços

Paulo Ricardo Zanoni[1], **Luis Carlos Erpen de Bona**[1]

[1]Departamento de Informática – Universidade Federal do Paraná

{paulo, bona}@c3sl.ufpr.br

Resumo. *Este artigo apresenta os principais problemas relacionados aos sistemas de backup distribuído e abordagens para suas soluções. Um modelo para implementação de serviços de backup distribuído também é proposto. O modelo é dividido em serviços que colaboram entre si. Os serviços de grades computacionais podem ser utilizados para a implementação dos mesmos.*

1. Introdução

A perda de dados armazenados em sistemas computacionais pode trazer grandes prejuízos. Por isso, sistemas de backup que permitem recuperar dados em caso de falhas são fundamentais. As grades computacionais podem oferecer de forma segura uma grande capacidade de armazenamento distribuído, provendo assim um ambiente ideal para implementar sistemas distribuídos de backup. Neste artigo são apresentados os principais problemas dos sistemas de backup distribuído e abordagens resolvê-los. Outra contribuição deste artigo é apresentar um modelo para construção de sistemas de backup distribuídos. O modelo é dividido em serviços com tarefas específicas que comunicam entre si e com serviços de informações das grades subjacentes para realizar a tarefa do backup.

2. Sistemas de Backup Distribuído

O primeiro passo na realização de um backup é a seleção dos dados a enviar. Inicialmente, todos os arquivos selecionados são enviados. Em backups posteriores pode-se enviar apenas aqueles alterados ou então somente as alterações, como no algoritmo *rsync* [Tridgell 1999]. Determinar essas diferenças é uma tarefa custosa, porém sistemas de arquivos especiais são uma alternativa para realizar o processo. Após a seleção, os dados devem ser preparados para o envio. Uma das decisões é a escolha do método de redundância a ser utilizado. Pode-se espalhar réplicas dos arquivos pela rede ou então adotar um algoritmo de dispersão da informação. Essa escolha impacta na disponibilidade e na confiabilidade do sistema [Rodrigues and Liskov 2005]. O uso de controle de versão também pode alterar a forma com que os arquivos são enviados, além de implicar em maior espaço utilizado pelo backup. Outros fatores relevantes incluem a compressão dos arquivos; o uso de criptografia e assinaturas digitais, que aumenta a segurança mas pode criar a necessidade de armazenar arquivos como chaves digitais fora do sistema; e também a união de arquivos pequenos em conjuntos maiores, visando diminuir uma sobrecarga de envio de informações [Lian et al. 2005]. Então, os dados devem ser distribuídos na rede. Esta tarefa envolve selecionar os nodos receptores e enviar as informações. Pode-se utilizar métodos centralizados ou então tabelas hash distribuídas. Após o envio, a disponibilidade dos dados deve ser garantida. Para tanto, uma estratégia é monitorar a rede. Deve ser possível detectar se um nodo saiu da rede apenas temporariamente ou permanentemente, se os arquivos que foram enviados ainda estão armazenados em seus

receptores e também se ainda há a necessidade de guardar backups recebidos de outros nodos [Lillibridge et al. 2003].

Entretanto, os serviços existentes nas grades computacionais, como os de infraestrutura de segurança, gerenciamento de alocação de recursos, monitoramento e descoberta de informações podem ser utilizados para resolver muitos dos problemas relacionados ao backup distribuído [Foster and Kesselman 1997].

3. Sistema proposto

O sistema proposto neste artigo é composto de cinco serviços. A implementação de cada um pode ser feita de maneira independente, permitindo o uso tanto das técnicas convencionais quanto dos sistemas das grades computacionais.

O serviço de informações tem conhecimento de quais nodos estão disponíveis no momento. Ele também guarda informações sobre o espaço disponível em cada nodo e as utiliza para encontrar receptores para os backups realizados. Ainda, ele guarda uma lista contendo todos os destinos de todos os backups realizados. O serviço de envio é o responsável por descobrir os dados alterados, prepará-los e enviá-los. Ele pode, por exemplo, selecionar os arquivos, calcular suas diferenças, dividi-los em pedaços, comprimi-los, encriptá-los, aplicar o *erasure coding* e então enviá-los. O serviço de armazenamento mantém e gerencia os arquivos recebidos de outros nodos. Caso haja falta de espaço em disco ele pode descartar alguns desses arquivos. O serviço de recuperação possui uma interface com o usuário que é utilizada para recuperar os dados em caso de falha. Para tanto, ele comunica-se com os outros serviços. Por fim, o serviço de monitoramento e controle verifica constantemente se os arquivos enviados para a rede permanecem acessíveis, se os arquivos armazenados de outros nodos ainda são necessários e também minimiza os impactos negativos do sistema de backup no computador sobre o qual está funcionando. Todas essas tarefas são realizadas através da comunicação com os outros serviços.

4. Considerações finais e trabalhos futuros

Definir um sistema de backup distribuído envolve resolver diversos problemas. Este artigo mostrou esses problemas e suas alternativas de solução. Um modelo para implementação de sistemas de backup distribuído foi proposto. Este modelo é composto de cinco serviços que juntos realizam a tarefa do backup distribuído. Como trabalho futuro resta refinar a especificação do sistema e implementá-lo. O ambiente em que se pretende utilizá-lo é a rede de 2100 laboratórios do Projeto Paraná Digital.

Referências

Foster, I. and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *The Intl. Journal of Supercomputer App. and High Performance Computing*, pages 115–128.

Lian, Q., Chen, W., and Zhang, Z. (2005). On the impact of replica placement to the reliability of distributed brick storage systems. In *ICDCS '05*, pages 187–196.

Lillibridge, M., Elnikety, S., Birrell, A., Burrows, M., and Isard, M. (2003). A cooperative internet backup scheme. In *USENIX ATEC '03*, pages 3–3.

Rodrigues, R. and Liskov, B. (2005). High availability in dhts: Erasure coding vs. replication. In *Fourth International Workshop on Peer-to-Peer Systems*.

Tridgell, A. (1999). *Efficient Algorithms for Sorting and Synchronization*. PhD thesis.

Um mecanismo de *Checkpointing* para ZeliGrid

Jeane Cezário e Alexandre Sztajnberg

DICC/IME, Universidade do Estado do Rio de Janeiro (UERJ)

{jeane, alexszt}@ime.uerj.br

Abstract. *In this work we present a mechanism for state persistence (checkpointing) and migration of the components, integrated to ZeliGrid, aiming to maintain the application running according to its non-functional requirements, with no need to reinitiate its components when a reconfiguration occurs.*

Resumo. *Neste trabalho apresentamos um mecanismo de persistência do estado (checkpointing) e migração dos componentes da aplicação, integrado à ZeliGrid, com o objetivo de manter a aplicação executando segundo seus requisitos não-funcionais, sem a necessidade de reiniciar seus componentes quando ocorre uma reconfiguração.*

1. Introdução

Neste trabalho apresentamos um mecanismo de *checkpointing,* com persistência de estado, e migração, integrado ao ZeliGrid [Granja 2006], um *middleware* que oferece suporte à execução, de aplicações distribuídas com requisitos não-funcionais dinâmicos em um ambiente de grades computacionais. ZeliGrid provê serviços que permitem à aplicação: (i) definir seus requisitos não-funcionais; (ii) descobrir os recursos disponíveis na grade e selecionar os nós da grade que melhor atendam a estes requisitos, (iii) implantar e executar componentes distribuídos; e um (iv) serviço de reconfiguração, que possibilita à aplicação ter seus componentes dinamicamente adaptados em face à mudança na disponibilidade dos recursos utilizados. O mecanismo sendo proposto evita que os componentes reconfigurados sejam completamente reiniciados.

2. Integração de *checkpointing* ao ZeliGrid

Em nosso trabalho, fizemos uso da serialização de objetos da plataforma Java [Sun 2008] para realizar o salvamento do estado de um componente, transformado-o em objeto, e persistindo o mesmo em uma memória estável, a cada intervalo de *checkpoint*. O mecanismo de serialização permite transformar um objeto em um fluxo de dados, que pode ser usado para transportar o objeto por uma conexão de rede, por exemplo, ou persisti-lo em uma memória não volátil, como um arquivo em disco. Da mesma forma, um objeto serializado pode ser reconstituído em uma nova instância [Sun 2008].

3. Arquitetura do módulo de checkpointing

A arquitetura é apresentada através da Figura 1, o módulo de *checkpointing*, e a Figura 2, que contém o diagrama de seqüência com as interações entre as classes da Figura 1 e as classes dos módulos de ZeliGrid no salvamento um componente (a seqüência de restauração não foi ilustrada por questões de espaço).

1. A classe *Aplicação* deve estender a classe abstrata *Checkpoint*, implementando em seu método *run()* todo o processamento a ser executado. O método *main()* deve criar uma instância da classe *Aplicação*, e uma instância da classe *SingleJobExecutor*.
2. Ao ter seu método *run()* ativado classe *SingleJobExecutor* instancia a classe *Temporizer*, passando a referência ao objeto da classe *Aplicação* (3:) e inicia duas *threads*: a do objeto *Aplicação* e a do *Temporizer* (4: e 5:).
3. A *thread* do objeto *Temporizer*, controla o intervalo de *checkpoint*. No momento de *checkpoint* a *thread* da *Aplicação* é suspensa pela *thread* de *Temporizer*, o objeto é serializado, persistido, e em seguida a *thread* da *Aplicação* é reativada (6:, 7: e 8:).
4. *CheckpointHandler* contém os métodos para serializar (*saveState*) e recuperar (*restoreState*) o estado do objeto que representa a classe da aplicação.

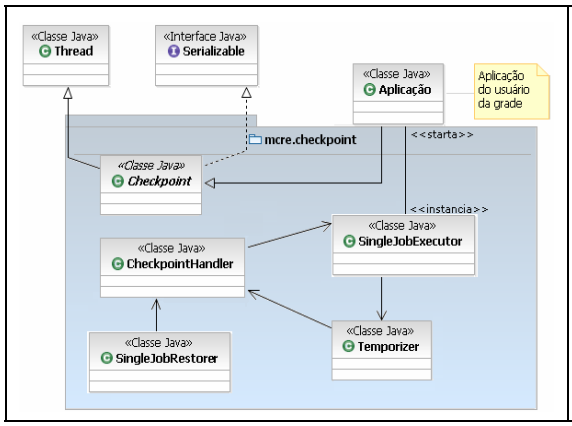


Figura 1. Diagrama de Classes

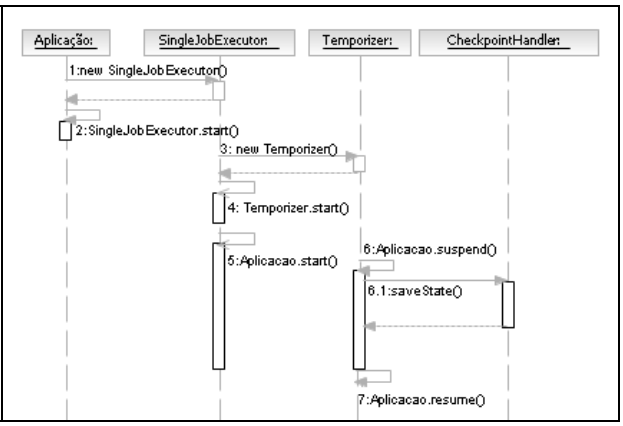


Figura 2. Salvamento de Estado

O método *restoreState()* é responsável por recuperar o objeto serializado e passar sua referência para a classe *SingleJobExecutor*. Esta realiza novamente as ações descritas no passo 2, utilizando como parâmetro o objeto reconstruído a partir do arquivo com a imagem serializada do objeto *Aplicação*. Dessa maneira, a execução do componente é reiniciada a partir do estado salvo no último *checkpoint*.

4. Conclusão

A técnica de checkpointing proposta, embora simples, provê alguma tolerância a falhas e torna a execução das aplicações em ZeliGrid mais eficiente. Entretanto, impõe ao desenvolvedor uma disciplina de programação através de herança e na adoção de um padrão de estrutura dos objetos. Estamos também cientes das limitações técnica da abordagem, dado que a pilha de execução não é salva. Atualmente, a implementação do mecanismo de *checkpointing-migração* e a integração com ZeliGrid estão sendo refinadas, e possíveis soluções para contornar as limitações estão sendo investigadas.

Referências

Granja, S. R., Sztajnberg, A., "Zeligrid: uma arquitetura para a implantação de aplicações com requisitos não-funcionais dinâmicos em Grades Computacionais", IV WCGA / SBRC 2006, Curitiba, Junho, 2006.

Sun Microsystems, "Java Object Serialization Specification", Janeiro, 2008. http://java.sun.com/j2se/1.3/docs/guide/serialization/spec/serialTOC.doc.html

Uma Arquitetura para Otimização de Desempenho em Grades Computacionais Oportunistas

Raphael de Aquino Gomes[1], **Fábio Moreira Costa**[1], **Fouad Joseph Georges**[1]

[1]Instituto de Informática – Universidade Federal de Goiás
Bloco IMF 1, Campus Samambaia, CEP: 74001-970 Goiânia – GO, Brazil

{raphael,fmc}@inf.ufg.br, fouadjg@gmail.com

Resumo. *Investigamos o uso de técnicas de recuperação de falhas e adaptação dinâmica como forma de alcançar uma melhor qualidade do serviço concedido às aplicações e de otimizar o desempenho global em grades oportunistas. Os resultados desse trabalho serão incorporados no middleware InteGrade.*

1. Introdução

Grades computacionais oportunistas são essencialmente formadas por nós heterogêneos e não-dedicados, oferecendo garantias limitadas a um **melhor-esforço** que não é desempenhado com todo rigor. O desempenho desses ambientes depende das aplicações locais, pois na ocorrência destas os recursos devem ser liberados. Existe a possibilidade de utilização de recursos adicionais, sendo migração a ação mais indicada. Mas migrações frequentes podem afetar a execução, além da possibilidade de não existirem recursos adicionais [Liang and Nahrstedt 2004] e o fato de que a falta de recurso pode ser passageira ao ponto da migração não ser justificável.

Este trabalho discute a análise de padrões de aplicações locais versus o estado de progresso das tarefas de grade como forma de racionalizar as migrações dessas tarefas e, consequentemente, otimizar o desempenho. É proposto o uso de técnicas de adaptação dinâmica como alternativa à migração. Trata-se de um trabalho em andamento, sendo desenvolvido sobre o middleware de grade InteGrade [Goldchleger 2004].

2. Arquitetura para Otimização de Desempenho

Apresentamos uma arquitetura para otimização do desempenho no middleware InteGrade. A Figura 1 ilustra os módulos a serem incluídos:

Local user pattern anomaly and Burst Analiser **(LBA):** Localizado nos nós provedores de recursos. Responsável por analisar a utilização dos recursos buscando identificar comportamentos anômalos ou consumo que prejudique aplicações de grade.

Performance Manager **(PM):** Localizado no nó de gerenciamento. Responsável por decidir se a tarefa deve ser migrada ou se algum mecanismo adaptativo deve ser acionado. Também seria responsável por criar réplicas de aplicações que estivessem enfrentando o problema.

Adaptation Manager **(AM):** Com uma versão nos nós compartilhados e uma versão global situada no nó de gerenciamento, é responsável por implementar mecanismos adaptativos. Esses mecanismos podem atuar no nó onde a tarefa está executando como alternativa à migração ou podem atuar nos módulos da grade.

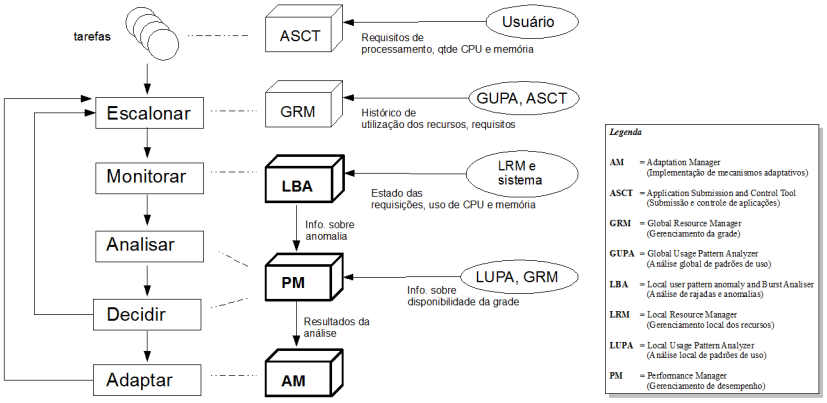


Figura 1. Arquitetura para Otimização de Desempenho no InteGrade.

De acordo com o novo protocolo de execução de aplicações, após a submissão da aplicação e início de sua execução, o LBA passaria a monitorar seu estado e a utilização dos recursos. Ao detectar utilização do recurso acima de um certo nível, que perdure por um período não-suportável, o LBA aciona o PM.

O PM instancia uma réplica da aplicação e analisa os dados fornecidos pelo LBA buscando identificar se a utilização dos recursos configura uma rajada ou anomalia. Em ambos os casos, o PM compara o tempo de sua duração com o tempo necessário para migrar a aplicação para outro nó, visando identificar a melhor saída entre migração e adaptação. Informações adicionais, como características dos nós, podem ser considerados na comparação.

A aplicação é migrada para outro nó, ou são realizados mecanismos adaptativos pelo AM (que visa permitir a continuação da execução da aplicação de grade no nó atual. Como alternativa, ou mesmo como operação adicional, a adaptação pode ser feita em outros componentes da grade visando evitar que o problema encontrado ocorra novamente).

3. Considerações Finais

Nos propomos a investigar uma forma mais eficiente de recuperação, tanto em termos de ganhos de desempenho quanto em relação a questões de sincronismo, concorrência e consistência entre as tarefas da aplicação no middleware InteGrade. Uma outra contribuição é a inclusão de mecanismos adaptativos nesse middleware, os quais permitirão a adaptação dinâmica dos serviços visando melhorias em seu desempenho global.

O trabalho encontra-se atualmente em andamento, com a definição da arquitetura geral já consolidada. As próximas etapas consistirão na implementação e avaliação das técnicas propostas, procurando demonstrar e quantificar seu uso e benefícios.

Referências

Goldchleger, A. (2004). Integrade: Um sistema de middleware para computação em grade oportunista. Master's thesis, Instituto de Matemática e Estatística – Universidade de São Paulo.

Liang, J. and Nahrstedt, M. (2004). Supporting quality of service in a non-dedicated opportunistic environment. *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*, pages 74–81.

Uma Estratégia de Baixo Custo Computacional para Levantamento e Informação de Recursos em uma Grade

Lourival A. de Góis[1], Marcos M. Tenório[1], Walter da Cunha Borelli[2]

[1]PIBIC/FUNTEF – Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa – PR – Brasil

[2]Departamento de Telemática, Faculdade de Engenharia Elétrica e de Computação – Universidade de Campinas (UNICAMP)

gois@utfpr.edu.br, mmtenorio@gmail.com, borelli@dt.fee.unicamp.br

Resumo. *Este artigo propõe a aplicação de uma estratégia de levantamento e informação de recursos ociosos em um sistema de gerenciamento descentralizado. A abordagem adotada prioriza o baixo custo computacional inerentes a estes processos através da geração dos índices de carga e ociosidade através de vários tipos de médias móveis que refletem de forma diferente a situação atual dos recursos compartilhados, bem como a comunicação dos mesmos ao gestor do domínio sem sobrecarregar a infra-estrutura de redes.*

1. Introdução

Para que os serviços de uma grade computacional possam ser oferecidos, são vários os desafios que deverão ser assumidos na concepção desta plataforma, como por exemplo, o levantamento de recursos ociosos, os contratos estabelecidos entre seus participantes e os mecanismos de escalonamento de tarefas [Kacsuk 2002]. Em [Kale 2003] foram abordadas algumas técnicas para gerenciamento de recursos em ambientes distribuídos de larga escala, no entanto, não foram considerados nos mecanismos utilizados para levantamento e informação dos índices de carga, problemas relacionados aos custos computacionais envolvidos nos processos. Este artigo estuda e apresenta um modelo de gerenciamento de recursos descentralizado para uma grade computacional proprietária [Góis 2006], propondo um mecanismo de descoberta e informação de recursos que priorize a diminuição dos *overheads* inerentes a estes processos.

2. Definição dos Índices de Disponibilidades Através de Médias Móveis

A abordagem adotada neste trabalho visa o levantamento de recursos ociosos em um ambiente distribuído heterogêneo, a partir do conhecimento prévio de recursos em cada nó gerenciado, com freqüência adaptativa de envio de mensagens para atualização de seus índices de carga. Com isto em foco, em intervalos periódicos cada proprietário de recursos envia para o gerenciador sua taxa de utilização de processador e de memória. Estas informações serão normalizadas segundo índices de performances gerados por um utilitário de *benchmark* e registradas em um banco de dados centralizado para serem utilizadas posteriormente pelos mecanismos de escalonamento de tarefas.

Para estes estudos foram utilizadas as médias móveis [Spiegel 2001], as quais buscam a partir de um conjunto de informações levantadas sobre recursos em um determinado período, estabelecer suas médias de utilização e disponibilidades para então, transmiti-las ao nó gestor visando subsidiar a determinação do período de

amostragem e o tipo de média mais apropriados para o modelo apresentado. Para isto foi utilizado um conjunto de microcomputadores de configuração bastante heterogênea interconectados por uma rede local e em pleno uso por seus usuários em suas atividades normais. Seus dados foram coletados em intervalos fixos de cinco segundos e os experimentos foram repetidos alternando a média móvel em estudo sob amostragens contendo 10, 30 e 60 coletas. Os resultados obtidos a partir destes experimentos tiveram como objetivo a identificação de qual das médias móveis apresentaram valores que melhor acompanham os índices atuais de carga dos equipamentos em estudo. Dentre elas, a que mais se aproximou deste objetivo foi a média móvel ponderada por utilizar em seu cálculo a atribuição de pesos maiores aos índices mais recentes, sendo portanto, selecionada neste trabalho como base para o algoritmo adotado pelos proprietários de recursos para a definição e o envio de seus índices de carga ao nó gestor. Este algoritmo é baseado em índices gerados por este tipo de média com transmissões condicionadas aos limites de carga estabelecidos nos contratos firmados entre os participantes do aglomerado.

7. Conclusões

Neste artigo foi abordado um dos desafios da modelagem e implementação de ambientes computacionais distribuídos que é o gerenciamento de recursos, por ser a base dos processos de escalonamento, alocação e execução de aplicações distribuídas. Foi proposto e implementado um modelo descentralizado de gerenciamento de recursos para uma grade computacional priorizando o baixo consumo de processador, memória e infra-estrutura de redes nos processos envolvendo a geração dos índices de carga e as transferências dos mesmos para o gestor do sistema. A abordagem utilizada foi a utilização de mecanismos que comprovaram que é possível manter índices de carga confiáveis no gestor gerados através da média da carga apresentada pelos agregados em períodos pré-estabelecidos. Foram estudadas no modelo proposto as médias móveis simples, modificada, exponencial e ponderada, e concluiu-se que a exponencial é mais apropriada para períodos de coletas inferiores a cinqüenta segundos e a ponderada, por valorizar os índices mais recentes, foi adotada neste trabalho por gerar valores que mostraram claramente a tendência dos índices de carga dos recursos sob diferentes períodos, provendo portanto, o nível de confiabilidade do modelo de escalonamento de tarefas a ser utilizado na grade.

8. Referência Bibliográfica

Kacsuk P., Kranzlmuller D., Volkert J. and Nemeth Z., (2002). “Distributed and Parallel Systems: Cluster and Grid Computing”, In Kluwer International Series in Engineering and Computer Science.

Kale L. V., Kumar S., Potnuru M., DeSouza J., and Bandhakavi S. (2004). Faucets: Efficient Resource Allocation on the Computational Grid, International Conference on Parallel Processing (ICPP 2004).

Spiegel, M. R. (2001). Probabilidade e Estatística. Ed. Makron Books, RJ.

Gois L.A and Borelli W. C. (2006). Um ambiente hierárquico para Definição e Negociação de Recursos Ociosos em uma Grade Institucional, 24º SBRC, IV WorkShop de Computação em Grid e Aplicações.

Análise e disposição de recursos de rede em grades computacionais

Alex Camargo, Alfredo Goldman

[1]Instituto de Matemática e Estatística – (IME/USP) São Paulo – SP – Brasil

acamargo@gmail.com, gold@ime.usp.br

Abstract. *Grid computing frameworks has been a popular theme in the last years, for being a concrete proposal for managing distributed resources. Some of this frameworks, like Integrade [1], are focused on the execution of strongly coupled parallel aplications, where communication optimization is a crucial efficiency factor. In this paper we study the inference of the local network's physical topology.*

Resumo. *Arcabouços de grades computacionais têm se popularizado nos últimos anos, por serem uma proposta concreta para gerenciamento e uso de recursos distribuídos. Alguns destes arcabouços, como o Integrade [1], tem como característica permitir a execução de aplicações paralelas fortemente acopladas, onde a otimização de comunicação é fator crucial de desempenho das aplicações. Neste artigo apresentamos um estudo sobre inferência de topologia física das redes locais que formam o arcabouço.*

1. Introdução

O objetivo deste trabalho é levantar informações sobre o estado da rede entre nós da grade para tentar inferir a topologia física da mesma, de forma a auxiliar a tomada de decisões de alocação de tarefas seguindo algum critério de otimização. Pretendemos descobrir, do jeito menos custoso e o mais prático possível, essa topologia física e o estado das ligações entre os nós da rede. Com esta informação em mãos, propor um algoritmo de escolha de nós que otimize a comunicação entre os mesmos

O objetivo do levantamento de topologia em rede local, apenas em nível 2 (segundo o modelo de camadas OSI), é descobrir nós da rede agrupados em um mesmo switch, bem como as ligações entre esses switches, para criar uma representação em forma de árvore dessa rede, e poder inferir a capacidade de comunicações entre os nós. Pode-se assumir a distribuição dos switches em estrutura de árvore, pois isso vem definido do protocolo 802.1D. Uma alternativa é coletar alguns dados através dos switches, o que nos dá uma intrusão bem menor na carga da rede, mais precisão, mas exige um acesso privilegiado a esses switches, o que pode necessitar de permissão ou ajuda do administrador da rede. Essa coleta pode ser feita através do protocolo SNMP. O modo de acesso aos dados desse protocolo está padronizado no próprio padrão IEEE 802.3.

Existem ferramentas proprietárias que realizam monitoramento de redes locais baseadas em coletas de dados SNMP, para efeito de administração de rede, mas voltadas para visualização de gráficos e configuração dos ativos de rede. Praticamente todo fabricante de ativos dispõe de algum software próprio para realização dessa tarefa. Muitos deles usam protocolos proprietários em seus switches para coleta desses dados. De

qualquer forma, não se encontram no geral interfaces para acesso aos dados coletados nos equipamentos, e o fato de serem soluções proprietárias não é compatível com os objetivos deste trabalho.

2. Descobrimento da topologia física baseado em coleta de dados por SNMP

O ponto central do algoritmo será baseado nas informações das tabelas de endereçamento MAC dos switches, que contêm a informação de por qual porta é acessível determinado MAC. O principal desafio do algoritmo é como "casar", ou sobrepor, as informações extraídas de vários switches de forma a identificar os que estão interligados e obter uma topologia global que inclua todos eles.

Primeiramente descobrimos, dado um domínio administrativo, quais ativos são switches e quais são roteadores. Através de uma aplicação do Teorema da conexão direta [2], determinam-se as portas dos switches que estão conectadas umas às outras. Desenvolveu-se uma regra para tratar casos de ativos sem acesso ou suporte SNMP, ou switches interligados por segmentos compartilhados. Definiremos como raiz o switch que tem o melhor equilíbrio no número de hosts nas subárvores formadas por cada uma de suas portas, e através de uma recursão montamos a topologia da rede vendo como cada switch é alcançado a partir da sua raiz.

Esse modelo já é usado por um sistema em estado experimental (de onde foram retiradas as implementações de algoritmos mostrados nesse artigo), que já faz a coleta de dados, e mostra a topologia visualmente. O aperfeiçoamento deste sistema leverá à disponibilização de um serviço para consulta de topologia e a implementação de um algoritmo para determinação do melhor conjunto de nós disponíveis para execução de uma tarefa que precisa de muita comunicação entre os nós.

3. Conclusão

Descrevemos neste artigo a nossa pesquisa sobre o que tem sido feito na área de levantamento de topologia de redes, com enfoque em redes locais. Nosso objetivo foi obter resultados que dessem suporte a infra estrutura de grades.

Referências

[1] *InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines.* Andrei Goldchleger, Fabio Kon, Alfredo Goldman, Marcelo Finger, and Germano Capistrano Bezerra. Concurrency and Computation: *Practice & Experience*. Vol. 16, pp. 449-459. March, 2004.

[2] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri and A. Silberschatz. "Topology Discovery in Heterogeneous IP Networks"In Proceedings of IEEE INFOCOM, 2000.

Gerenciando Reservas de Recursos de Grade

Leonardo Kunrath[1], **Carlos Becker Westphall**[1]

[1]Dep. de Informática e Estatística– Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brazil

{leonardo,westphal}@lrg.ufsc.br

Abstract. *In this work, we discuss about how our Grid resource reservation manager addresses four features: efficience, resource utilization, flexibility and abrangence.*

Resumo. *Neste trabalho, discutimos sobre a forma com que nosso gerenciador de reservas de recursos de Grade provê as características de eficiência, boa utilização dos recursos, flexibilidade e abrangência.*

1. Introdução

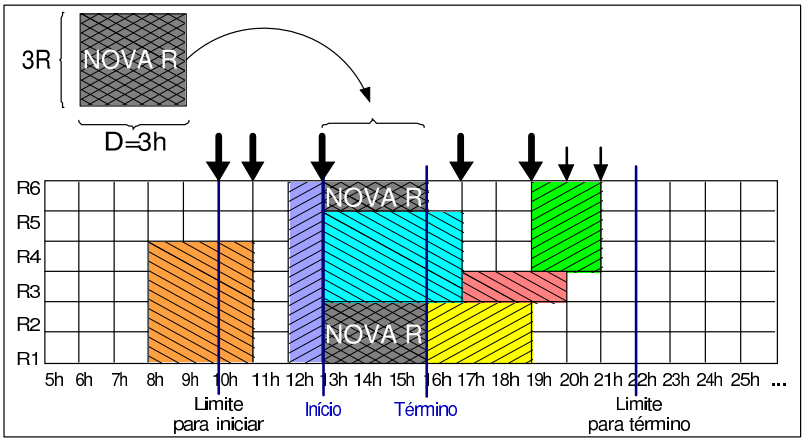
Reservas de recursos são fundamentais para garantir qualidade de serviço em Grades. Elas permitem que determinados recursos se tornem exclusivos para aplicações ou usuários, de forma que se possa utilizar o potencial dos recursos ao máximo. Por isso, os gerenciadores de reservas de recursos devem ser capazes de fazer boa utilização dos recursos, devem ser eficientes, devem abranger vários tipos de recursos, devem ser flexíveis para suportar reservas imediatas, antecipadas ou sob-demanda. Este trabalho traz uma melhoria significativa na questão de abrangência em relação ao nosso trabalho anterior [Kunrath et al. 2008], que suportava apenas reservas parciais de um recurso; enquanto este apresenta suporte a reservas coordenadas de recursos idênticos.

2. Estratégias para Gerenciar Reservas

Um gerenciador de reservas atua no sentido de controlar o uso dos recursos através de reservas, e possui duas funções principais: negociação e escalonamento. O escalonamento é o núcleo da gerência de reservas e nele são escolhidos os recursos e o período mais adequado dentre os possíveis para que a reserva aconteça. Seus requisitos fundamentais são:

- **Eficiência** - é preciso que a complexidade do algoritmo seja baixa, tanto em relação ao número de recursos gerenciados quanto ao número de reservas já agendadas. Nossa proposta é o uso de um algoritmo *on-line* de *empacotamento em fita* para abstrair as reservas, que são representadas por retângulos que devem ser colocados sobre uma fita, que representa os recursos.
- **Utilização** - é preciso que o escalonador preze pela máxima utilização dos recursos. A utilização em um determinado momento pode ser medida como a capacidade utilizada em relação à capacidade total. Nosso escalonador apresenta uma solução eficiente por usar um algoritmo linear, que percorre a lista de reservas até encontrar uma solução adequada.

- **Flexibilidade** - o escalonador deve ser capaz de fazer o agendamento de reservas imediatas, antecipadas e sob-demanda. Isso é obtido pela definição de reserva utilizada, com os atributos de **limite para iniciar**, **limite para término** e **duração**; que são enviados na requisição da reserva; que são utilizados pelo escalonador para escolher os valores de **início** e **término** da reserva.
- **Abrangência** - o escalonador deve suportar reservas de diferentes tipos de recursos, utilizando reservas adequadas para cada tipo. Nossa proposta permite que sejam feitas reservas parciais de um recurso (tal como reservar uma certa quantidade de banda de rede), reservas de um recurso por inteiro ou reserva de vários recursos (como reservas de X processadores para uma aplicação paralela).



A figura mostra um exemplo de utilização do algoritmo desenvolvido para reservar recursos idênticos. Os retângulos representam as reservas. A idéia é que o algoritmo deve encontrar um local apropriado para encaixar o novo retângulo (nova reserva), ou seja, achar o período de tempo (horizontal) e os recursos para a reserva (vertical). Na figura, chega uma requisição de reserva de três recursos, por três horas; o escalonador percorre a fita e encontra, dentro dos limites de tempo, um período no qual a reserva é efetuada.

Os testes realizados apresentaram como resultado, para reserva de 16 recursos idênticos, utilização média de aproximadamente 80% com dez reservas agendadas, 90% com cem e 95% com mil reservas ou mais. Para 4 recursos, no entanto, estes valores foram menores, alcançando apenas 70%, 87% e 93%, com dez, cem e mil reservas agendadas, respectivamente. A etapa de escalonamento, isoladamente, apresentou tempo de resposta inferior a um segundo para mil reservas agendadas, menor do que três segundos para duas mil, e menor que seis segundos para três mil.

3. Conclusões

Foram propostas estratégias para que se tenha um gerenciador de reservas de recursos eficiente, flexível, abrangente e com boa utilização dos recursos. Este trabalho também aumentou a abrangência do anterior [Kunrath et al. 2008] por permitir reservas de recursos idênticos.

References

Kunrath, L., Westphall, C. B., and Koch, F. L. (2008). Towards advance reservation in large-scale Grids. To appear in *The Proceedings of The Third International Conference on Systems (ICONS 2008)*. IEEE Computer Society Press.

Políticas e Práticas de Certificação para Grades

Luiz M. R. Gadelha Jr. , Fábio Licht , Vívian Medeiros , Bruno Schulze

[1]Laboratório Nacional de Computação Científica
Av. Getúlio Vargas, 333 – Quitandinha
25.651-075 – Petrópolis – RJ – Brasil

{lgadelha,licht,vivian,schulze}@lncc.br

Abstract. *To become operational, a certificate authority must define its policies and practices, which can be used by relying parties to assert its trustworthyness. In this work it is done a survey of the current practices and policies employed by grid certificate authorities. In particular, some experiences in setting up grid certification authorities.*

Resumo. *Para se tornar operacional, uma autoridade certificadora deve definir suas práticas e políticas, que podem ser utilizadas por partes confiantes para avaliar seu nível de confiança. Neste artigo, é realizado um levantamento das atuais práticas e políticas empregadas por autoridades certificadoras para grades. Em particular, são descritas algumas experiências de implantação de autoridades certificadoras para grades.*

1. Introdução

Uma autoridade certificadora (AC) deve determinar suas práticas e políticas para que partes confiantes avaliem seu nível de segurança. Existe uma preocupação de que as práticas e políticas adotadas pelas ACs de grades permitam a interoperabilidade entre as mesmas. A *International Grid Trust Federation* (IGTF) é um fórum onde este problema é discutido, nele são estabelecidas políticas e recomendações para os provedores de serviços de certificação digital para grades. Estas ACs foram classificadas de acordo com suas práticas e políticas. As *ACs clássicas* normalmente não possuem informações prévias sobre os solicitantes de certificados, requerendo uma validação mais rigorosa. No Brasil, a *BrGrid CA* [Rebello 2006] é uma AC clássica credenciada à IGTF. As *ACs integradas a bases de usuários* possuem bases de dados sobre os solicitantes de certificados que podem ser utilizadas para validar a identidade dos mesmos. Estas ACs normalmente atuam no escopo de uma instituição e atendem a um público bem definido. Esta característica facilita a emissão do certificado pois os usuários precisam apenas fornecer dados presentes na base para se validar. As *ACs para certificados de curta duração* utilizam mecanismos de autenticação pré-existentes para validar a identidade dos solicitantes de certificados e emitem certificados com prazo de validade de aproximadamente 12 dias. A seguir, são descritas algumas experiências de implantação de ACs para grades.

2. Sistema Nacional de Processamento de Alto Desempenho

O Sistema Nacional de Processamento de Alto Desempenho (SINAPAD) iniciou em 2005 a implantação de um ambiente de computação em grade, baseado no *middleware* Globus. Foi implantada uma AC [Gadelha 2006] para a grade do SINAPAD, cuja solução é baseada no software livre OpenCA, seguindo diversas recomendações de segurança típicas

para este tipo de serviço. Foram adotadas práticas e políticas características de uma AC integrada a bases de usuários. As requisições são geradas por uma ferramenta desenvolvida pelo SINAPAD e enviadas por e-mail para a AC. Os operadores da AC validam as requisições utilizando uma base de dados que contém informações de usuários pré-validados pelo sistema de admissão. Os certificados são emitidos pela AC e enviados aos usuários por e-mail, que são responsáveis pelo gerenciamento dos mesmos.

2.1. Virtual Community Grid

Uma solução desenvolvida pelo VCG (Virtual Community Grid) [Licht et al. 2006] foi a criação de uma AC dinâmica baseada na atribuição de direitos de geração de certificados temporários de host para usuários pré-cadastrados. Um usuário faz um cadastro em um portal, de posse desse cadastro o administrador permite ou não a adesão deste usuário. Uma vez associado ao portal, esse usuário recebe um hash que o identificará. Quando esse usuário desejar adicionar um determinado host à grade, a chave privada e a requisição de assinatura do certificado serão geradas, enviadas à AC, assinada por tempo determinado pelo administrador da grade e reenviado ao host do solicitante. O hash será solicitado e a associação deste host ao usuário será feita, cadastrando no banco de dados todas as máquinas deste usuário. Qualquer problema relacionado ao uso indevido por parte desse usuário através desse host poderá ser passível de punição estipulada pela política de segurança da grade. Todo o funcionamento dessa estrutura é automatizado, sem a necessidade de um gerente da AC para assinar o certificado dos hosts que forem sendo incluídos na grade, tornando essa estrutura mais simples de ser utilizada por pessoas leigas quanto à arquitetura e funcionamento da grade.

3. Conclusão

Foi descrita a experiência dos autores com a implantação de ACs para grades com características diferentes. A grade do SINAPAD tem um perfil de provedora de serviços, onde o conjunto de recursos muda com menos freqüência. A VCG está mais voltada a trabalhos colaborativos e tem um conjunto de recursos mais dinâmico. Em cada caso, foram adotadas práticas e políticas adequadas às características descritas. A Infra-estrutura de Chaves Públicas para Ensino e Pesquisa (ICPEDU) é uma iniciativa da RNP para incentivar a utilização de certificados digitais por instituições acadêmicas, permitindo que elas implantem suas próprias ACs. Como os *middlewares* para grades permitem confiar em diversas ACs simultaneamente, é interessante que sejam aceitos certificados que permitam interoperabilidade com outras grades e que sejam de conveniente obtenção, como é o caso de certificados emitidos por ACs vinculadas à ICPEDU e IGTF.

Referências

Gadelha, L. (2006). SINAPAD Grid PKI. *Slides*. 1st Meeting of The Americas Grid Policy Management Authority.

Licht, F., Schulze, B., and Ishikawa, E. (2006). Gerência e Fornecimento de Certificados de Confiança para Nós Móveis Usando Certificados de Curta Duração. In *IV Workshop de Computação em Grid e Aplicações (WCGA)*, pages 151–152.

Rebello, V. (2006). The Brazilian Grid Certification Authority (BrGrid CA). *Slides*. 1st Meeting of The Americas Grid Policy Management Authority.

Índice por Autores

Promoção

Patrono

Realização

Patrocínio

Apoio

ISBN 857669176-0