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

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

[1]Departamento de Ciência da Computação – IME/USP
[2]Departamento de Ciência da Computação – UFMA

USP - Universidade de São Paulo

IME - Instituto de Matemática e Estatistica

InteGrade

A VIDA É COMBATE

Roteiro

- Motivação e Objetivo
- Desafios e Abordagem Inicial
- Integrade
- MAG
- Tolerância a Falhas no MAG
- Avaliação de Desempenho
- Resultados
- Trabalhos Correlatos
- Conclusão e Perspectivas

Movitação e Objetivo

- Aplicações longas, entre elas simuladores, aplicações CAD e de processamento de sinais, requerem a alocação de recursos por um longo período de tempo
- Grades computacionais são alternativas atraentes para execução de aplicações que demandam alto poder computacional
- Prover tolerância a falhas em grades oportunistas como forma de otimizar/viabilizar a execução de aplicações longas nesses ambientes

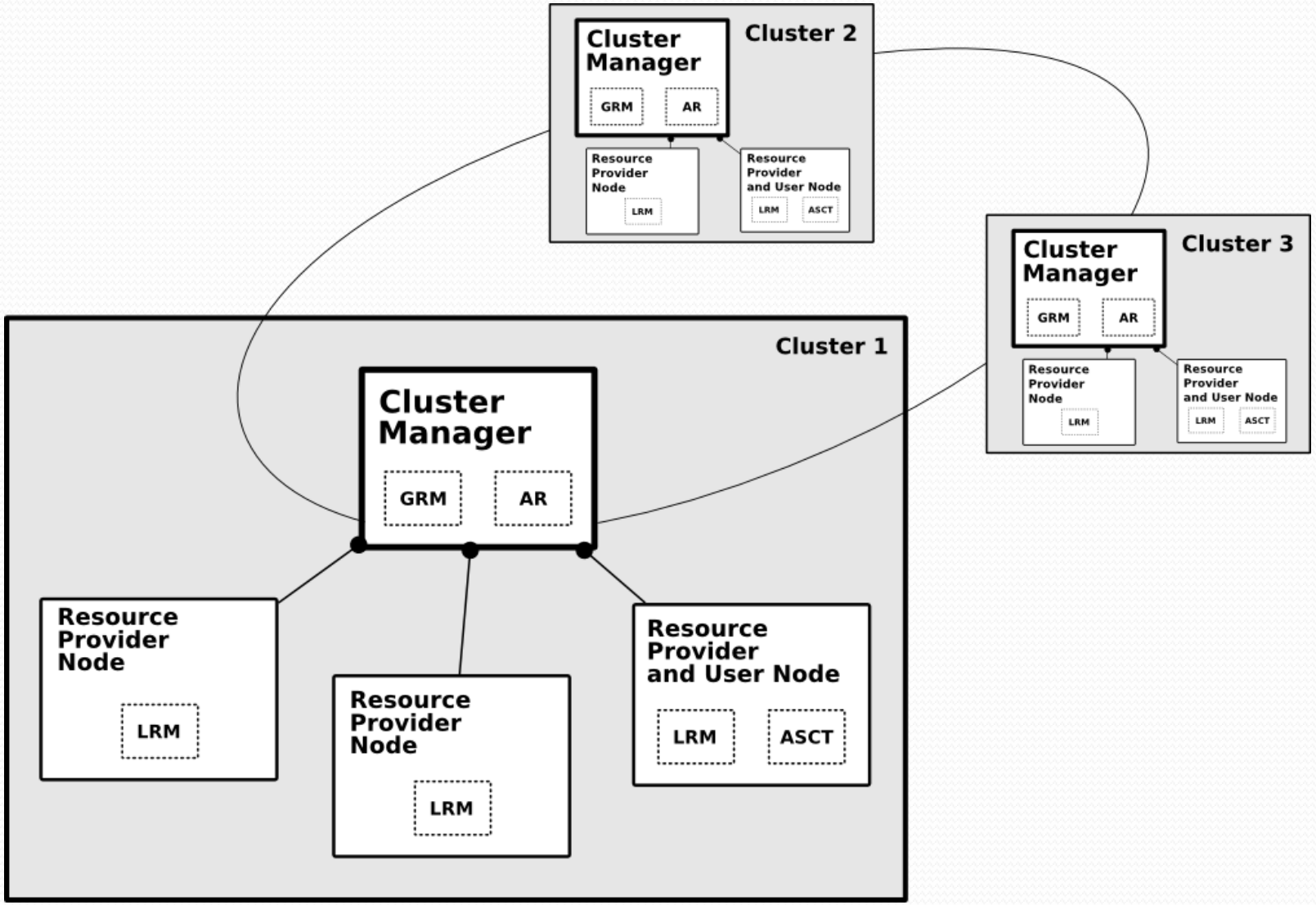
Desafios e Abordagem Inicial

- Desafios de um Middleware de Grade
  - Dispersão e heterogeneidade dos recursos, escalonamento, tolerância a falhas, segurança
- Agentes Móveis
  - Cooperação, autonomia, heterogeneidade, reatividade, mobilidade, proteção e segurança
  - Mecanismos de tolerância a falhas

Integrade

- Grade Oportunista que aproveita ociosidade dos recursos
  - Transparência aos usuários das máquinas compartilhadas
  - Otimizar a utilização dos recursos ociosos
- Componentes
  - Global Resource Manager (GRM)
  - Local Resource Manager (LRM)
  - Application Repository (AR)
  - Application Submission and Control Tool (ASCT)
  - Local User Pattern Analyser (LUPA)

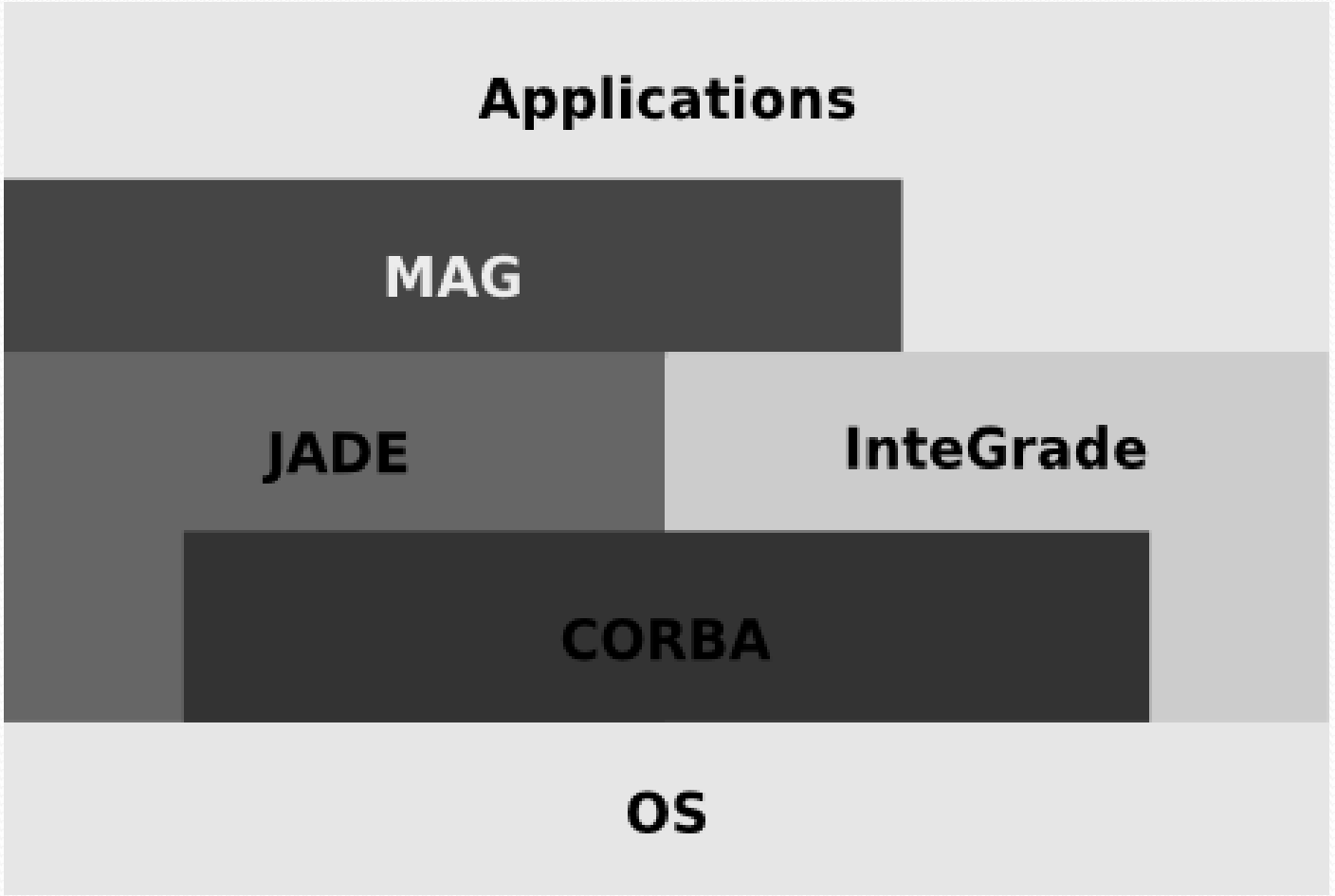
Arquitetura do Integrade



MAG

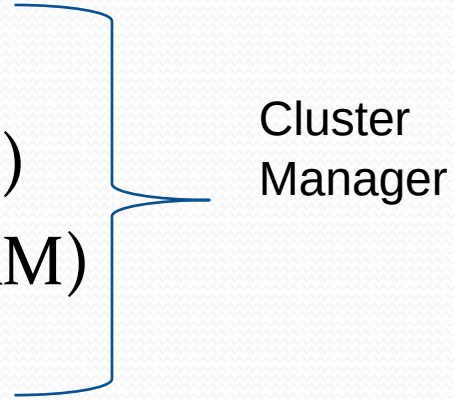
- Mobile Agents for Grid Computing Environments
  - Encapsulamento de aplicações Java regulares e paramétricas em agentes móveis
  - Aplicações devem fazer extensão da classe *MagApplication*
  - Classe *MagAgent* instancia *MagApplication* e controla seu ciclo de vida
- JADE: Plataforma de execução de agentes móveis
  - Serviços: comunicação, migração, monitoramento do ciclo de vida dos agentes

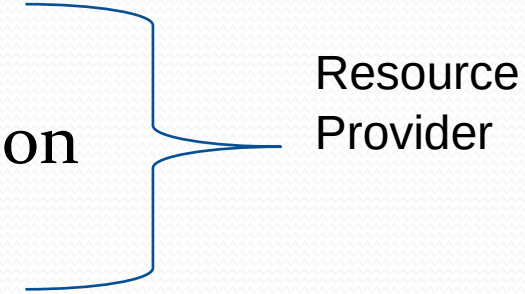
MAG



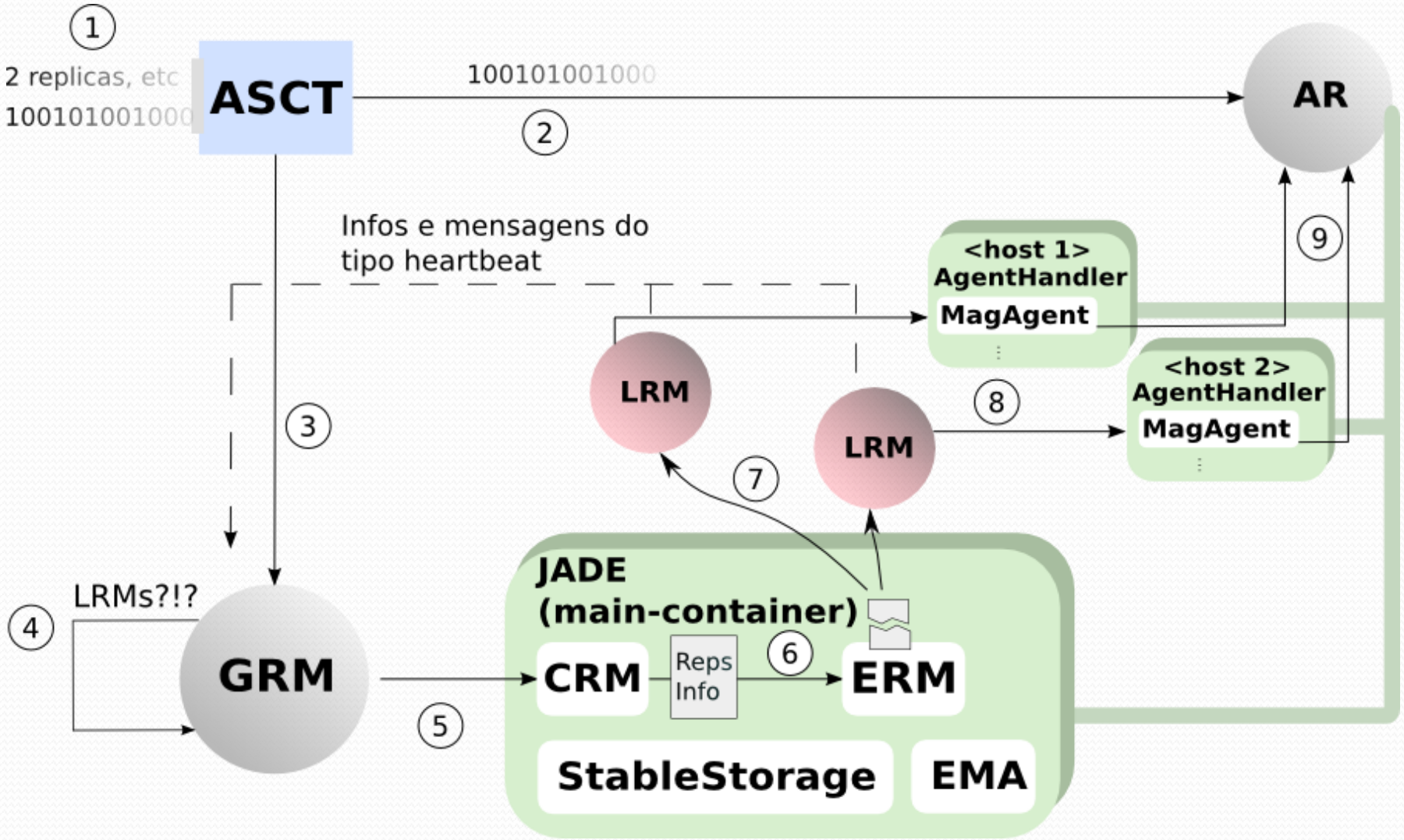
MAG: componentes

- Integrade + outros componentes
  - ExecutionManagerAgent (EMA)
  - ClusterReplicationManagerAgent (CRM)
  - ExecutionReplicationManagerAgent(ERM)
  - Stable Storage
  - AgentHandler
  - MagAgent e MagApplication
  - AgentRecover





Executando aplicações no MAG



Tolerância a Falhas no MAG

- Estratégias
  - Reenvio: aplicação é submetida novamente em caso de falha
  - Checkpointing: armazenamento periódico do estado de execução no StableStorage
  - Replicação: várias réplicas submetidas ao mesmo tempo em recursos distintos
  - Checkpointing com replicação: cada réplica salva seu estado de execução periodicamente

Tolerância a Falhas no MAG

- Tratamento de Falhas
  - Falhas no nível da aplicação
  - Realiza detecção de falhas de colapso (*crash*) nos nós
  - *runtimeException* é capturada através do método *uncaughtException (MagAgent extends ThreadGroup)*
  - *AgentRecover* solicita que outro *AgentHandler* retome a execução (reenvio)

Tolerância a Falhas no MAG

- Checkpointing
  - Obtido pela instrumentação do código binário
  - Arcabouço Brakes e MAG/Brakes
  - Captura estado de execução das threads após chamada dos métodos, respeitando um tempo mínimo
  - É realizado remotamente no StableStorage

Avaliação de Desempenho

- Ambiente de testes:
  - 6 máquinas do LCPD, IME-USP conectadas por uma rede de 100Mbps (1 GRM e 6 LRM's)
  - 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

Avaliação de Desempenho

- Metodologia
  - Foi utilizado como exemplo de aplicação uma classe Java que faz concatenação sucessiva de cadeias de caracteres
  - Motivos: alto sobrecusto ao mecanismo de checkpoint, e uso intenso de memória
  - Convenção temporal (minuto=hora)
  - Tempo de execução livre de falhas (referência):

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

Avaliação de Desempenho

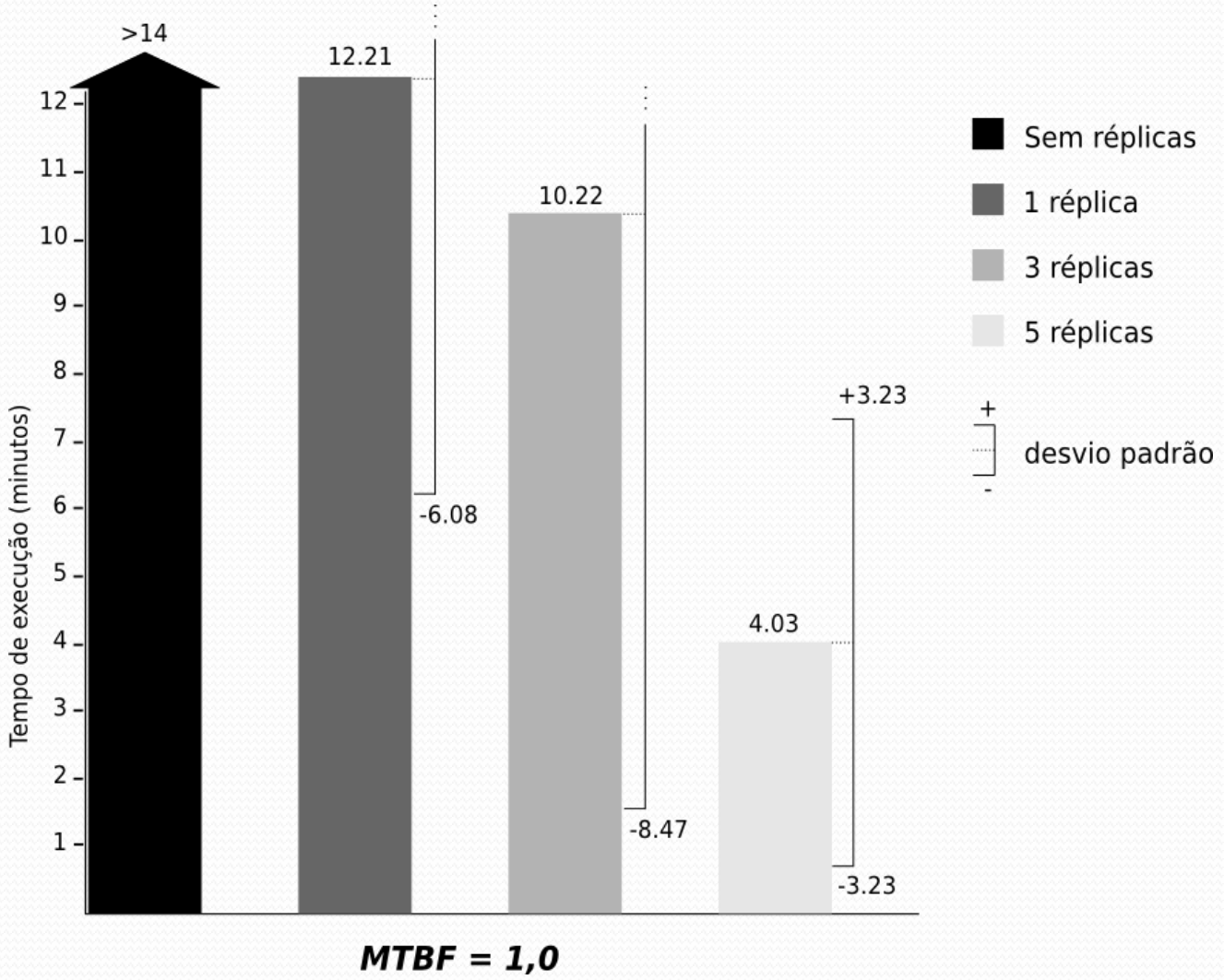
- Metodologia
  - Tempo Médio Entre Falhas (TMEF): média de tempo para que uma falha ocorra no ambiente de execução
  - Número de Réplicas: número de réplicas submetidas
  - TMEF: 0,5 (trinta segundos) e 1 (um minuto)
  - 0, 1, 3 e 5 réplicas em cada experimento
  - 30 execuções para cada configuração
    - Média aritmética do tempo de execução
    - Desvio padrão
  - O que vale é o tempo de execução da réplica que encerrou primeiro

Avaliação de Desempenho

- Modificações no ERM
  - Gerar falhas de colapso de acordo com o valor do TMEF
  - Comportamento probabilístico
  - A cada 10 segundos a probabilidade de gerar a falha é de 1/3 para TMEF = 30segs e de 1/6 para TMEF = 1min
- A cada falha gerada um nó é escolhido randomicamente para que suas computações sejam abruptamente encerradas

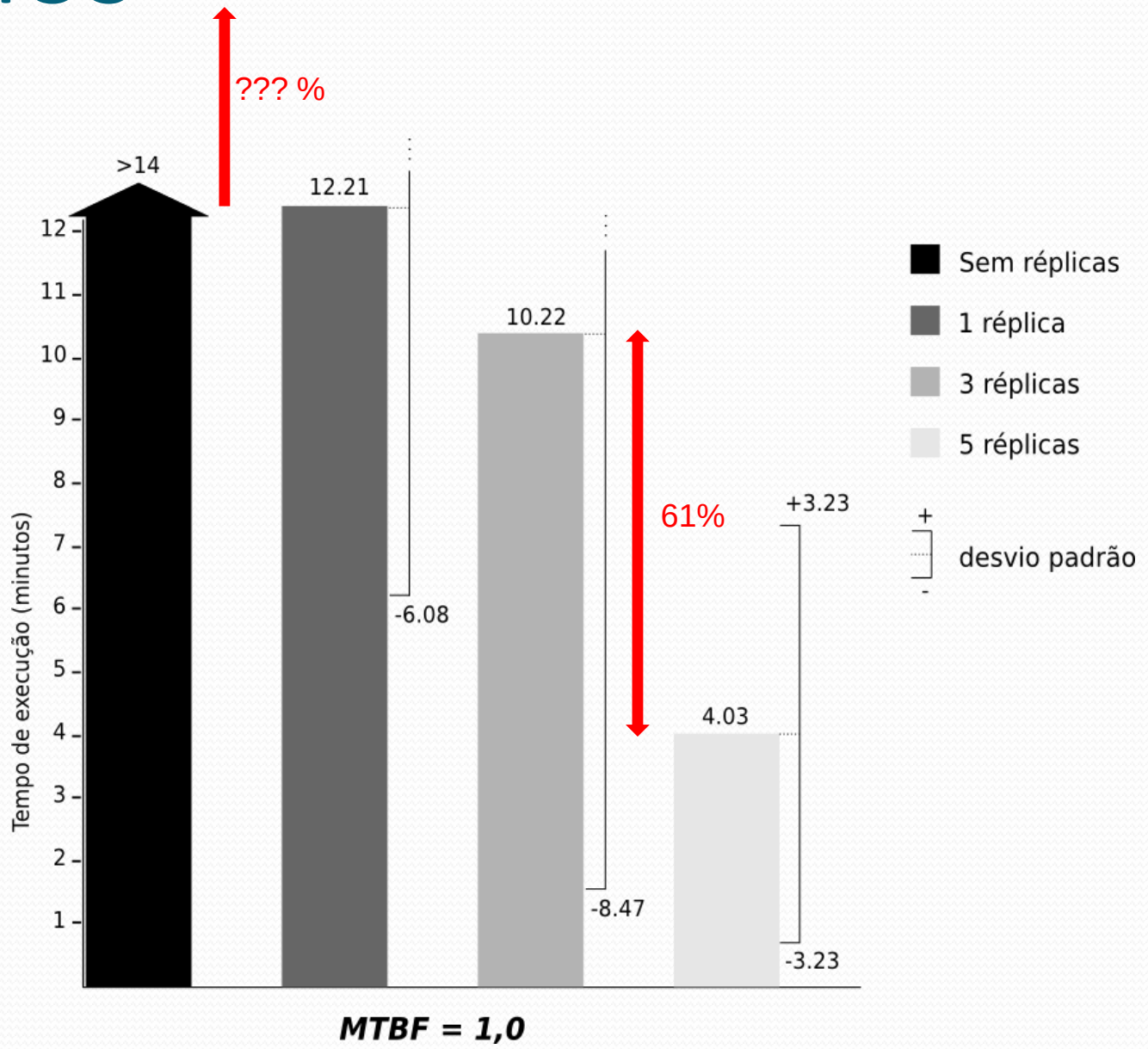
Resultados

- Replicação



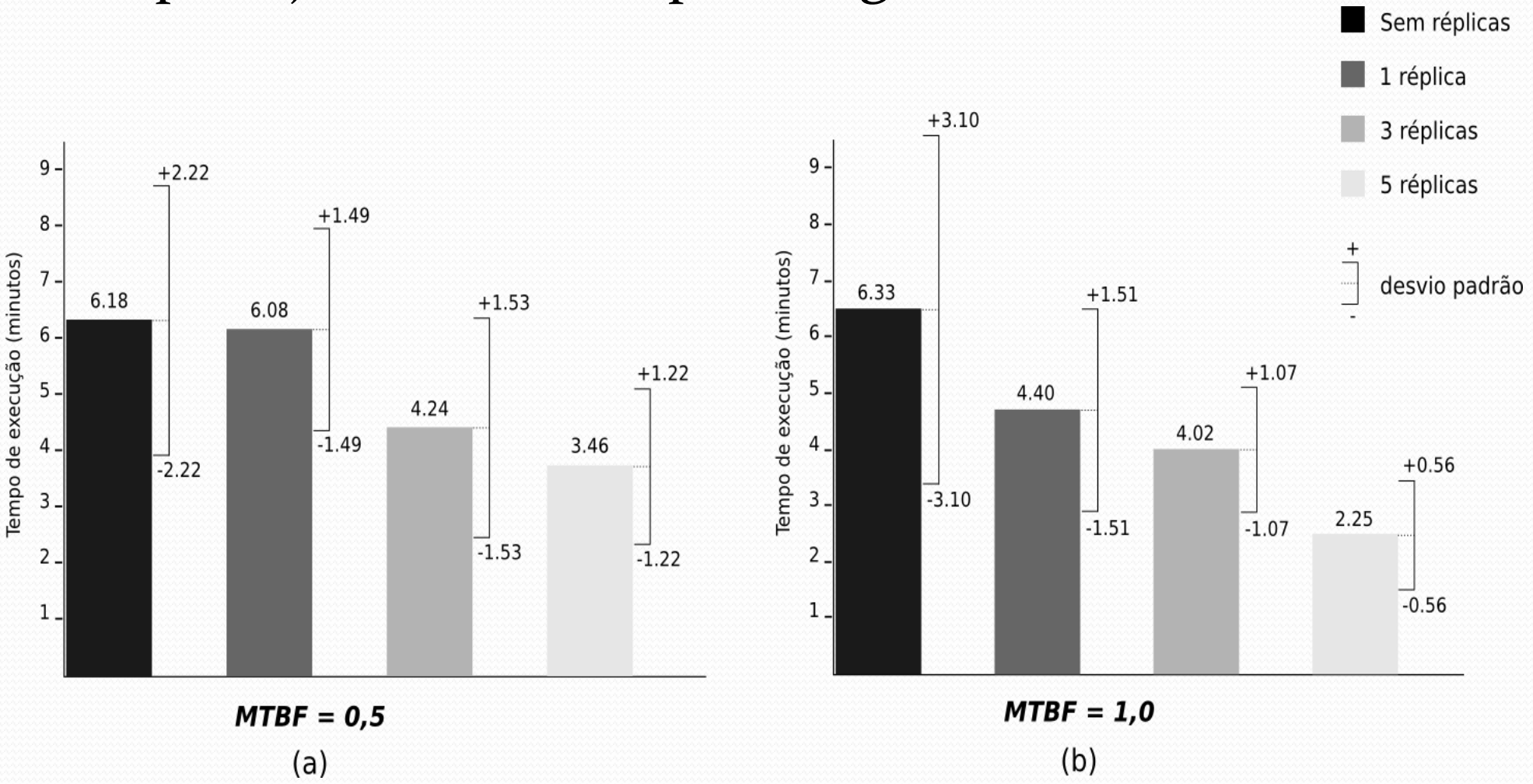
Resultados

- Replicação



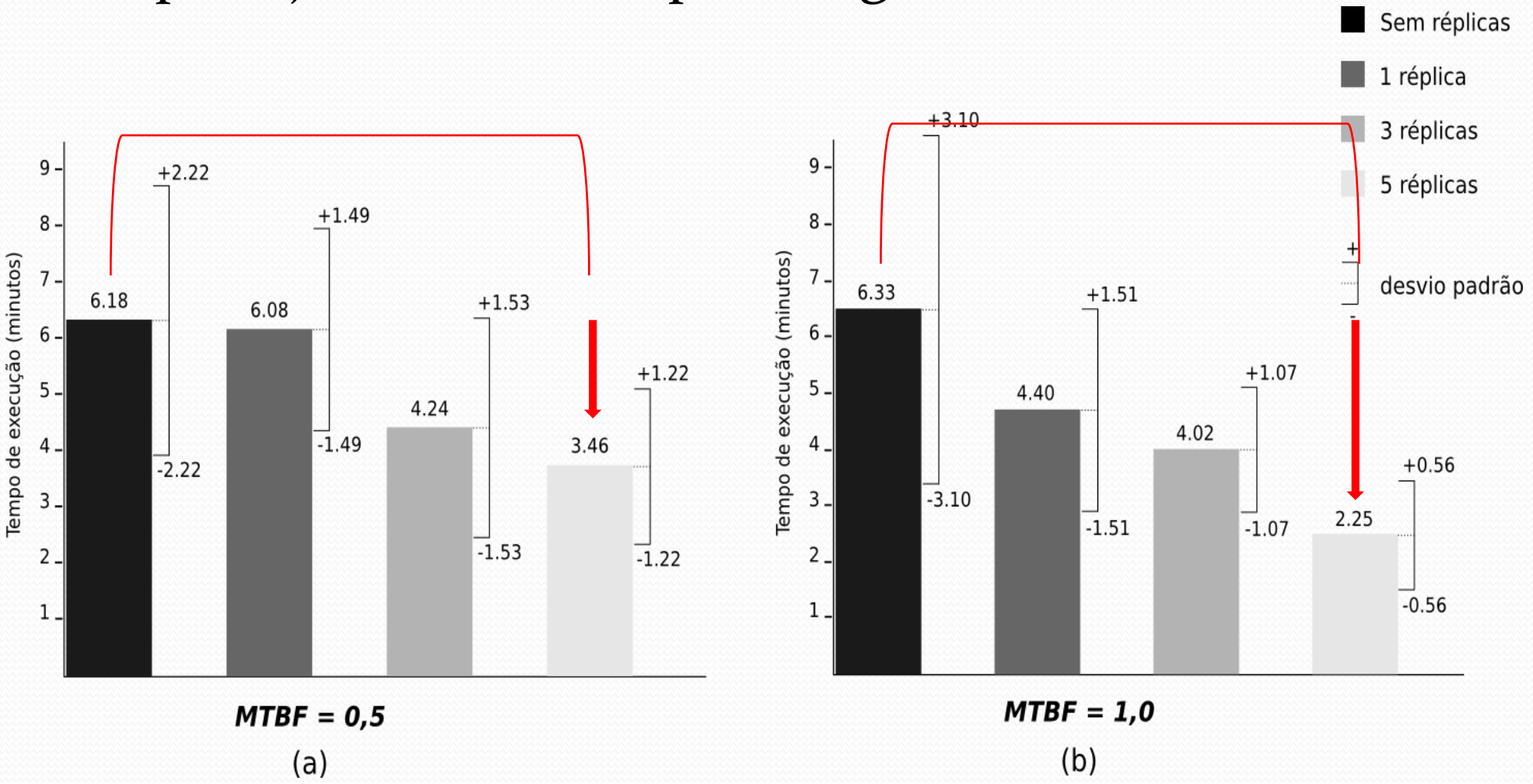
Resultados

- Replicação com Checkpointing



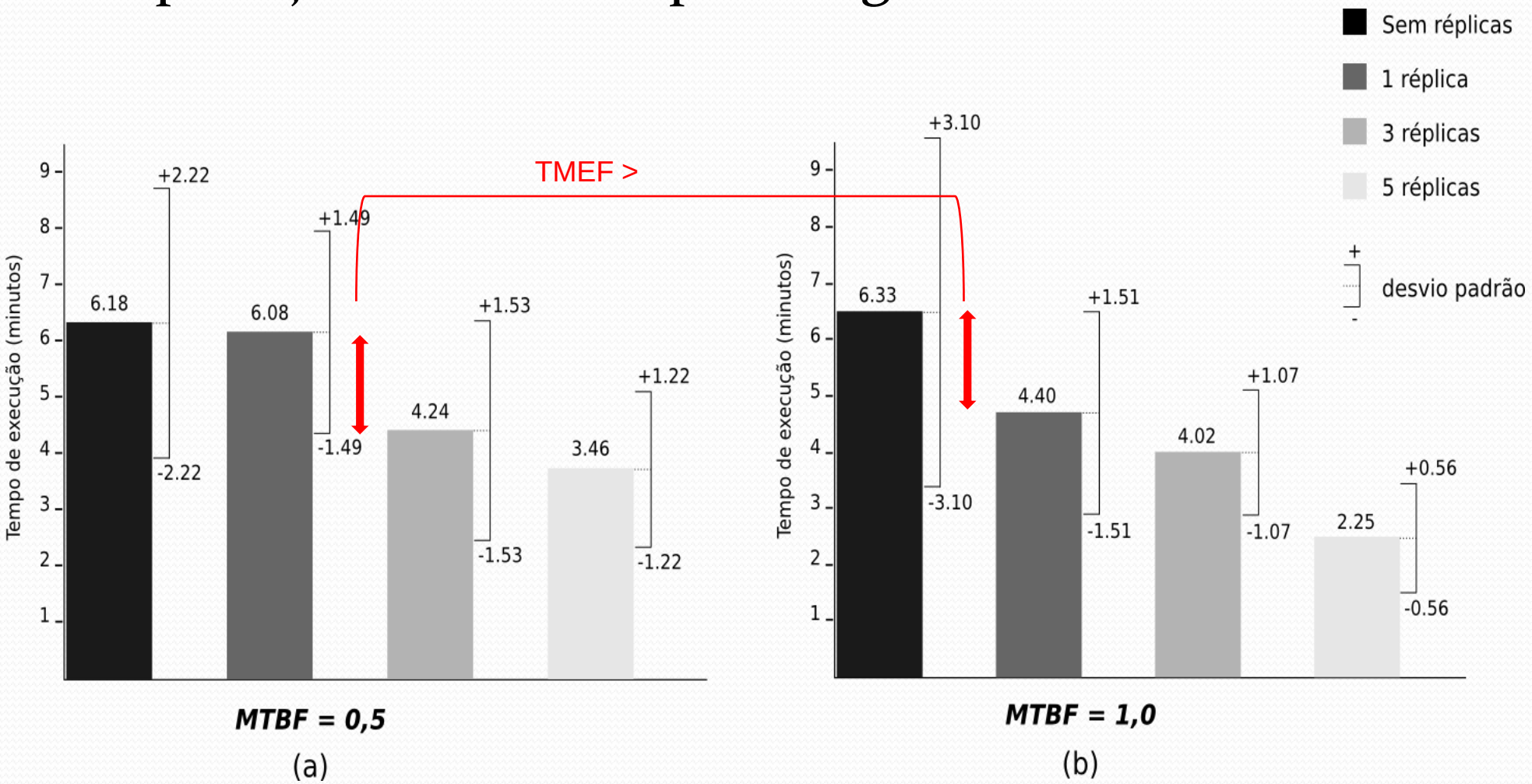
Resultados

- Replicação com Checkpointing



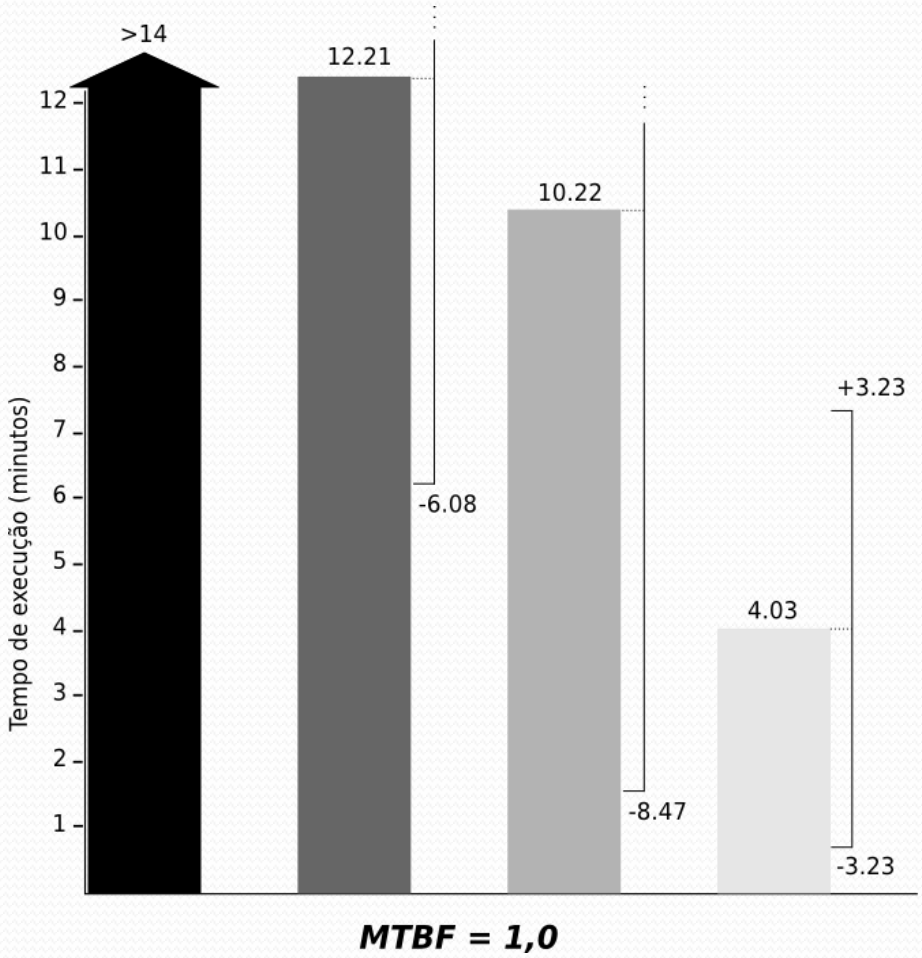
Resultados

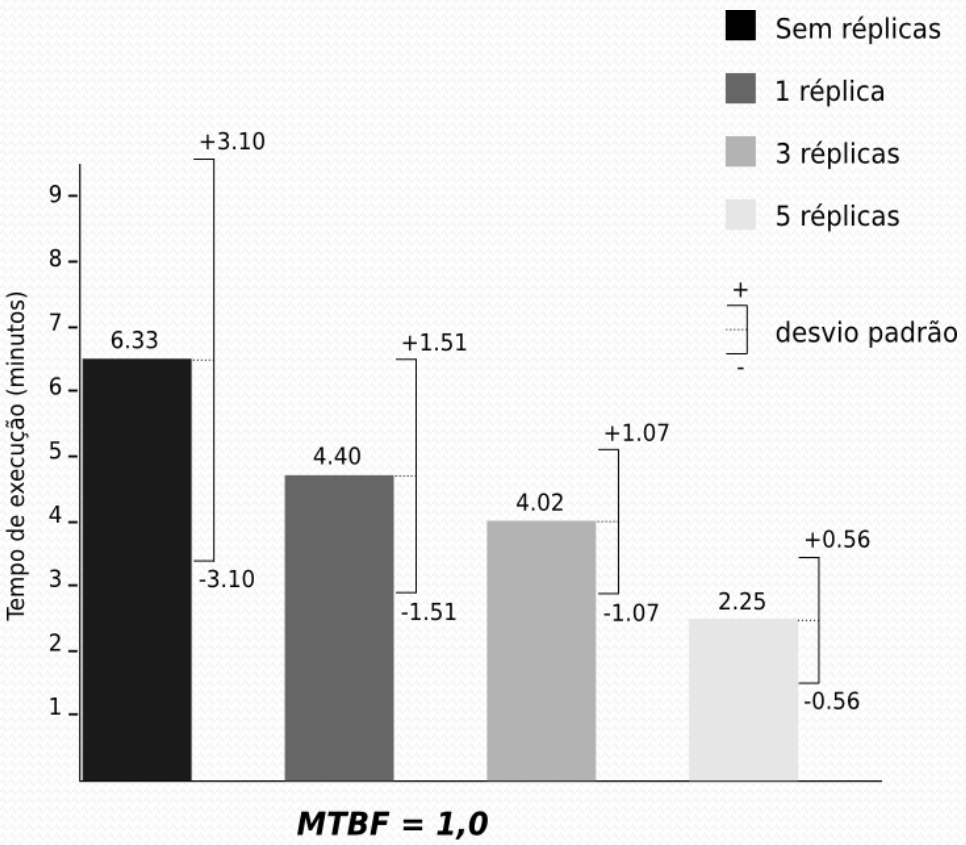
- Replicação com Checkpointing



Resultados

- Replicação X Replicação com Checkpointing

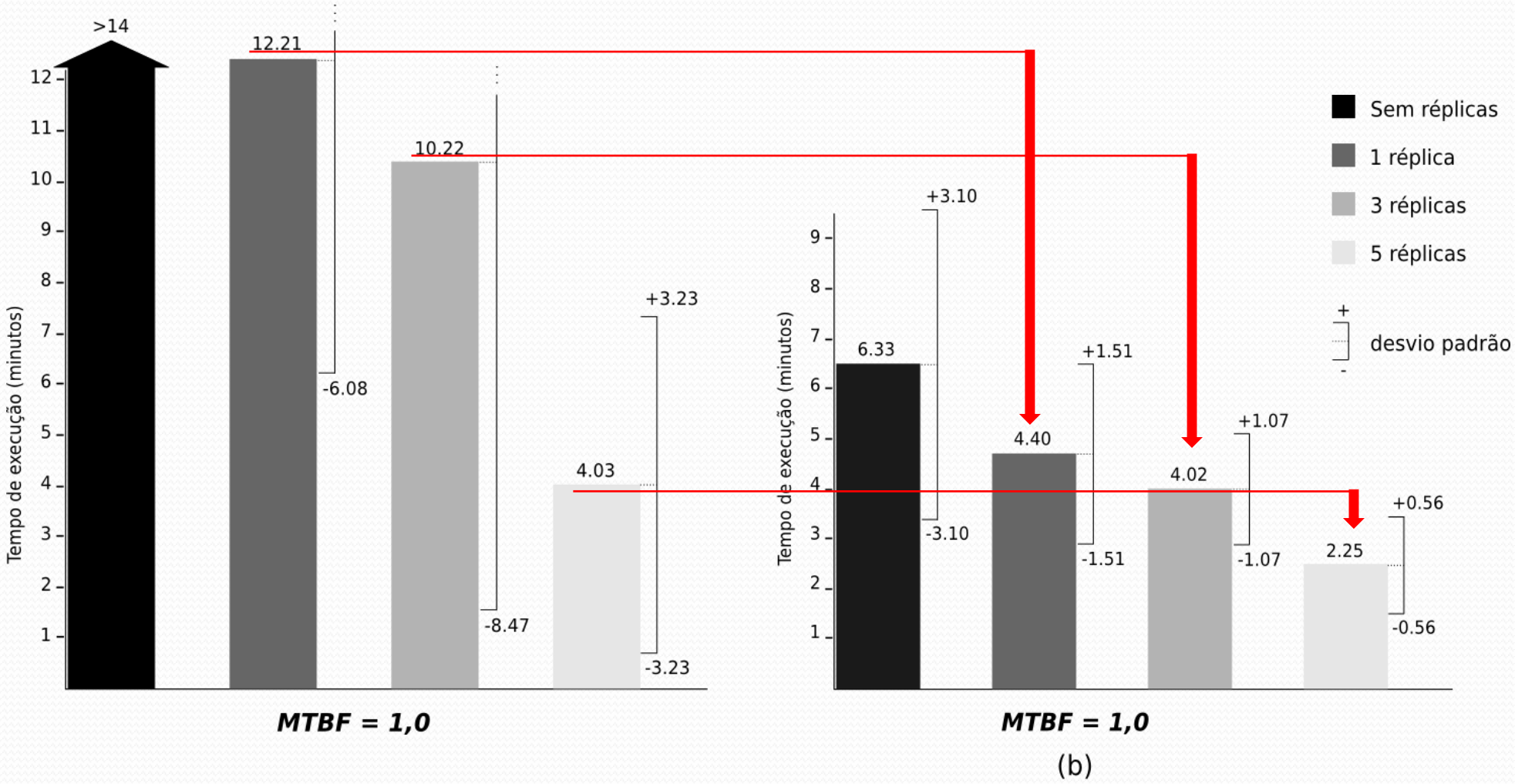




(b)

Resultados

- Replicação X Replicação com Checkpointing



(b)

Resultados

- Vantagem evidente na utilização de checkpointing
  - Evita perda da computação realizada
  - Baixa sobrecarga em nosso exemplo
- Resultados validaram as simulações realizadas em AutoGrid: Towards an Autonomic Grid Middleware [sallem et all. 2007]

Resultados

- Quanto menor o TMEF mais réplicas são necessárias para manter o tempo de execução abaixo de um tempo desejado
- Desvios padrões altos
  - 49% com replicação e checkpoint e 83% com replicação
  - Fatores: heterogeneidade e variação na disponibilidade dos recursos

Trabalhos Correlatos

- A flexible framework for fault tolerance in the grid [Hwang and Kesselman 2003]
- Uwagents: A mobile agent system optimized for grid computing [Fukuda and Smith 2006]
- Anthill: A framework for the development of agent-based peer-to-peer systems [Bagaoglu et al. 2002]
- The organic grid: selforganizing computation on a peer-to-peer network [Chakravarti el al. 2005]
- Labs of the world, unite!!! [Cirne et al. 2006]
- Framework for mobile agents on computer grid environments [Barbosa and Goldman 2004]
- Fault tolerance in a mobile agent based computational Grid [Lopes and Silva 2006]

Conclusões e Perspectivas

- Ajustes dinâmicos nos mecanismos de tolerância a falhas devem reduzir ainda mais o *makespan*
- O uso de checkpoints e de replicação, em conjunto, ajudam a obter uma estimativa de tempo mais precisa
- Futuramente devemos avaliar a substituição de réplicas como recurso de otimização

Obrigado!

Perguntas?

Vinicius Pinheiro
vinicius@ime.usp.br