Lines of work in HPC

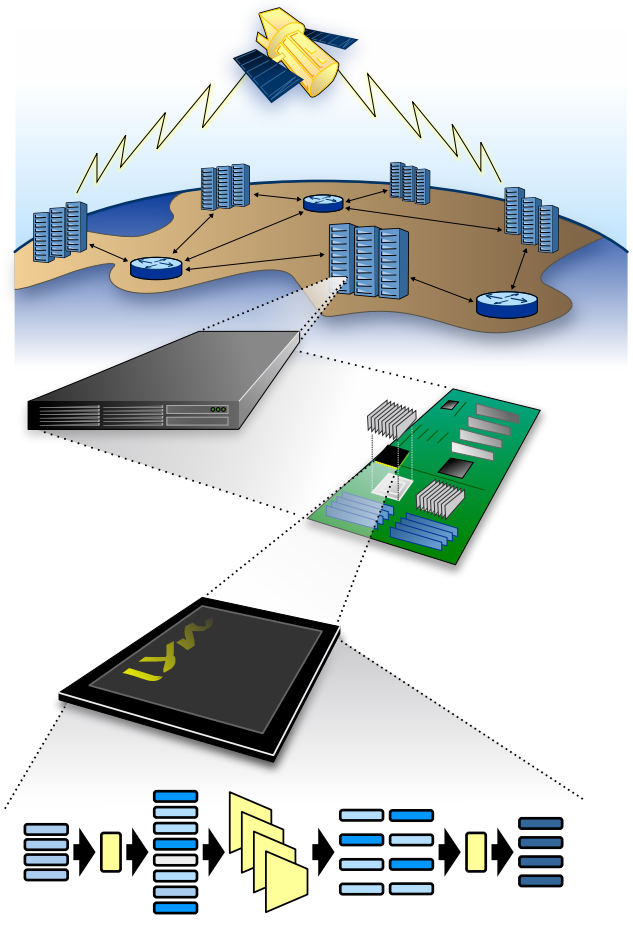
Resource Information Policies
Hardware Assisted Tera-scheduling

Esteban Mocskos

Departamento de Computación (Computer Science Department)
Facultad de Ciencias Exactas y Naturales (School of Sciences)
Universidad de Buenos Aires

8 and 9 of November, 2012

Overview



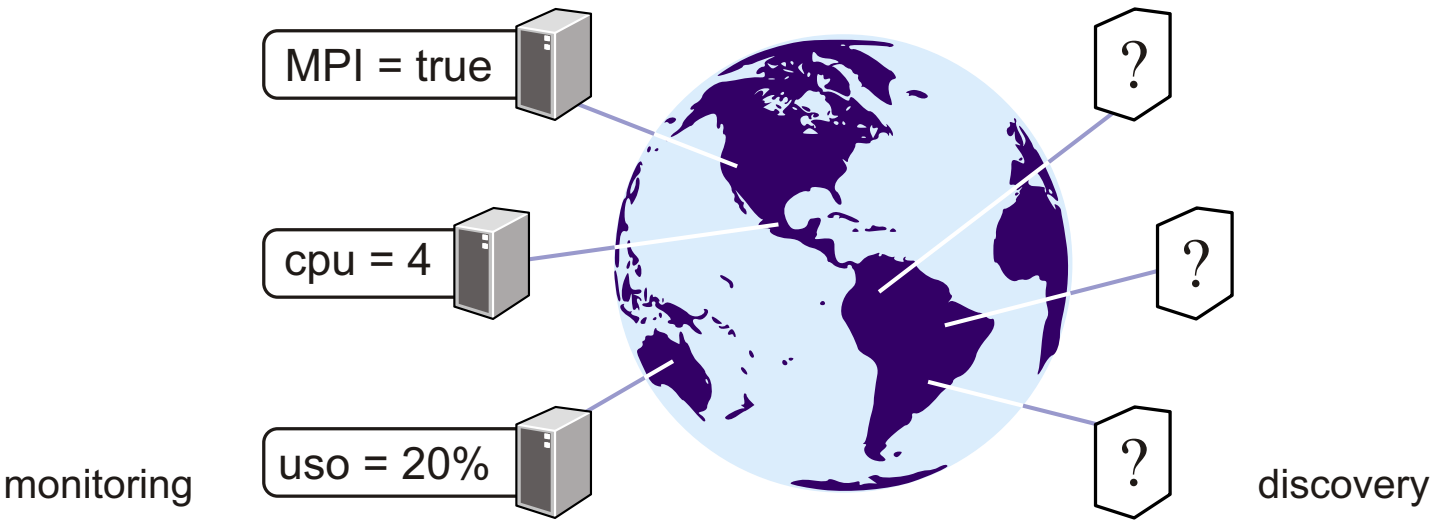
Resource Information Policies

- More and more computing elements will be available to be interconnected.
- The scheduling process needs up-to-date information about resources in the system.
- Any centralized point of failure must be avoided.

Harc

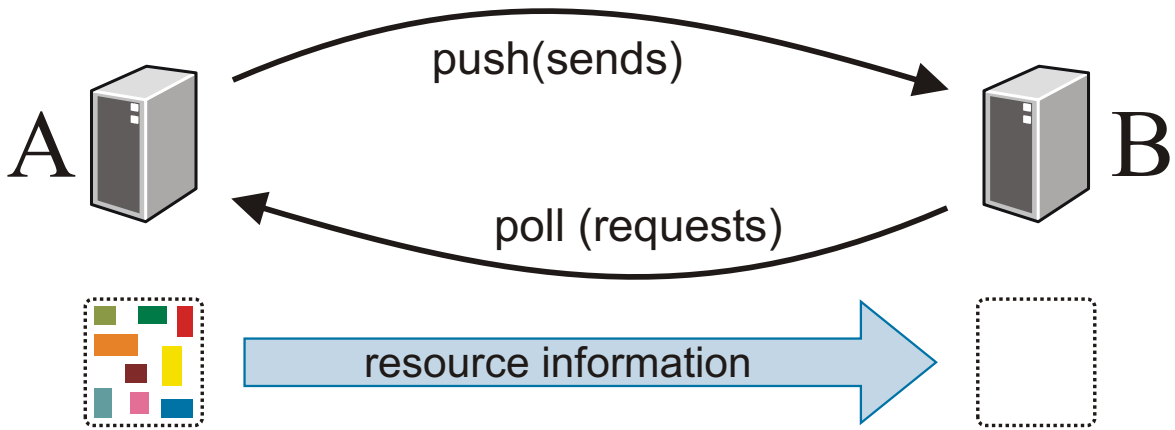
Grid Computing

- Grid technologies allow sharing resources geographically distributed in a transparent way.
- For *efficiently* managing the resources, it is necessary to know their state and availability (Resource Monitoring and Discovery)



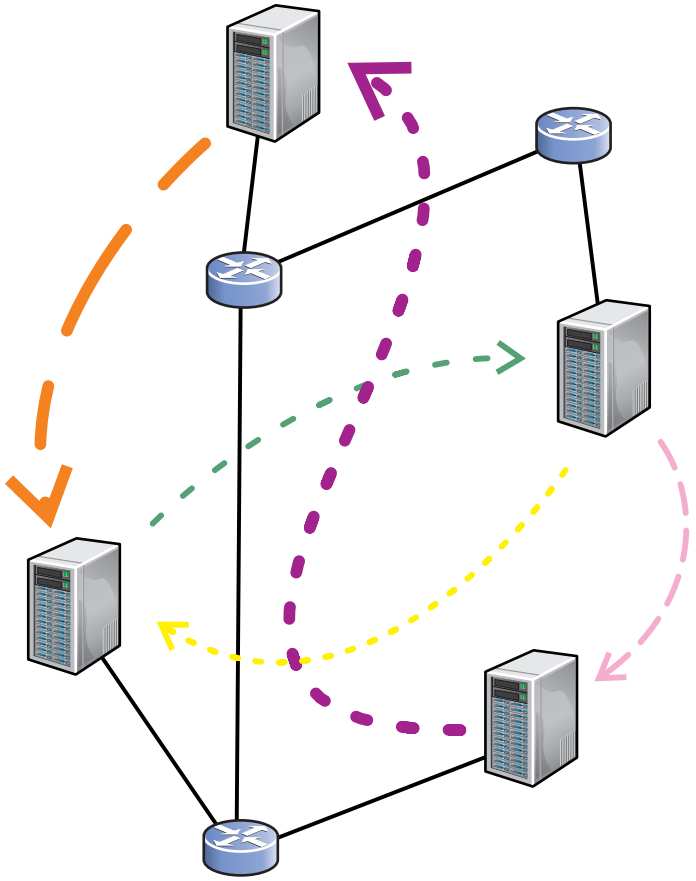
Mechanisms

- An *Index Service* uses two mechanisms to share resources information between nodes
  - *Push*: *A* sends information to *B*
  - *Poll*: *B* requests information from *A*



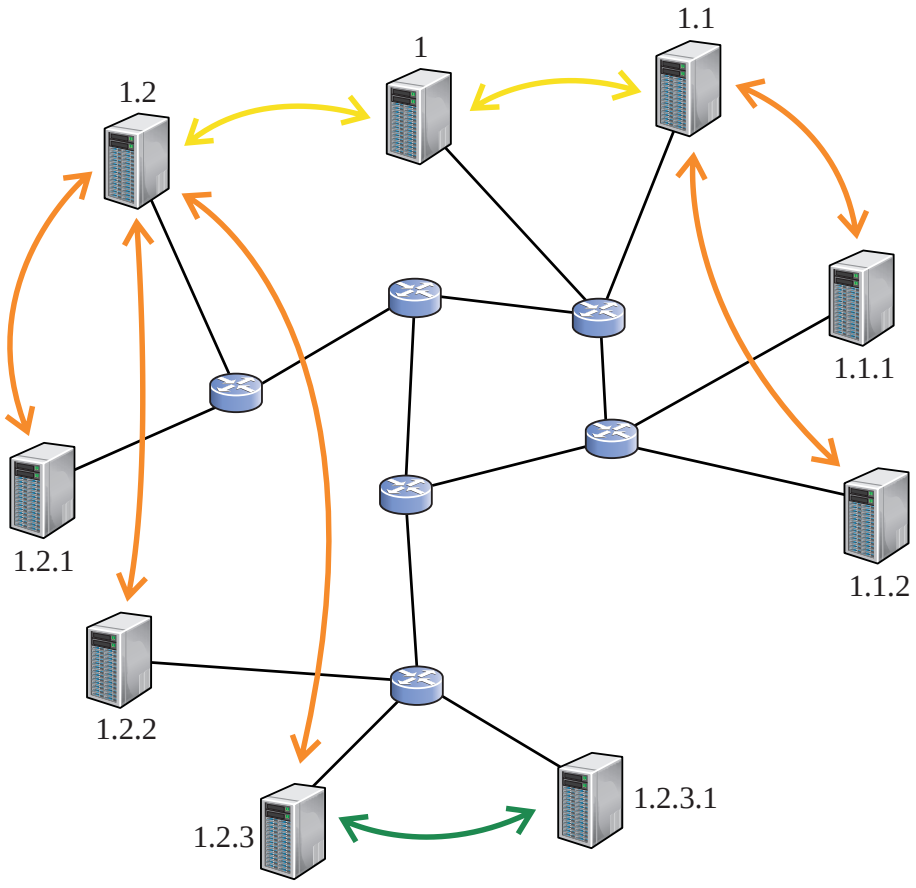
- When you configure an *Index Service* should indicate the providers of information and to which nodes it must provide
- The resource information is associated with a lifetime, can become obsolete (i.e. CPU usage).

Policies



- Defines how the nodes communicate with the others
- Determines the way each node sends and requests information
- Two main groups: predefined hierarchy or peer-to-Peer

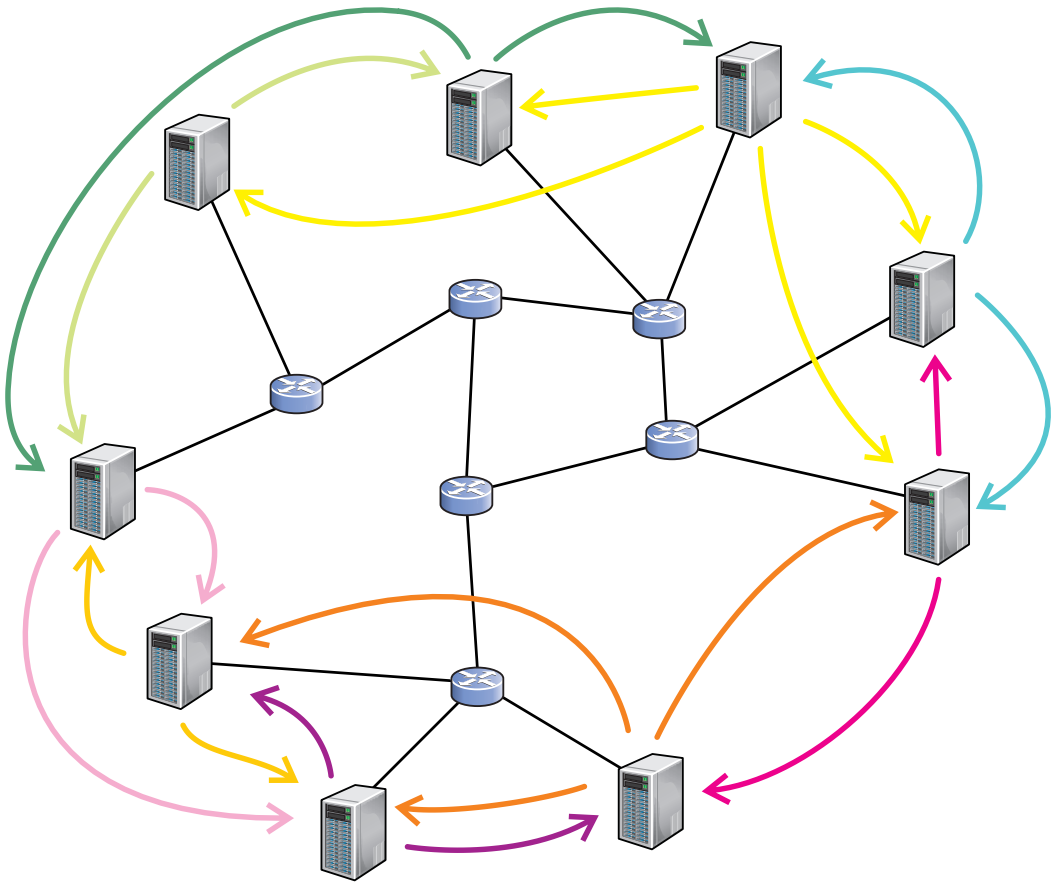
Hierarchical



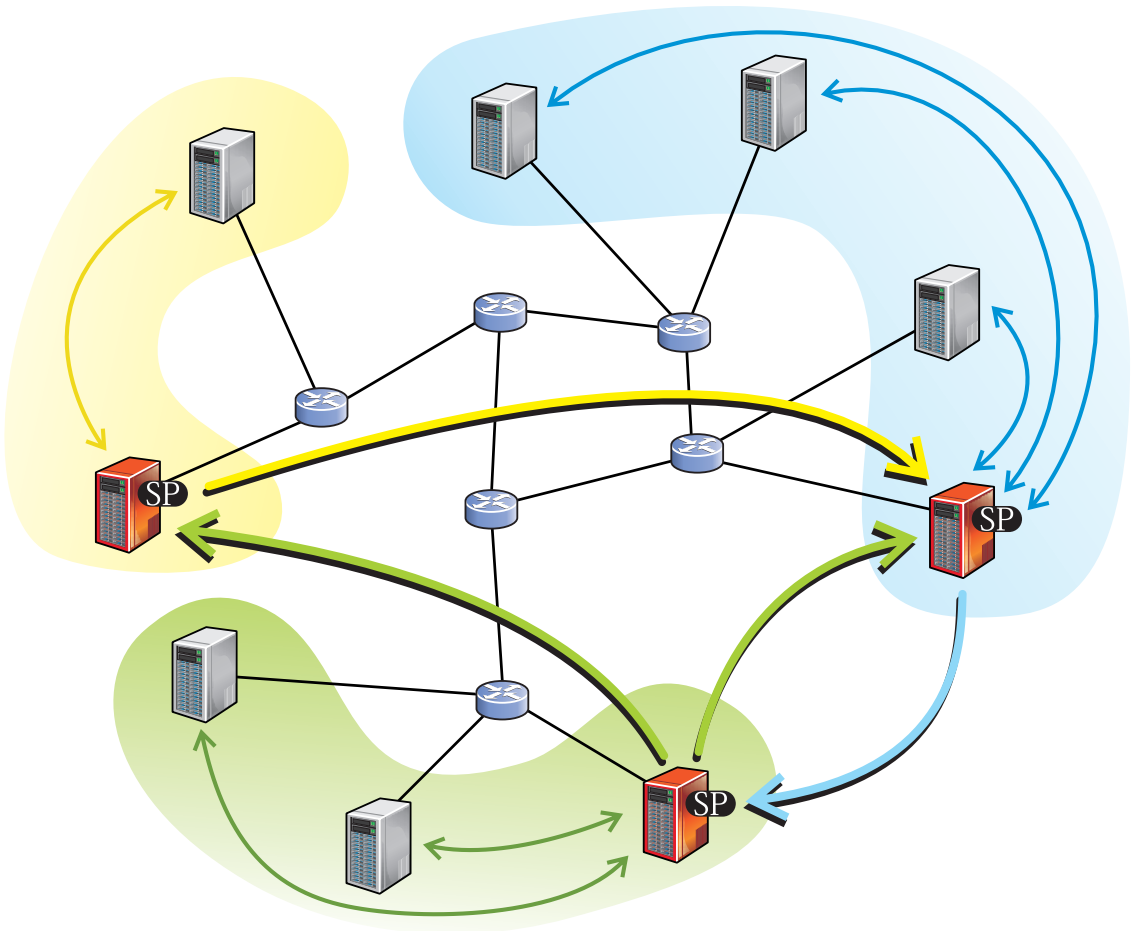
- Each node has at most one father and several sons (eventually none)
- The information flows from the nodes in the lower levels to the top level and in the opposite direction
- To ensure that the information reaches all the nodes, it must have enough lifetime to travel twice the height of the hierarchy

Random

- Each node is assigned a set of neighbors with whom shares information
- They send resource information randomly to any neighbor
- The information can be sent or requested.



Super-Peer



- The nodes are divided into disjoint subsets
- In each subset, one of the nodes is marked as a super-peer
- A centralized scheme is formed using the super-peer as the central point.
- Among the super-peers, the information is shared using a Random policy.

Evaluation of system performance

LIR: captures the amount of information that a particular host has from all the entire grid in a single moment. For the host k, LIR_k is:

$$LIR_k = \frac{\sum_{h=1}^{N} f(age_h, expiration_h) \cdot resourceCount_h}{totalResourceCount}$$

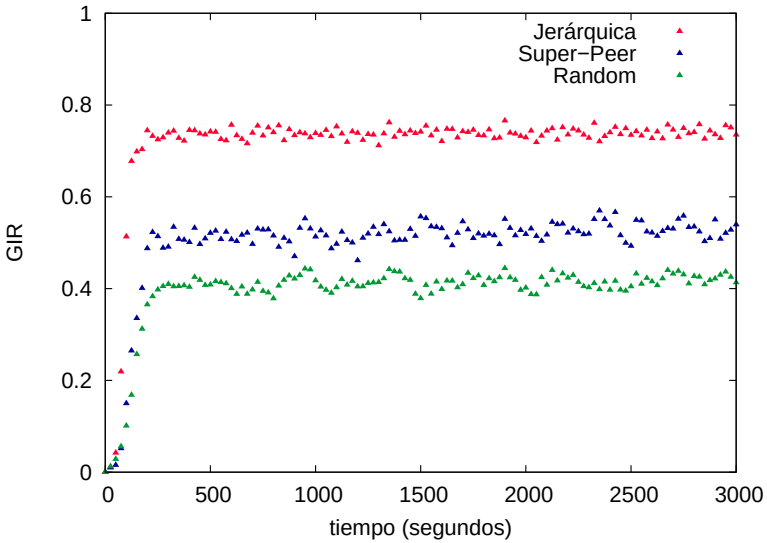
GIR: captures the amount of information that the whole system knows of itself, obtained as the mean value of every node's LIR.

• GridMatrix2

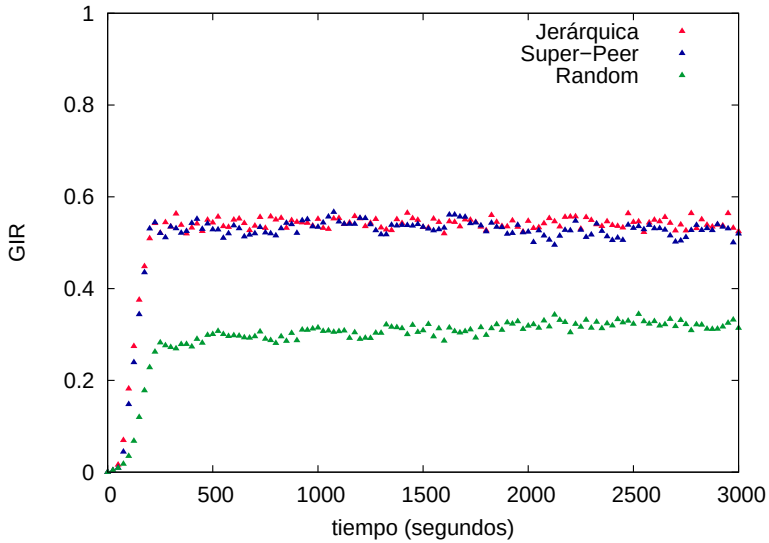
- It is an application for the simulation of the behavior of information distribution policies
- SimGrid2 is used as simulation engine
- Graphical interface for the design of the underlying network
- Simplifies network generation and measures the main properties
- Displays the simulation execution
- Coded in C++, QT and Python (multiplatform)

Basic network topology - Clique

- In this topology the longest path between two nodes is 2 hops
- For this reason it is expected that any policy behaves relatively well.



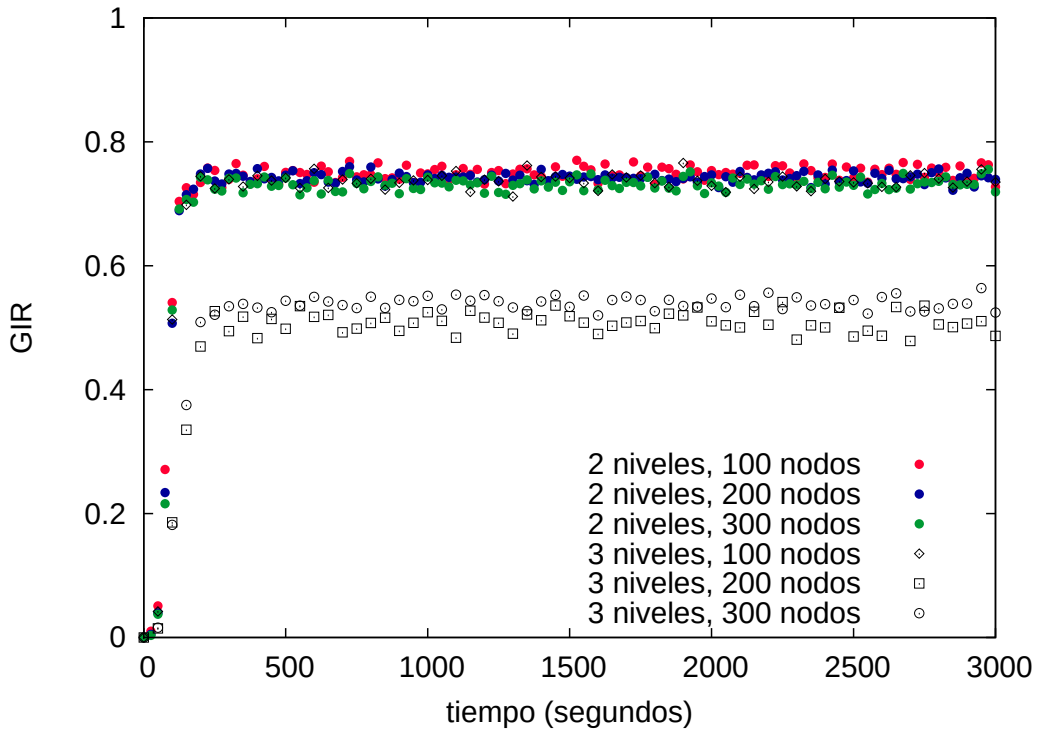
GIR of 100 nodes clique



GIR of 300 nodes clique

- Random has a gradual fall related to the size of the system
- Super-Peer has a stable and acceptable performance
- Hierarchical presents a fall for larger systems, pairing Super-Peer

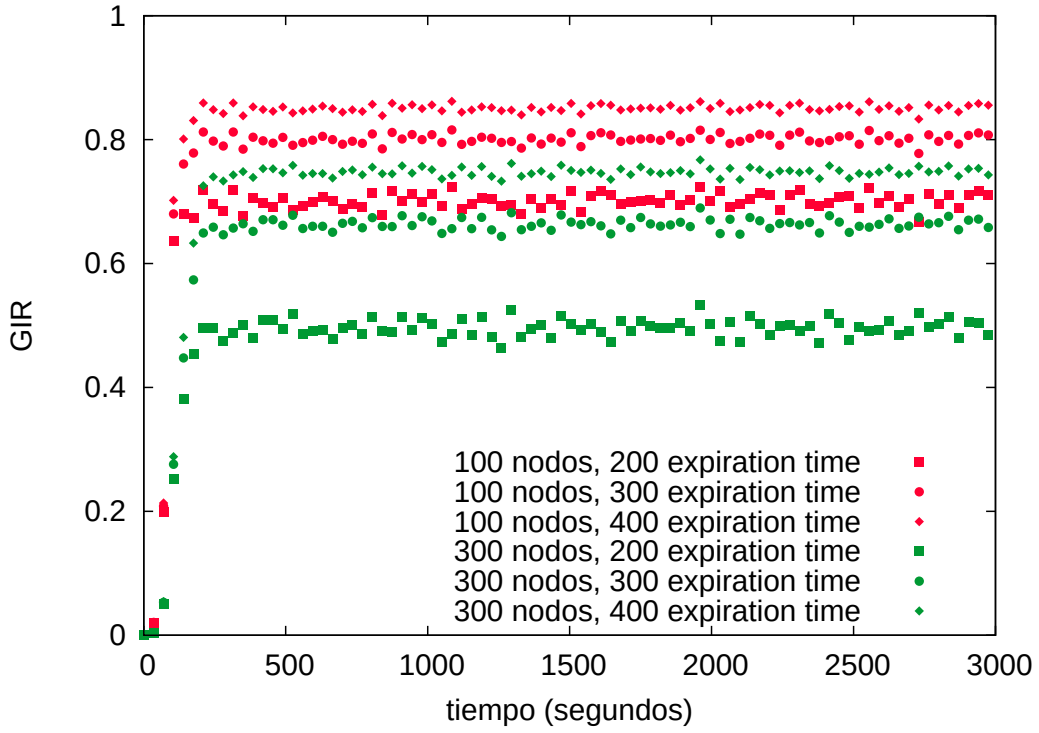
Basic network topology - Clique



GIR in clique with 2 and 3 level hierarchy, expiration time between 200 and 300 sec.

- For two levels, the performance remained around 0.8, regardless of the number of nodes
- In the case of three levels with 200 and 300 nodes, a sharp fall is observed due to the hierarchy construction algorithm

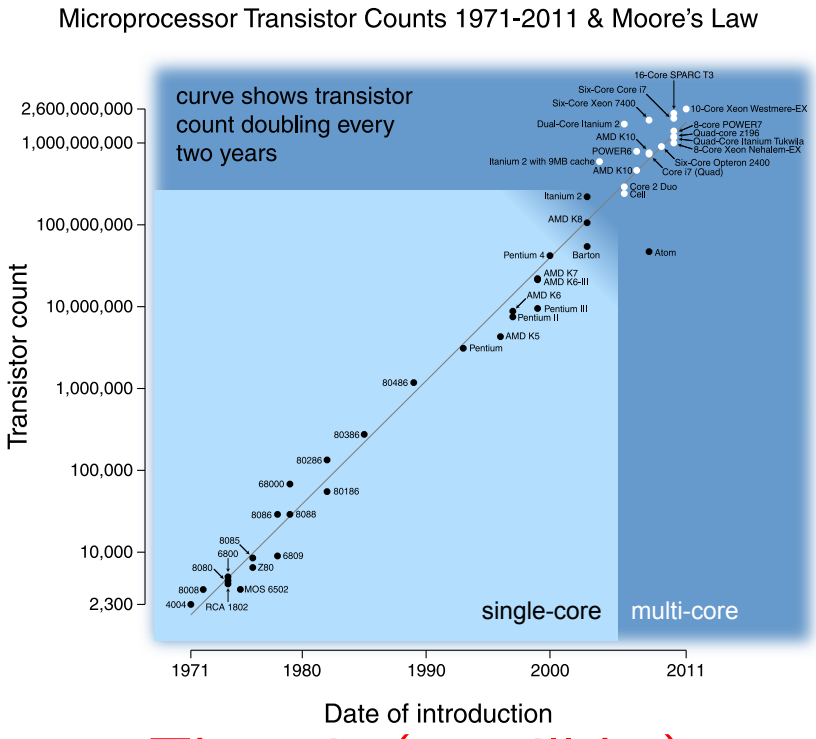
Basic network topology - Clique



GIR of clique for 3-level hierarchical policy, different expiration times

- Different expiration times were used to evaluate it impact.
- The improvement is remarkable, despite the unbalanced hierarchical structure

Problem



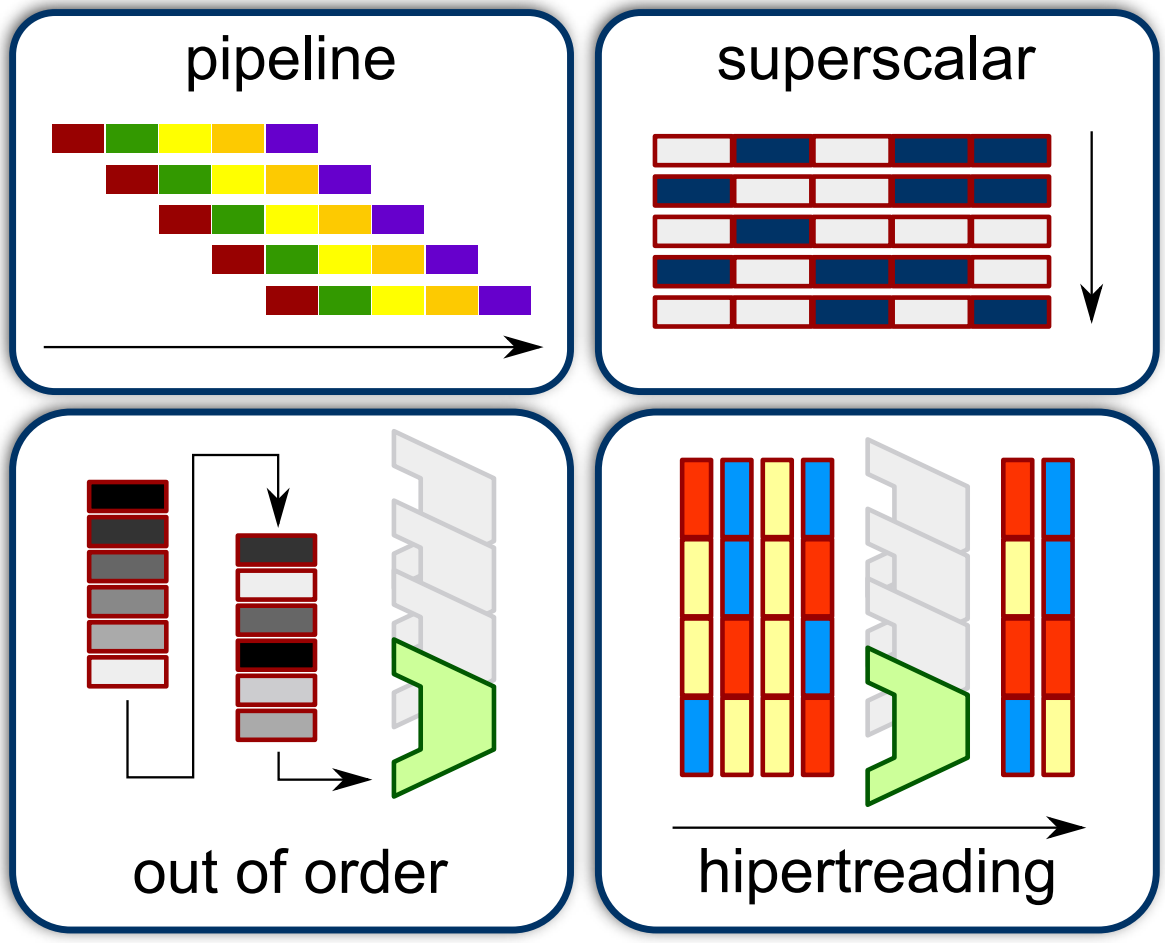
Today

- We would like a faster processor
- But... we can not increase the clock frequency

We need to run *more* instructions in parallel

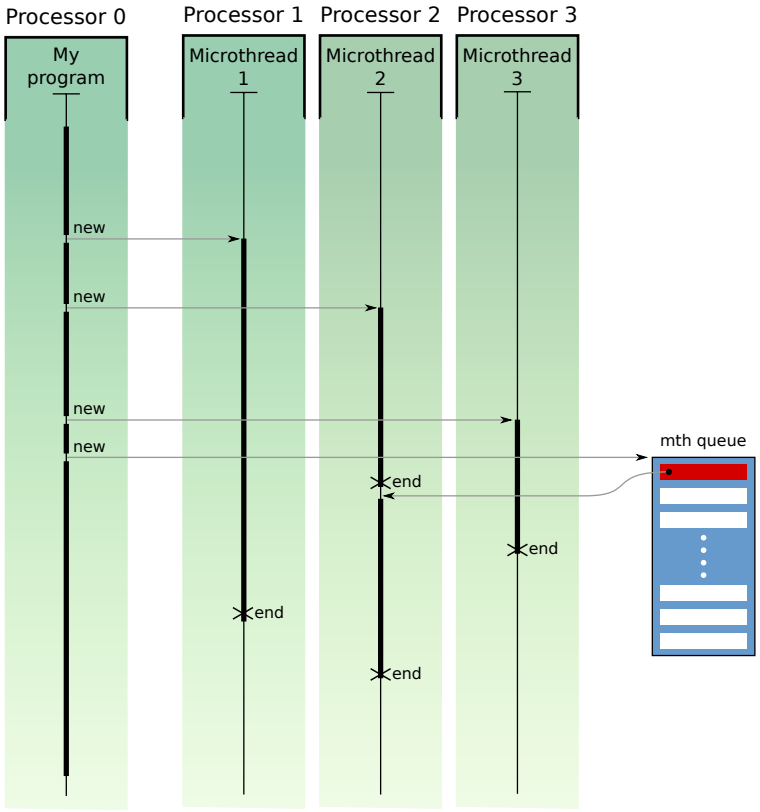
There is (or will be) a need for a new computing model, to take advantage of the increasing amount of cores in each chip

Instruction Level Parallelism - Current Techniques



- Only at single instruction level
- Do not take into account the information from each thread

Improve Instruction Level Parallelism - Microthreads



- One main thread and several short-lived threads collaborating (called microthread).
- Add new instructions to the ISA for launching microthreads.
- The microthreads run in a different core with a copy of the main thread context.
- Each microthread run until it terminates.

We need to do all of this very very very VERY fast

Execution model and new instructions

Assumption

Count with a mechanism to a launch a (micro)thread fast.

Objectives

- Less overhead allows us to further exploit parallelism.
- It must be well supported in software.

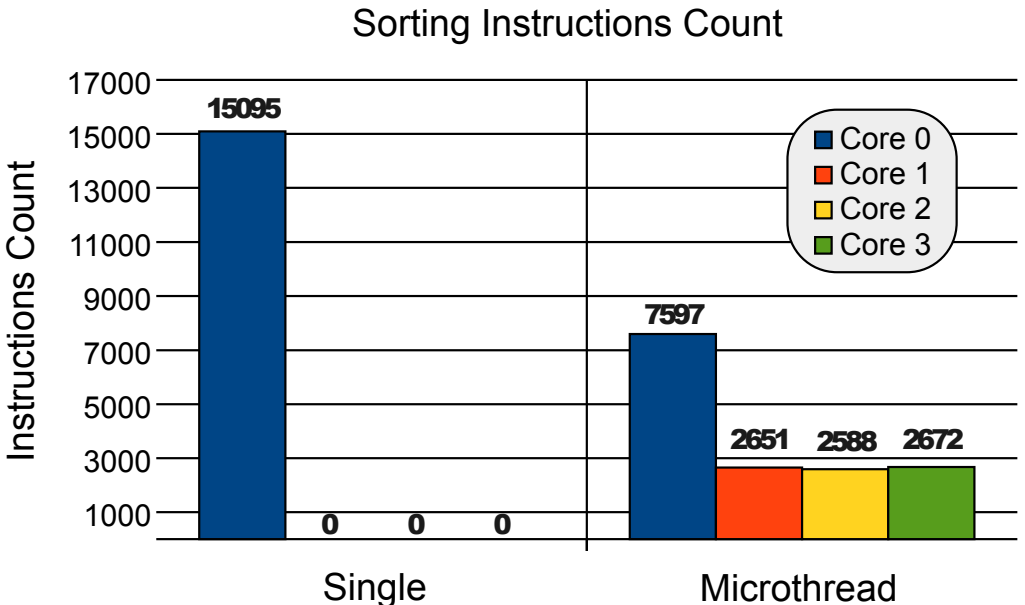
New Primitives

- Launch and stop threads: `mthRun`, `mthEnd`
- Synchronization: `waitForThreads`

How is it done?

Initial tests

- Sorting an array of 100 elements.
- No sense using standard mechanism (i.e. OpenMP): overhead
- Heap Sort Serial solution
  - needed ca 15000 instructionsm to finish
- Heap Sort Parallel solution
  - divide the array in four parts
  - run a heap sort in a microthread for each part
  - merge the four arrays (ca 7000 instructions)



Conclusions

- Studies of resource information policies thinking in very large Grids.
- Planning a new architecture to support the increasing number of cores to come
- Briefly: Trying to be a step ahead and understand the dynamics of the systems to come in the small and large scale.

Staff

These lines of work are pushed by:

- Paula Verghelet, undergraduate student writing her final thesis.
- Maximiliano Geier, PhD student.
- David González Márquez, PhD student.
- Alexis Tchach, PhD student.
- Pablo Turjanski, researcher.
- Diego Fernández Slezak, researcher.