Baile Project

Research line 3: Analysis of Choreography Service Allocation to Computing Nodes in a Grid or Cloud Environment

Emilio francesquini
Universidade de são Paulo
emilio@ime.usp.br

May, 2011

Motivations

- Grid and Cloud Environments
  - Distinct allocations lead to distinct execution times
    - Heterogeneous machines
    - Variations on the communication time between each machine pair
      - Network properties are dynamic
        - Location
        - Load
        - …
  - We intend to minimize the total execution time of a choreography

Possible techniques

- Migration, Duplication, and Preemption of Services and Choreographies
  - Lower, or even avoid if possible, the amount of communication needed
  - Lower the processing time in each and every machine taking part in the choreography enactment
- All these techniques would be based on measurable attributes of the whole system (e.g. free/committed CPU, memory, and network resources)

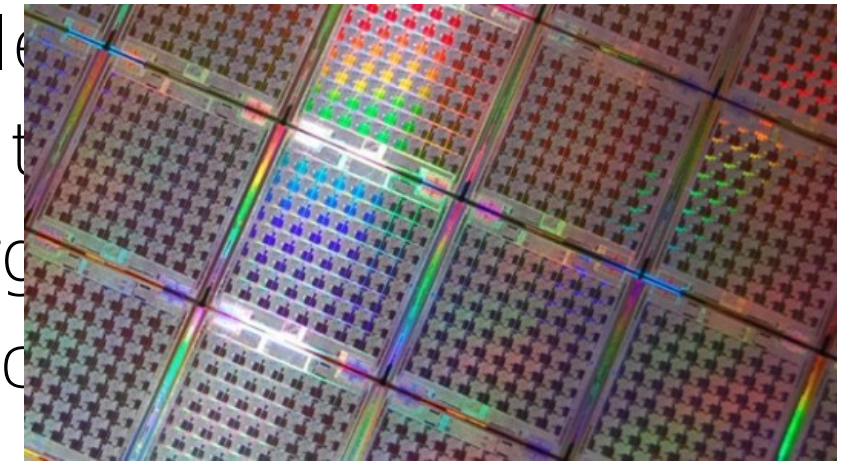
Meanwhile…

- There have been significant changes on the hardware context. Multi-core processor availability, for example
- An increasing share of t

Top 500 (www.top500.org

are endowed with multi-c

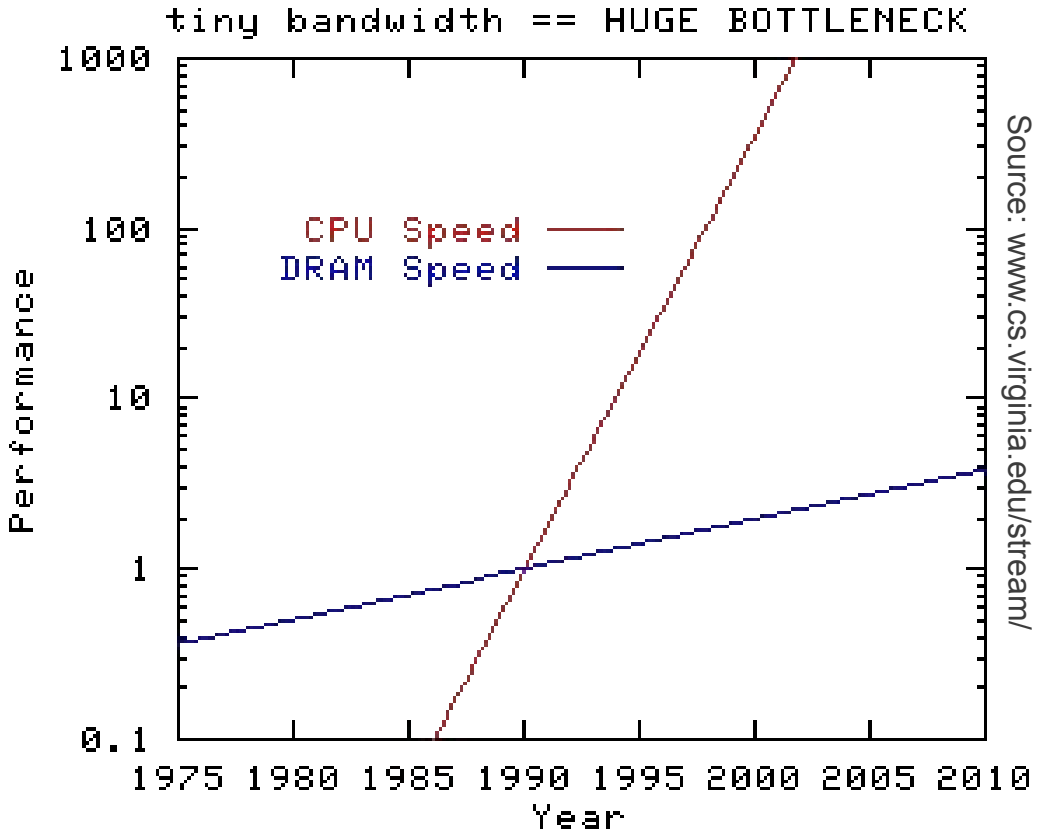
processors



Tilera Tile GX100 - 100-core general purpose processor

Why?

- Power Wall
- Memory Wall
- ILP Wall



Problems 1/2

- Currently existing software is not capable of profiting from the new hardware
- Most people find concurrent programming difficult
  - Apple GCD 

  - Actor Model [HBS73]

Problems 2/2

- Memory
  - Hierarchy
  - Bottlenecks
- Cache misses

Balle and Palermo [Balle2007], showed that a simple allocation change could lead to a speedup of up to 10.48% on a 16-core machine
Pousa et al. [Pousa08] showed, for a NUMA architecture, that a speedup of up to 31% could be achieved if the default Linux memory-page placement policy were changed

Summing Up

- A choreography can be seen as a parallel application composed of several tasks that communicate with each other
  - So to use a well known terminology we will not talk about the allocation of services to computers but the allocation of tasks to computing nodes
- To optimize the the performance of a large-scale choreography, we also need to optimize it locally
- Problems with the current solutions
  - They are not scalable (manual testing and choice of an allocation) for each application
  - There is no on-the-fly adaptation to changing environments/application behavior

Our Proposal

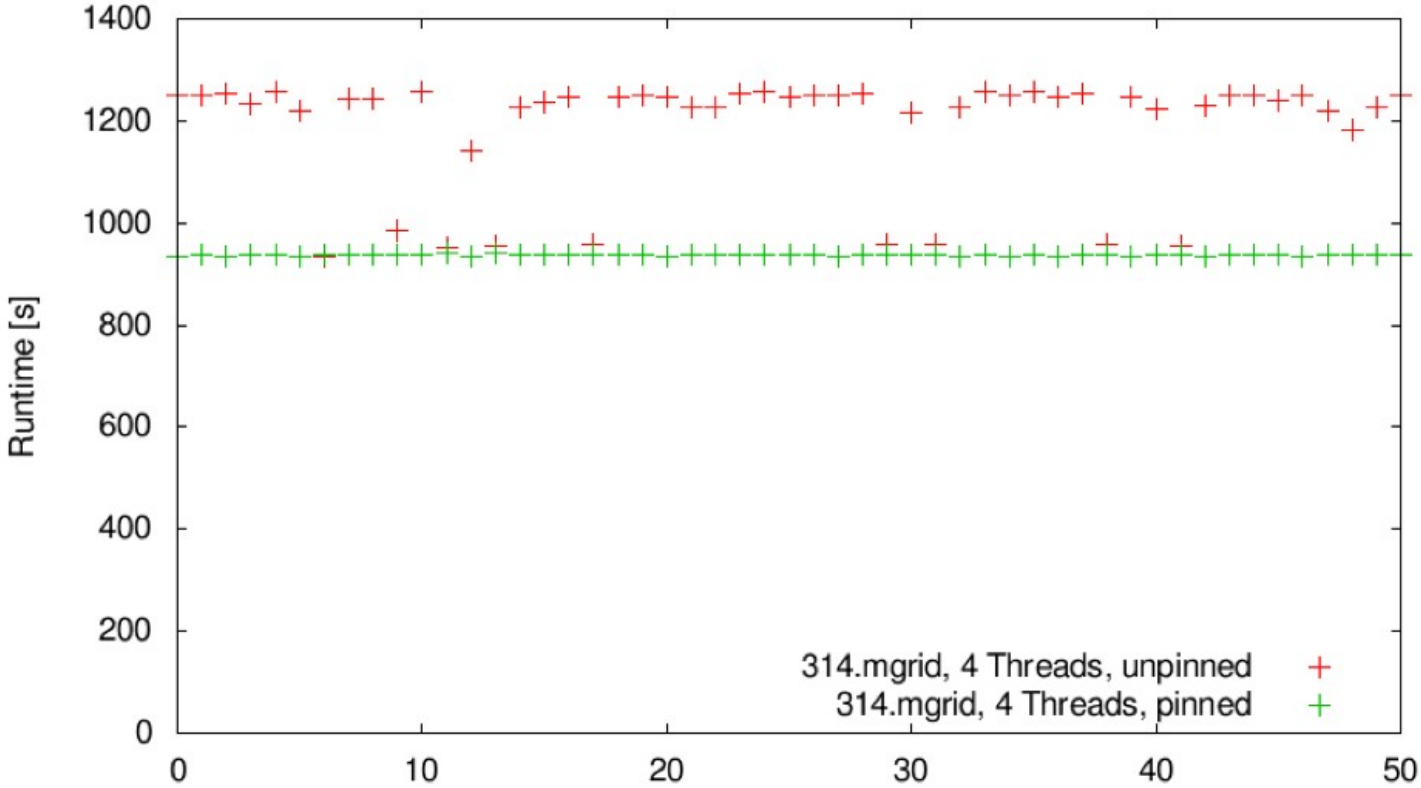
- Online and Offline Application and Environment Profiling
- Dynamic scheduling of tasks
- Dynamic placement of memory-pages
- Proceed gradually to a comprehensive solution
- For 2011 we propose
  - Study the current solutions (seek and point out their strengths and deficiencies)
  - Propose and implement a novel prototype scheduler that fixes the issues found on the current solutions

Two Approaches

- Top-down

- Bottom-up

Bottom-up

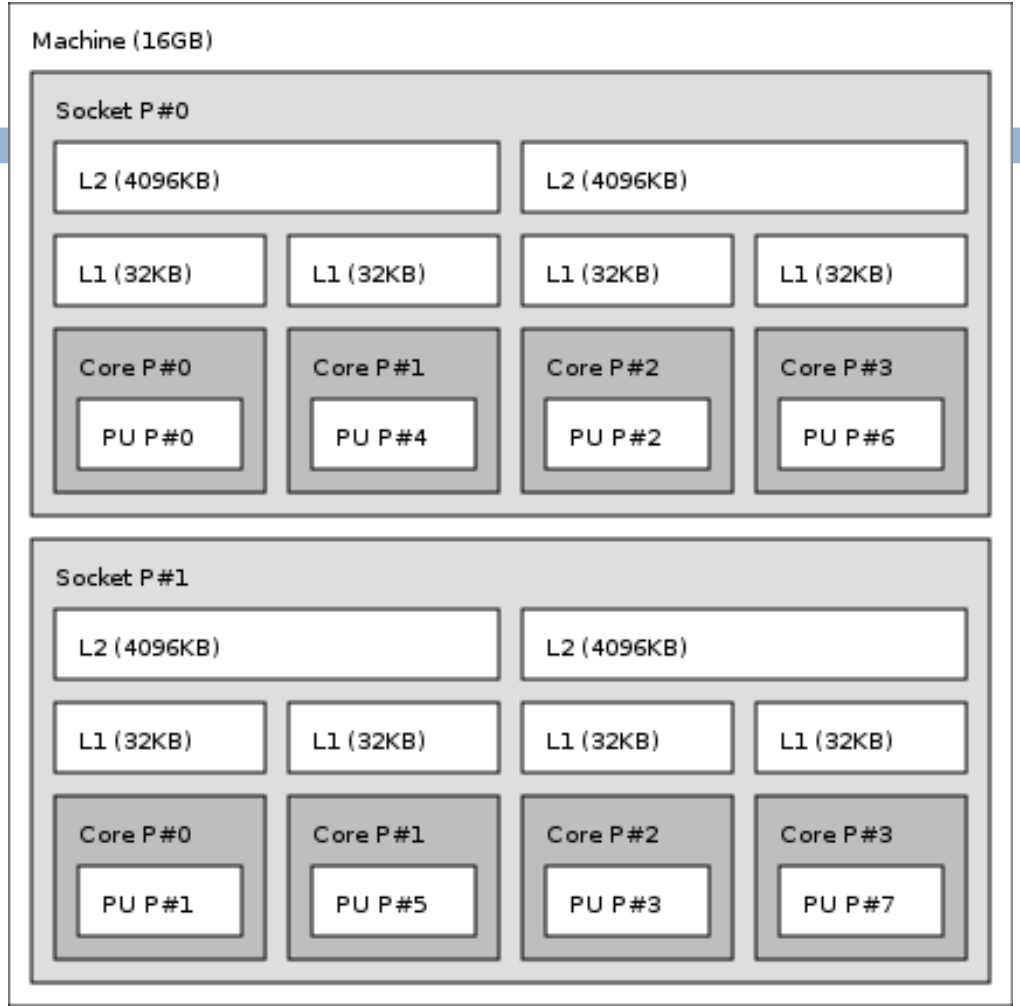

1400
1200
1000
800
600
400
200
0
Runtime [s]
0
10
20
30
40
50
314.mgrid, 4 Threads, unpinned
314.mgrid, 4 Threads, pinned
autopin [Klug08]

Bottom-up

- Hardware Performance Counters
  - Examples
    - Level 1/2/3 data/instruction cache misses
    - Cache Line Invalidation (SMP)
    - Data/Instruction translation lookaside buffer misses
    - Integer/FP instructions executed
    - FLOPS
    - Total cycles
    - MIPS
    - …

Bottom-up

hwloc



References

- [Balle2007] Enhancing an Open Source Resource Manager with Multi-Core/Multi-threaded Support, S. M. Balle and D. Palermo, Job Scheduling Strategies for Parallel Processing, 2007.
- [Pousa08]Christiane Pousa Ribeiro, Jean-Francois Mehaut, Alexandre Carissimi, Marcio Castro, and Luiz Gustavo Fernandes. Memory affinity for hierarchical shared memory multiprocessors. Computer Architecture and High Performance Computing, Symposium on, 0:59–66, 2009.
- [HBS73] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. 1973.
- [Klug08] Tobias Klug, Michael Ott, Josef Weidendorfer, and Carsten Trinitis: *"autopin - Automated Optimization of Thread-to-Core Pinning on Multicore Systems"* Transactions on High-Performance Embedded Architectures and Compilers, 3(4), 2008