Cluster Computing

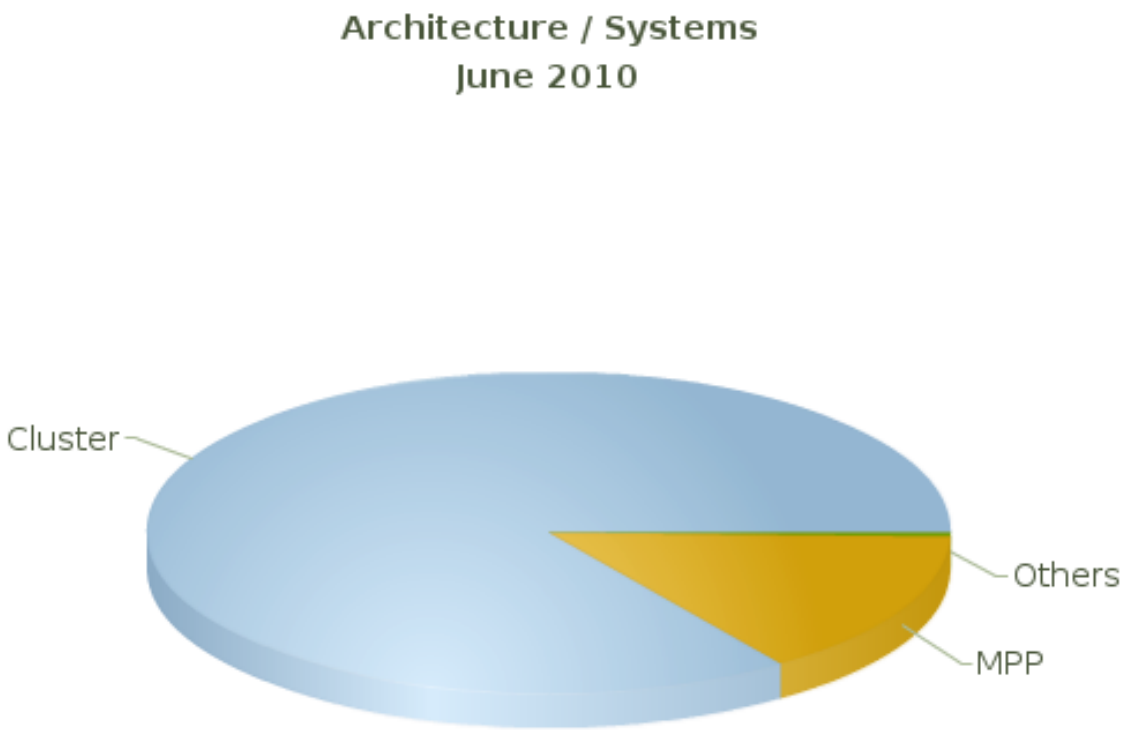
Resource and Job Management for HPC

SC-CAMP

16/08/2010

Summary

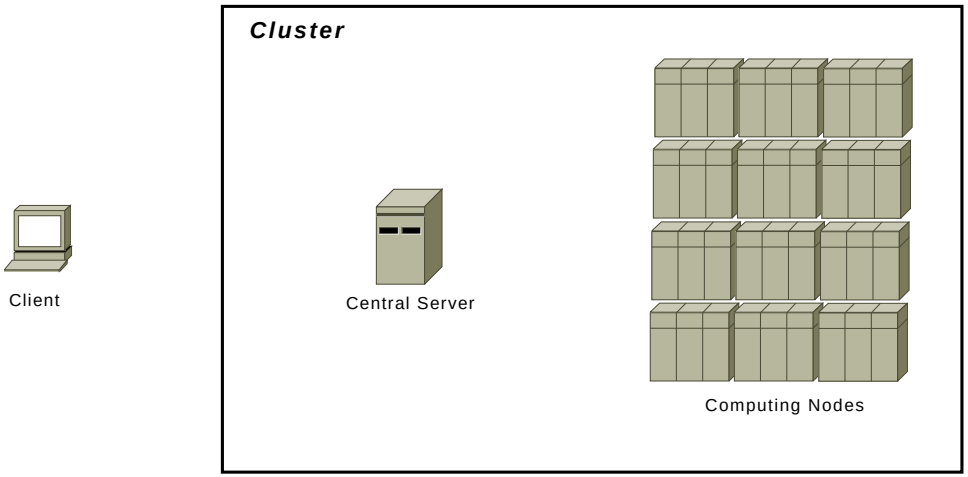
Top500 larger computing infrastructures



http ://www.top500.org/

Definition Cluster

A computer cluster for High Performance Computing is a group of linked computers, working together closely connected to each other through fast local area networks. A cluster is also called *parallel supercomputer*.



Concepts Cluster Computing

Process

Processes are runing on CPUs.

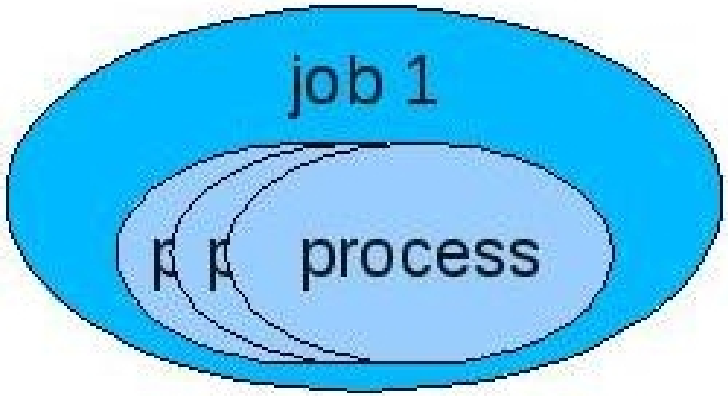
- A process is a program that is running into memory
- On UNIX, several processes may be running on one or several processors (multitask). They are hierarchical and belong to a particular user.
- Each UNIX process has a unique number on a given system. It is called the PID.
- A "thread" (also known as a "light process") is a program that shares a certain amount of memory with another thread belonging to the same father process (under Linux, threads are viewed with "ps -eLf")

Concepts Cluster Computing

Jobs

Processes can be groupped into jobs.

- A *job* may be a single process, a group of processes or even a batch.
- In our context (HPC clusters), a job is a set of processes that have been automatically launched by a scheduler by the way of a user's submitted script.
- A job may result in N instances of the same program on N nodes or processors of a cluster.

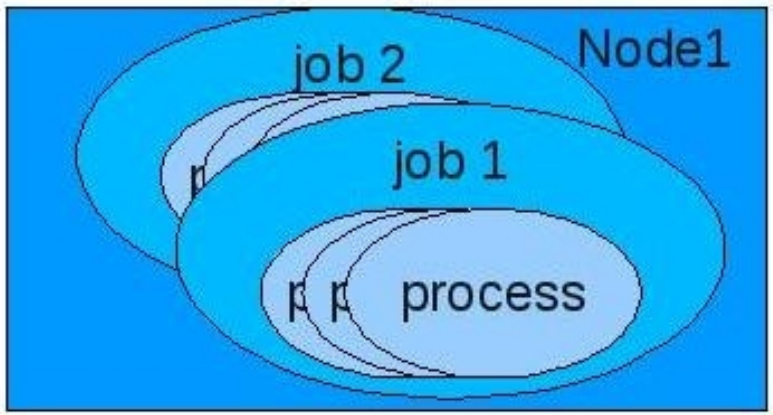


Concepts Cluster Computing

Nodes

Jobs are running on nodes.

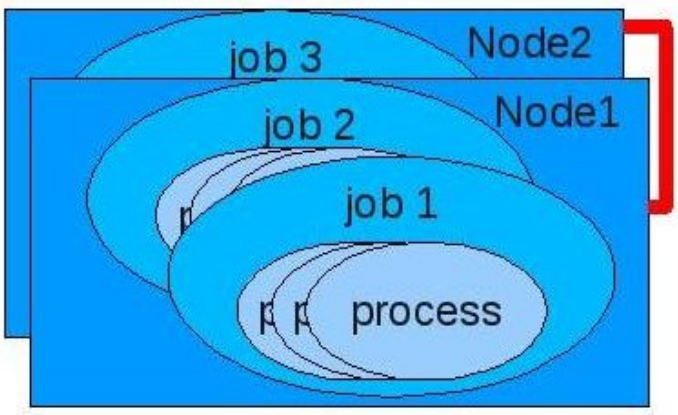
- A *node* is a computer having a set of p CPU, an amount of memory, one or several network interfaces and that may have a storage unit (local disk).



Concepts Cluster Computing

Computing network

Several nodes are connected to a computing network, generaly low latency network (Myrinet, Infiniband, Numalink,...)

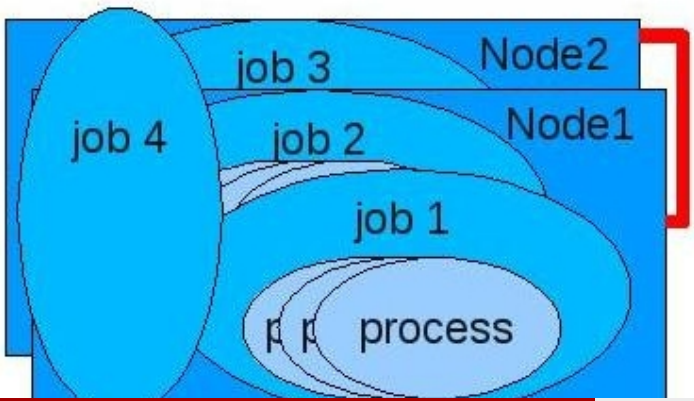


Concepts Cluster Computing

Types of jobs

Jobs maybe "parallel" or "sequential". A parallel job runs on several nodes, using the computing network to communicate.

- *Sequential Job* is a unique process that runs on a unique processor of a unique host.
- *Parallel Job* Several processes or threads that can communicate via a specific library (MPI, OpenMP, threads,...). Some of them are 'shared memory' parallel jobs (they run on a unique multiprocessor host) and some are 'distributed memory' parallel jobs (they may run on several hosts that communicate via a low latency high speed network) Warning : a 'fake' parallel job may hide several sequential jobs (several independent processes)



Concepts Cluster Computing

Types of jobs

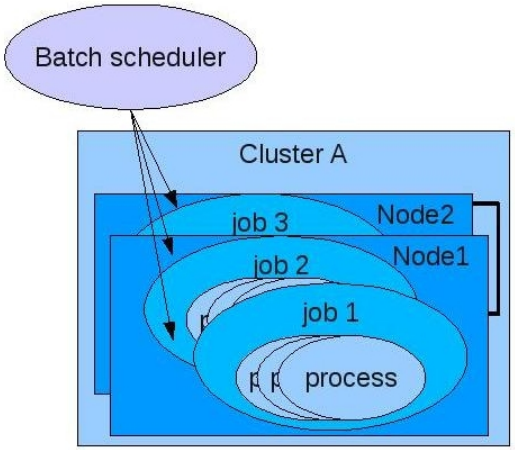
The Jobs can come at the form of Batch or Interactive type.

- A *batch* job is a script. A shell script is a given list of shell commands to be executed in a given order
  - Todays shells are so sophisticated that you can create scripts that are real programs with some variables, controle structures, loops,
  - We sometimes also call script a program that has been written in an interpreted language (like perl, php, or ruby)
- An *interactive* job is usually an allocation upon one or more nodes where the user gets a shell on one of the cluster nodes.

Concepts Cluster Computing

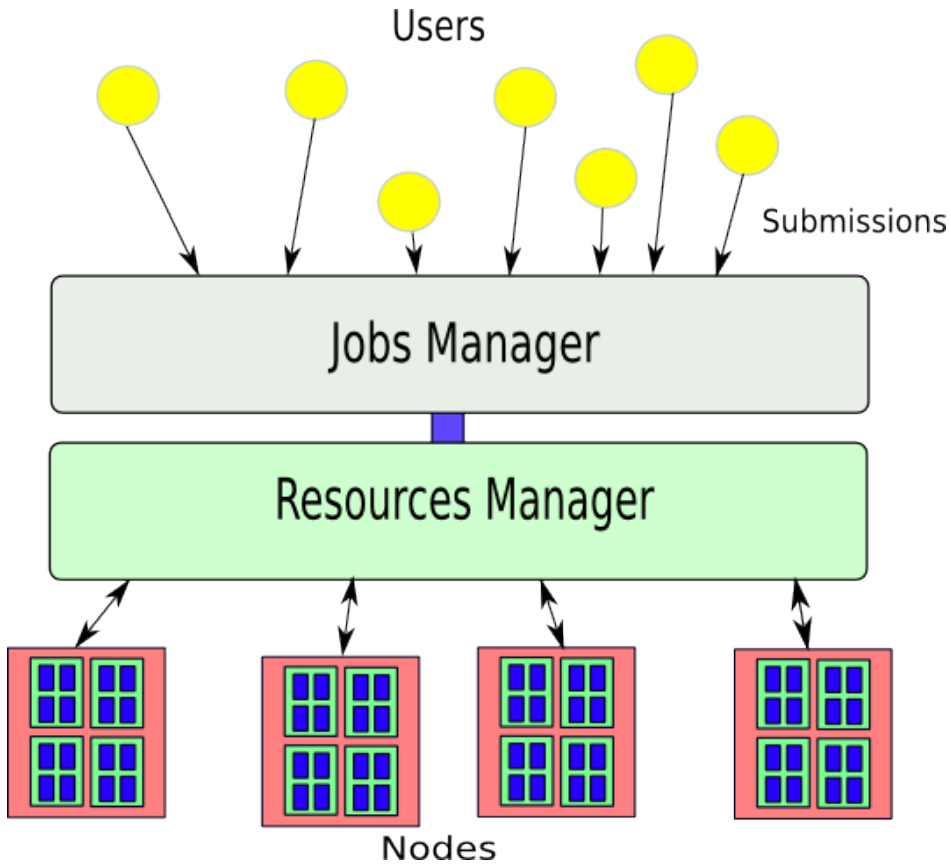
Resource and Job Management System

The *Resource and Job Management System* (RJMS) or Batch Scheduler is a particular software of the cluster that is responsible to distribute computing power to user jobs within a parallel computing infrastructure.



Concepts Resource and Job Management System

The goal of a Resource and Job Management System is to satisfy users demands for computation and achieve a good performance in overall system's utilization by efficiently assigning jobs to resources.



Concepts Resource and Job Management System

This assignement involves three principal abstraction layers :

- the declaration of a job where the demand of resources and job characteristics take place,
- the scheduling of the jobs upon the resources
- and the launching and placement of job instances upon the computation resources along with the job's control of execution.

In this sense, the work of a RJMS can be decomposed into three main subsystems : Job Management, Scheduling and Resource Management.

RJMS subsystems	Principal Concepts	Advanced Features
Resource Management	-Resource Treatment (hierarchy, partitions,..) -Job Launcing, Propagation, Execution control -Task Placement (topology,binding,..)	- High Availability - Energy Efficiency - Topology aware placement
Job Management	-Job declaration (types, characteristics,...) -Job Control (signaling, reprioritizing,...) -Monitoring (reporting, visualization,..)	- Authentication (limitations, security,..) - QOS (checkpoint, suspend, accounting,...) - Interfacing (MPI libraries, debuggers, APIs,..)
Scheduling	-Scheduling Algorithms (builtin, external,..) -Queues Management (priorities,multiple,..)	- Advanced Reservation - Application Licenses

TAB.: General Principles of a Resource and Job Management System

Scheduling Policy	Description
FIFO	jobs are treated with the order they arrive.
Backfill	fill up empty wholes in the scheduling tables without modifying the order or the execution of previous submitted jobs.
Gang Scheduling	multiple jobs may be allocated to the same resources and are alternately suspended/resumed letting only one of them at a time have dedicated use of those resources, for a predefined duration.
TimeSharing	multiple jobs may be allocated to the same resources allowing the sharing of computational resources. The sharing is managed by the scheduler of the operating system
Fairshare	take into account past executed jobs of each user and give priorities to users that have been less using the cluster.
Preemption	suspending one or more "low-priority" jobs to let a "high-priority" job run uninterrupted until it completes

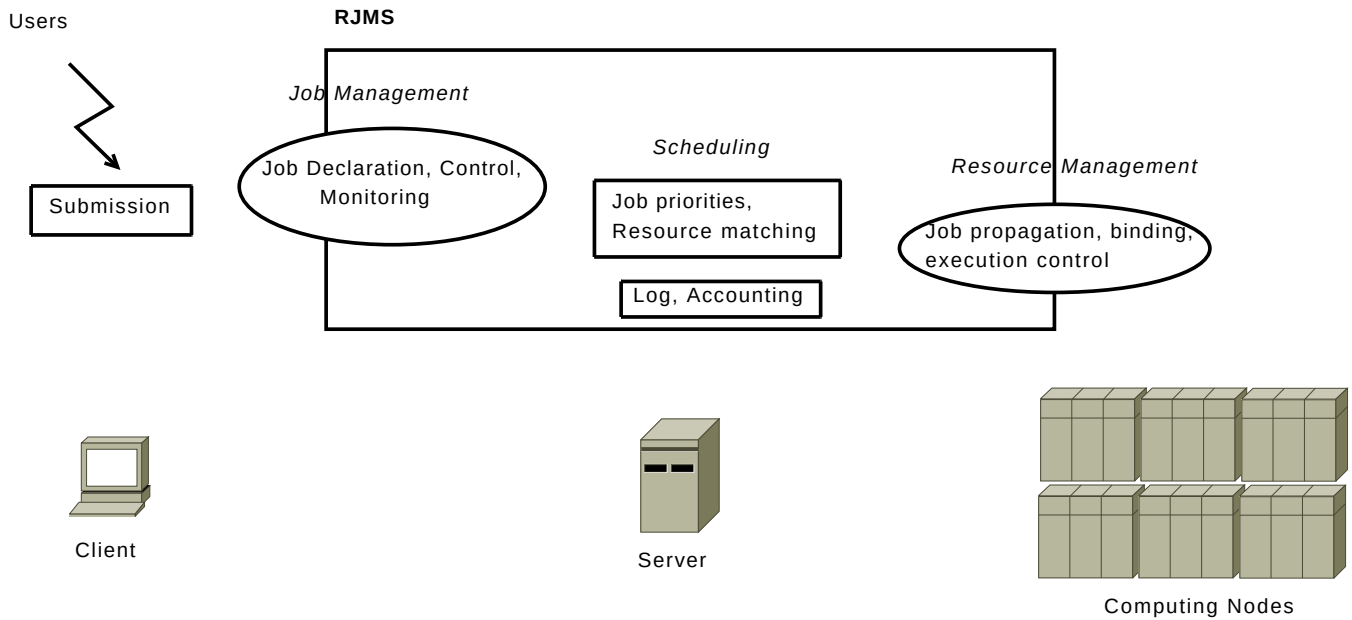
TAB.: Common Scheduling policies for RJMS

Opensource and Commercial RJMS

- Context : **High Performance Computing**
- A lot of them : **Condor** , **Sun Grid Engine (SGE)**, **MAUI/Torque**, **Slurm**, **OAR**, **Catalina**, *LSF* , **Lava**, *PBS Pro*, *Moab*, Loadleveler, CCS...
- `http://en.wikipedia.org/wiki/Job_scheduler`

RJMS General organization

- A central host
- Client programs (command line) to interract with users
- A lot of configuration parameters



RJMS Features (1/2)

non-exhaustive list

- Interactive tasks (*submission*)(shell) / Batch
- Sequential tasks
- Walltime limit. (**very important for scheduling !)**
- Exclusive / non-exclusive access to resources
- Resources matching
- Scripts Epilogue/Prologue (before and after tasks)
- *Monitoring* of the tasks (resources consumption)
- Job dependencies (*workflow*)
- Logging and accounting
- Suspend/resume

RJMS Features (2/2)

non-exhaustive list

- Array jobs
- First-Fit (Conservative Backfilling,)
- Fairsharing
- ...

Objectives

OAR has been designed with **versatililty** and **customization** in mind.

- Following technological evolutions (machines and infrastructures more and more complicated)
- Initial context : CIMENT ▸ ... and Grid5000 ▸ ...
- Different contexts adaptation (cluster, cluster-on-demand, virtual cluster, multi-cluster, lightweight grid, experimental platform as Grid'5000, *big cluster*, special needs).

CIMENT local HPC center

- Université Joseph Fourier (Grenoble)
- A dozen of heterogeneous supercomputers, more than 2000 cores today
- special feature : a lightweight grid (CiGri) making profit of unused cpu cycles through best-effort jobs (a OAR feature)
- Great collaborative work production/research

Grid 5000

- Grid for computing experiments
- 9 sites in France + 2 sites abroad
- Interconnection gigabit Renater
- More than 5000 cores today

OAR special features

non-exhaustive list

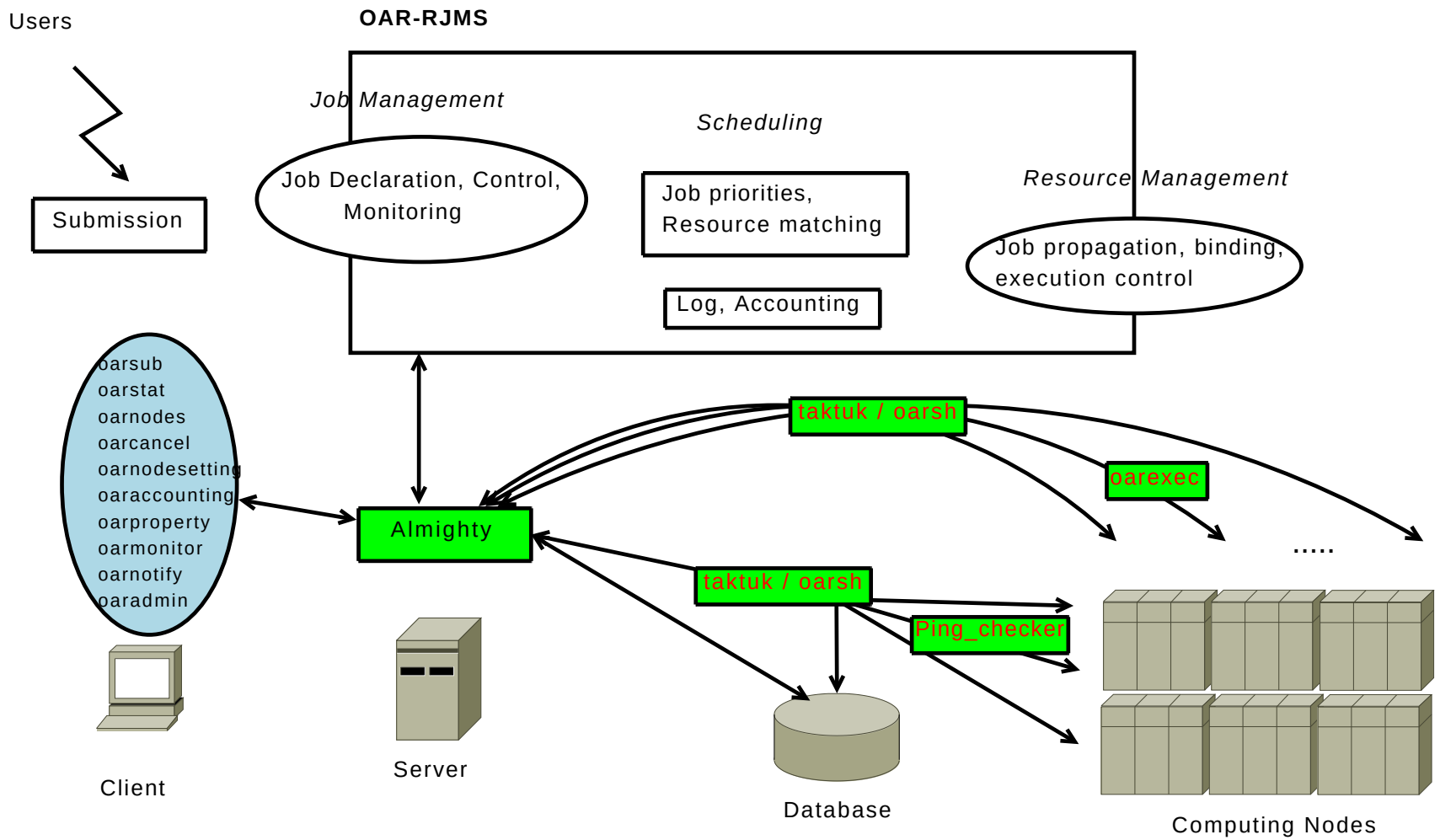
- Classical features +
- Advance Reservation
- **Hierarchical expressions into requests**
- **Different types of resources (ex licence, storage capacity, network capacity...)**
- **Besteffort tasks** (zero priority task, highly used by *CiGri*)
- **Multiple task types** (besteffort, deploy, timesharing, idempotent, power, cosystem ...) (customizable)
- **Moldable tasks**
- **Energy saving**

OAR : design concepts

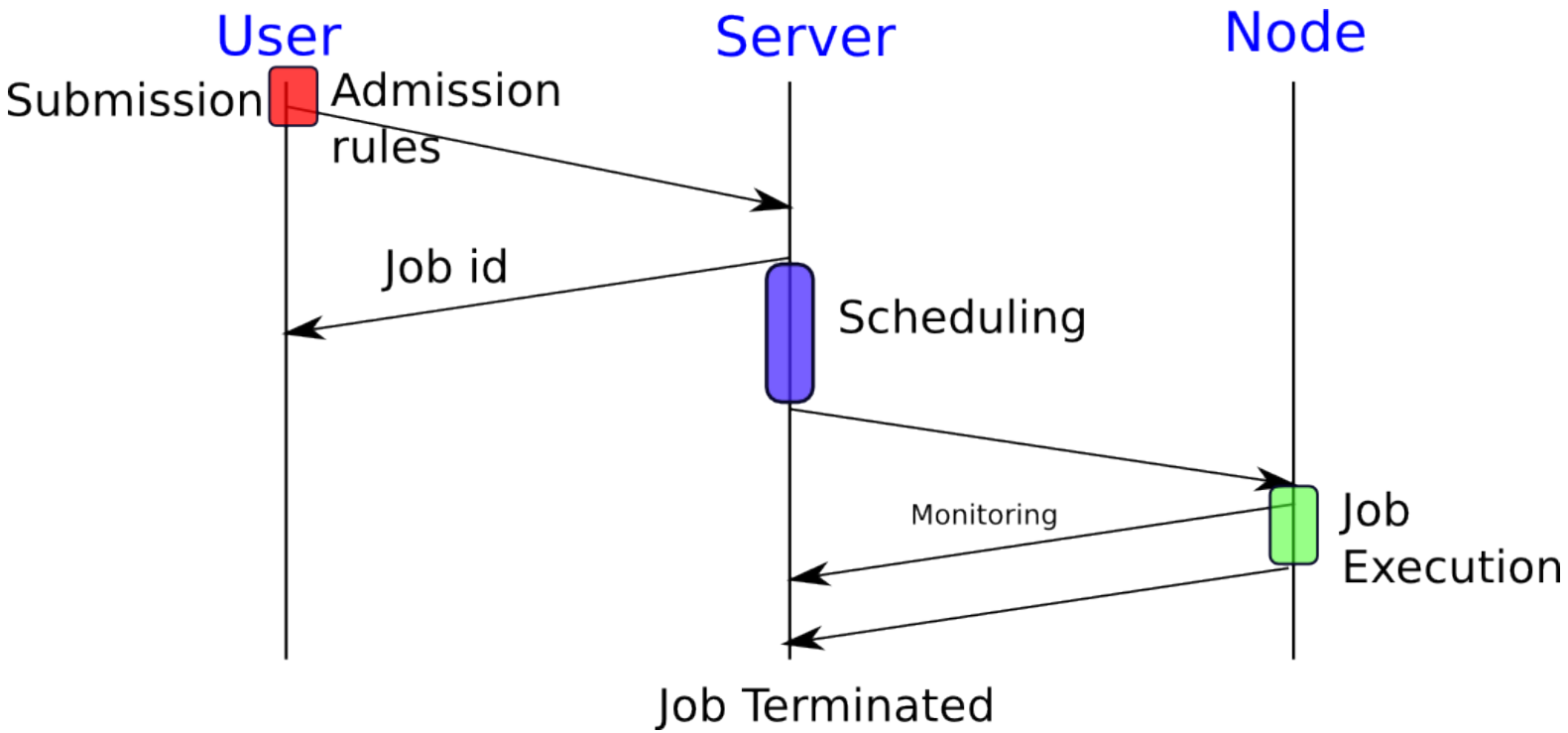
High level components use

- **Relational database** (MySql/PostgreSQL) as the kernel to store and exchange data
  - resources and tasks data
  - internal state of the system
- **Script languages** (Perl, Ruby) for the execution engine and modules
  - Well suited for system parts of the code
  - High level strcutures (lists, hash tables, sort...)
  - Short developpement cycles
- **Other components** : SSH, CPUSETS, Taktuk
  - **SSH, CPUSET** (isolation, cleaning)
  - **Taktuk** adaptative parallel launcher

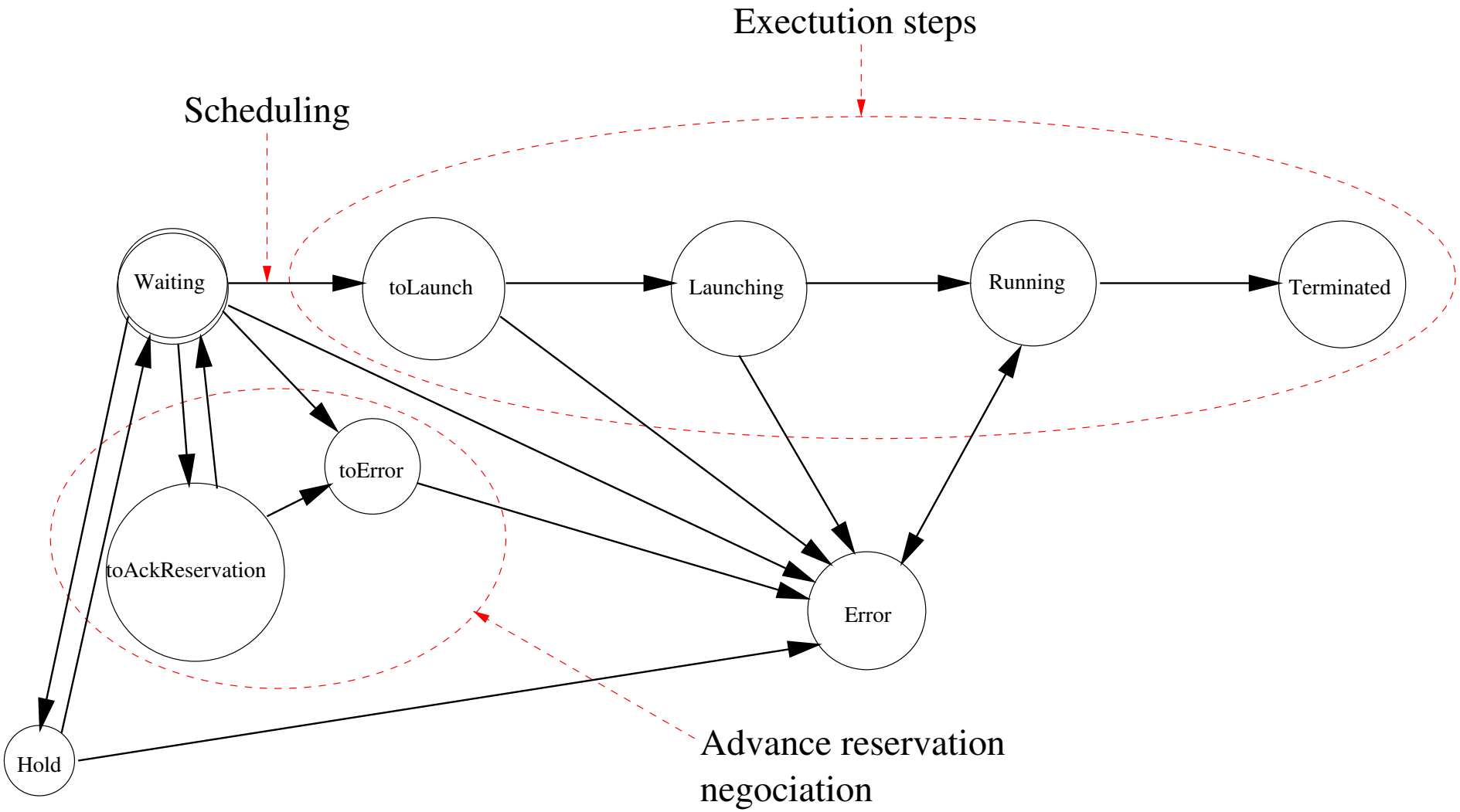
OAR : general organisation



Job life cycle



Task status diagram



Submission examples : OAR

Interactive task : [1]

- **oarsub -I nodes=4 -I**

Batch submission (with *walltime* and choice of the queue) :

- **oarsub -q default -l walltime=2 :00,nodes=10 /home/toto/script**

Advance reservation :

- **oarsub -r "2008-04-27 11 :00" -l nodes=12**

Connection to a reservation (using task's id) :

- **oarsub -C 154**

[1] **Note :** Each submission command returns an id number.

Hierarchical Resources

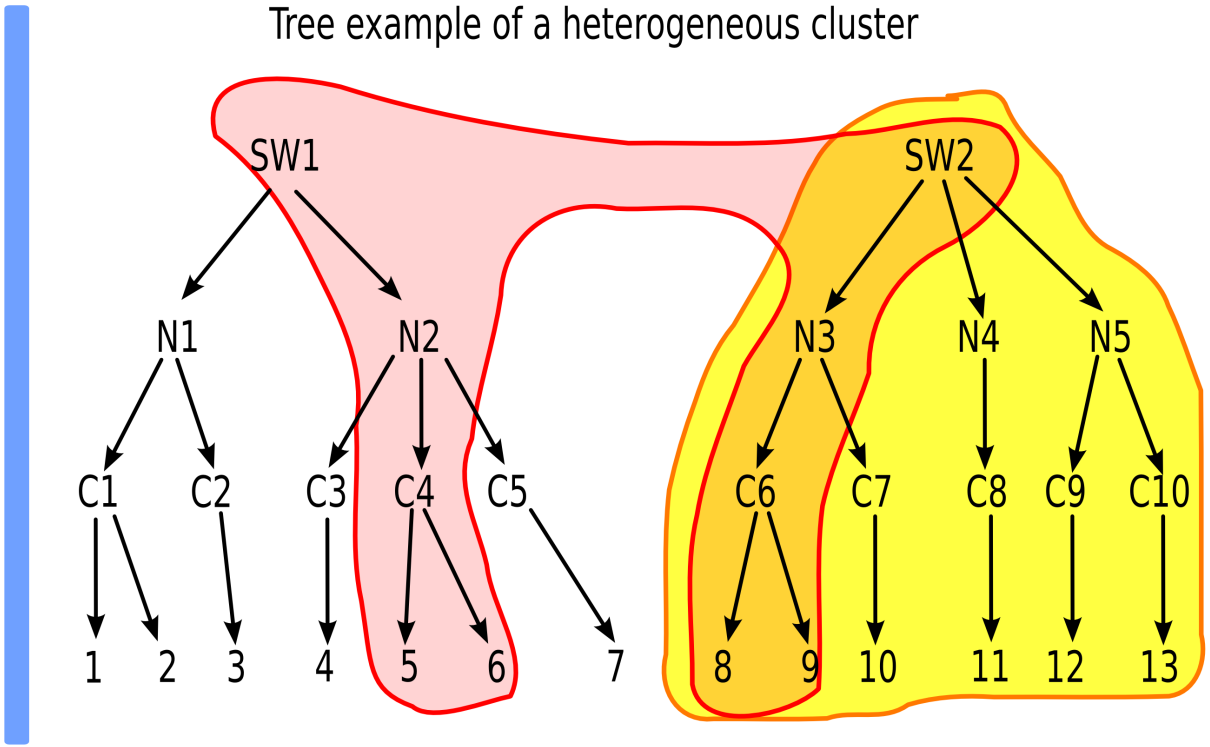
Resource property hierarchy

SWITCH

NODES

CPU

CORE



You can configure your own hierarchy with the property names that you want

`oarsub -l /switch=2/nodes=1/cpu=1/core=2` This command reserves 2 cores on a cpu on a node on 2 different switchs (so 2 computers)

`oarsub -l /switch=1` This command reserves 1 switch entirely

Scheduling

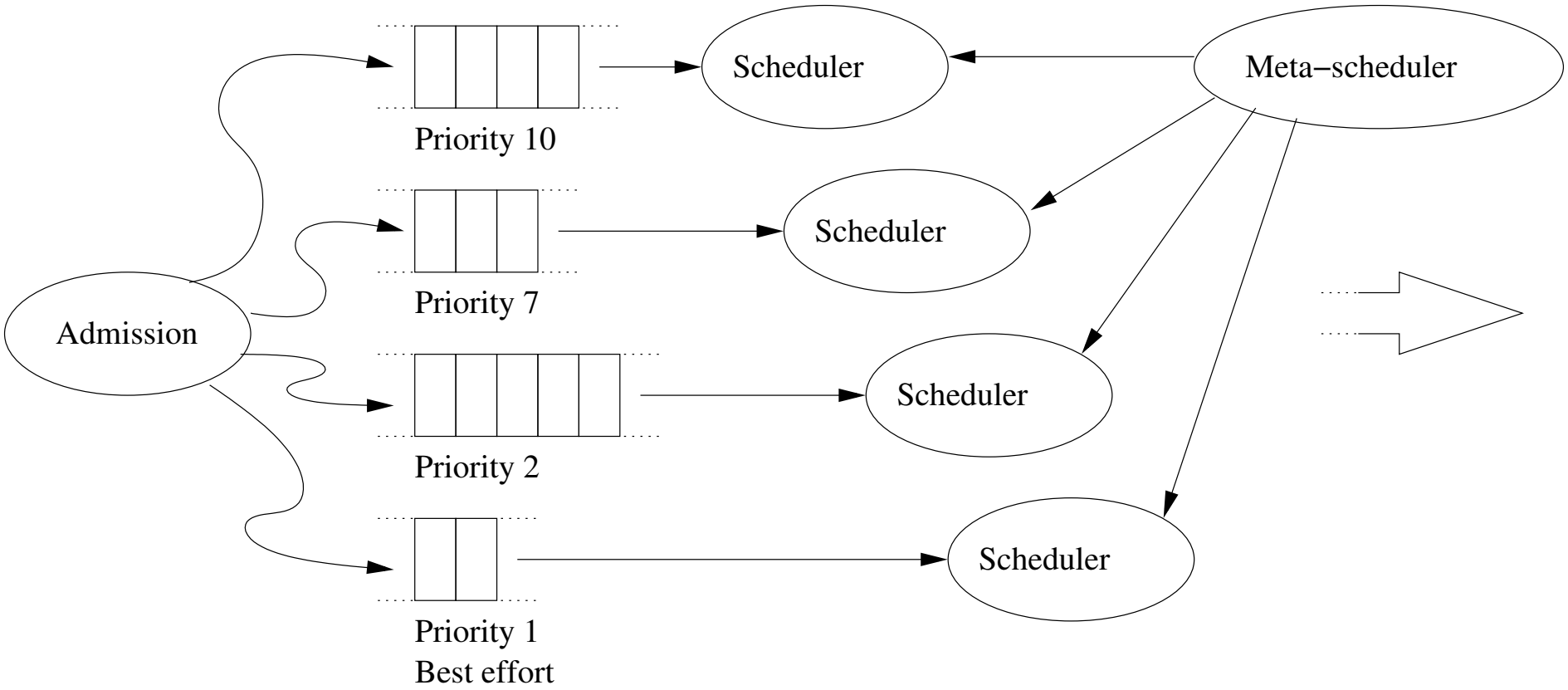
- Scheduling is the process that is responsible to assign jobs according to the users needs and predefined rules and policies, upon available computational resources that match with the demands.
- A typical scheduler functions in cooperation with queues which are elements defined to organize jobs according to similar characteristics (for example priorities).
[2]
- Numerous **parameters** are used to guide and scope the allocations and priorities.

[2] **Note :** scheduling is computed again at each major change in the state of a task.

Scheduling organization into OAR

Tasks are put into queues

- each queue as a priority level
- each queue obey to a scheduling policy

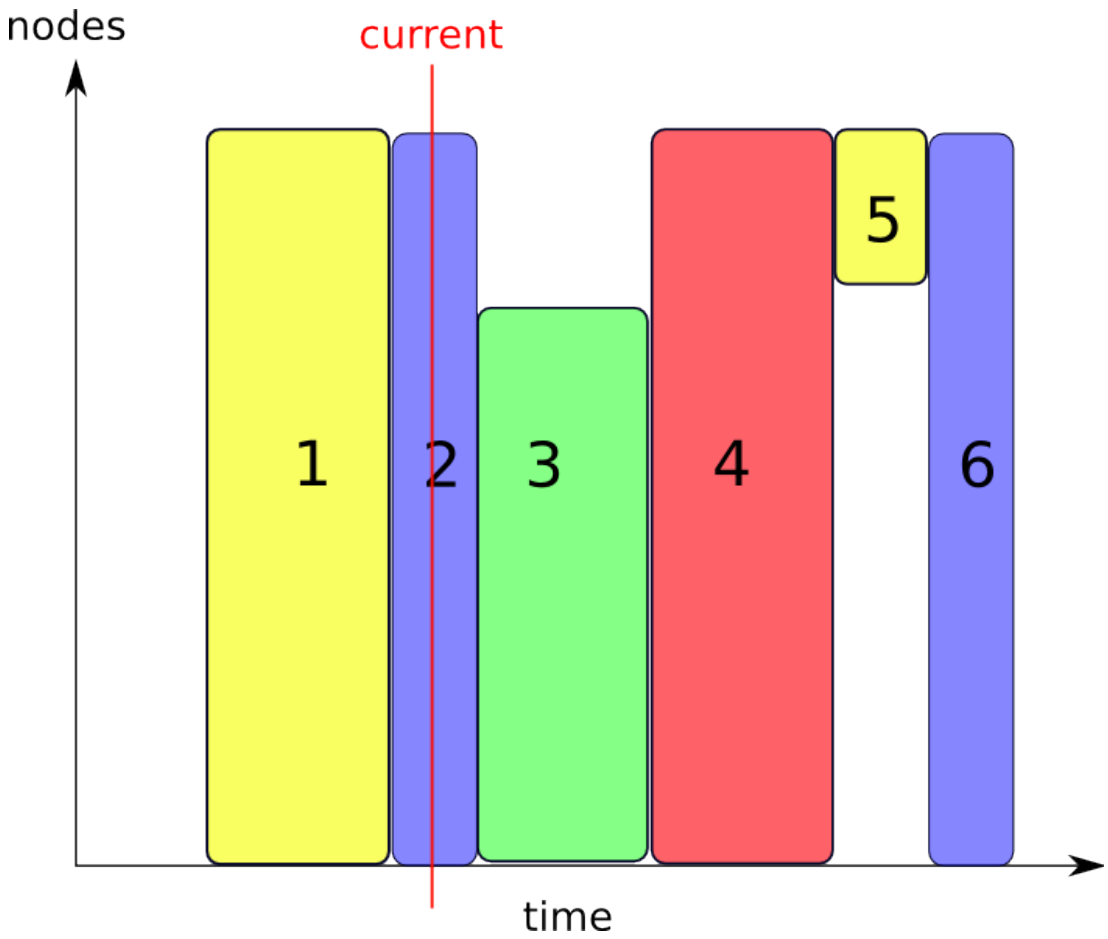


Resource matching

A preliminary step to scheduling

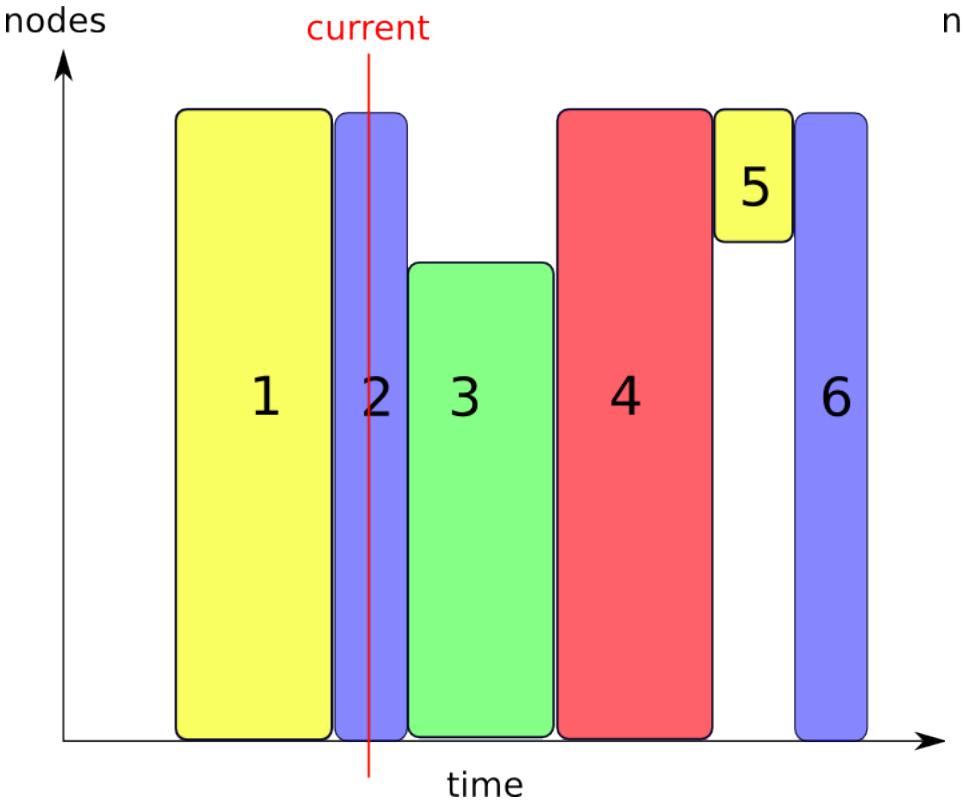
- Resources **filtering**
- Resources **sorting**
- Allows the user to have special needs
- memory, architecture, special hosts, OS, workload...

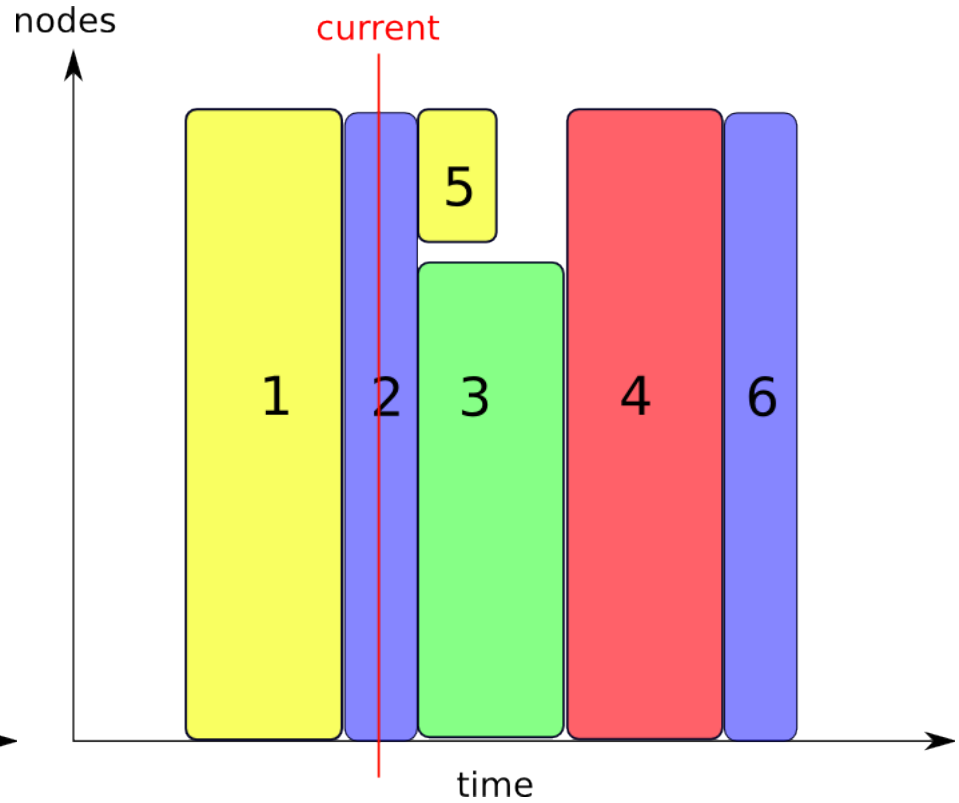
FIFO : Fisrt-In First-Out



First-Fit (Backfilling)

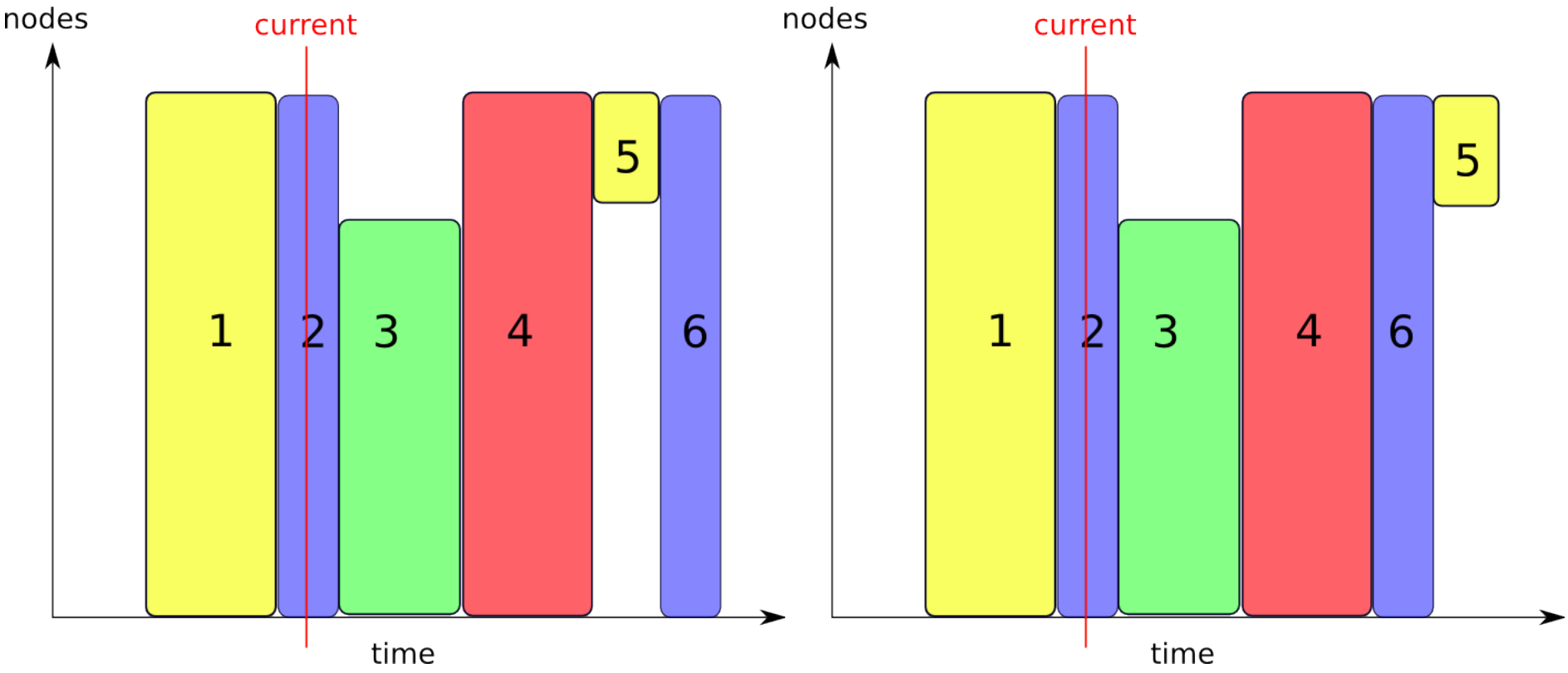
Holes are filled if the previous tasks order is not changed.



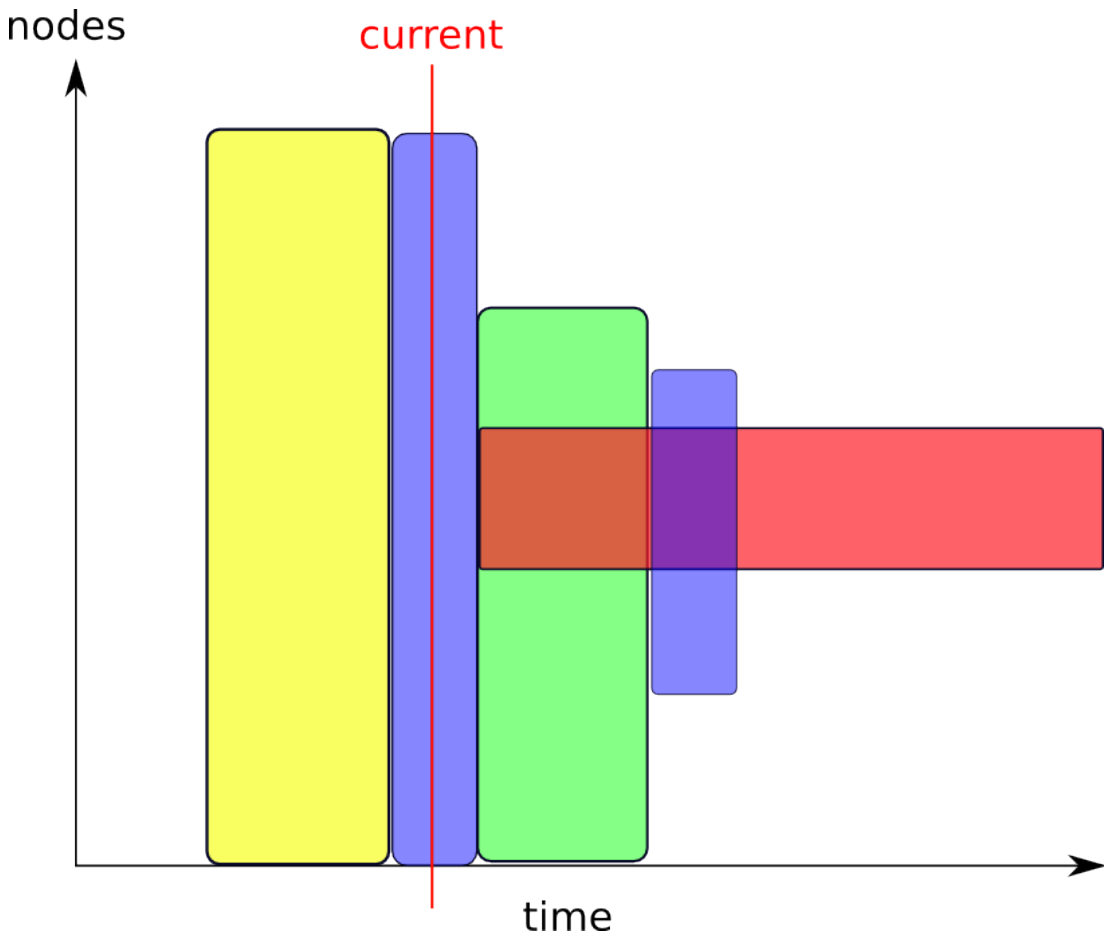


FairSharing

The order of tasks is computed depending on what has already been consumed (low time-consuming users are prioritized) A time-window and weighting parameters are defined.

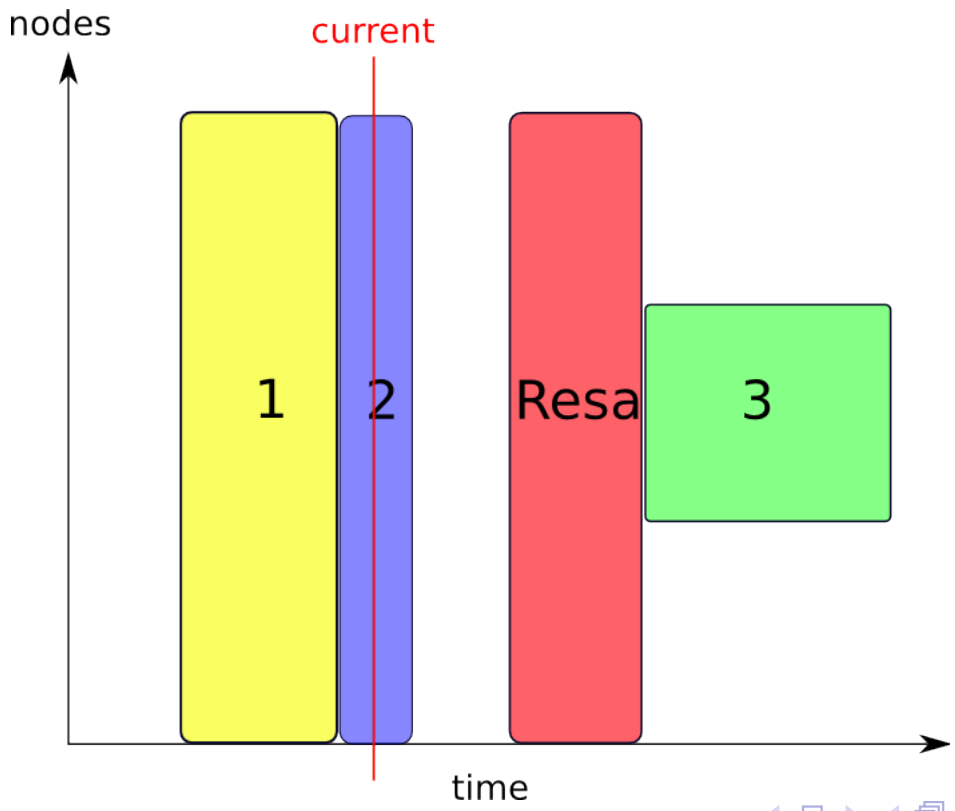


TimeSharing



Advance Reservation

- **Very useful** for demos, planification, multi-site tasks or grid tasks...
- **But :**
  - Restrictive for the scheduling
  - Resources are rarely used along the whole duration of the reservation (waste !)



OAR : Monika

OAR Cluster nodes

default summary

	Free	Busy	Total
network_address	4	15	32
resource_id	32	120	256

Reservations:

Reservations for property iru=0:

rli0n0	182270	182270	182270	182270	182270	182270	182270	182270
rli0n1	182270	182270	182270	182270	182270	182270	182270	182270
rli0n2	Free	Free	Free	Free	Free	Free	Free	Free
rli0n3	Free	Free	Free	Free	Free	Free	Free	Free
rli0n4	182267	182267	182267	182267	182267	182267	182267	182267
rli0n5	Free	Free	Free	Free	Free	Free	Free	Free
rli0n6	182271	182271	182271	182271	182271	182271	182271	182271
rli0n7	182271	182271	182271	182271	182271	182271	182271	182271
rli0n8	182271	182271	182271	182271	182271	182271	182271	182271
rli0n9	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy
rli0n10	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy
rli0n11	182294	182294	182294	182294	182294	182294	182294	182294
rli0n12	182282	182282	182282	182282	182282	182282	182282	182282

OAR : Gantt diagramm

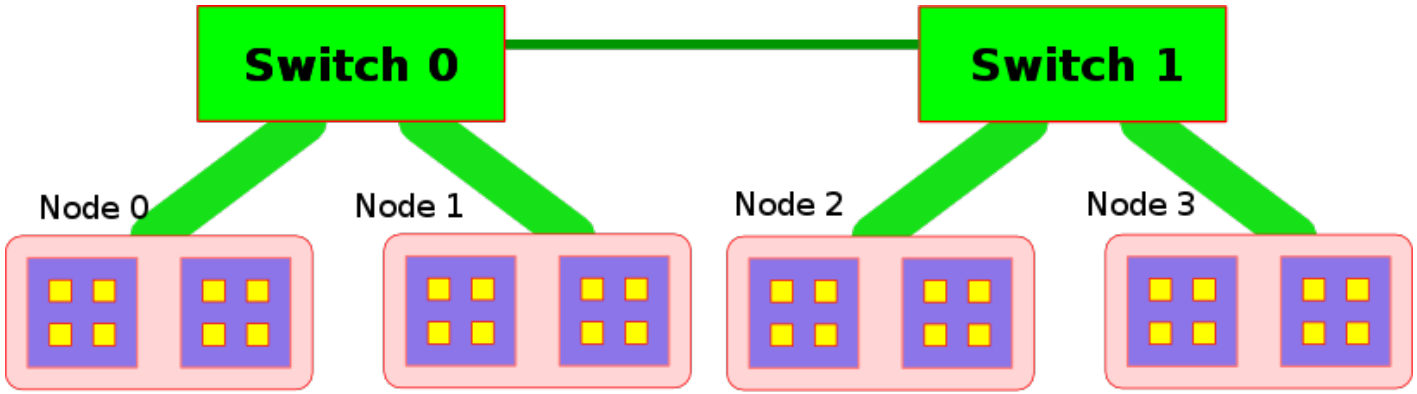
Research Issues

- Topological Constraints for improvement of application performance.
- Energy consumption mechanisms in HPC clusters

Topological constraints

Hardware evolution

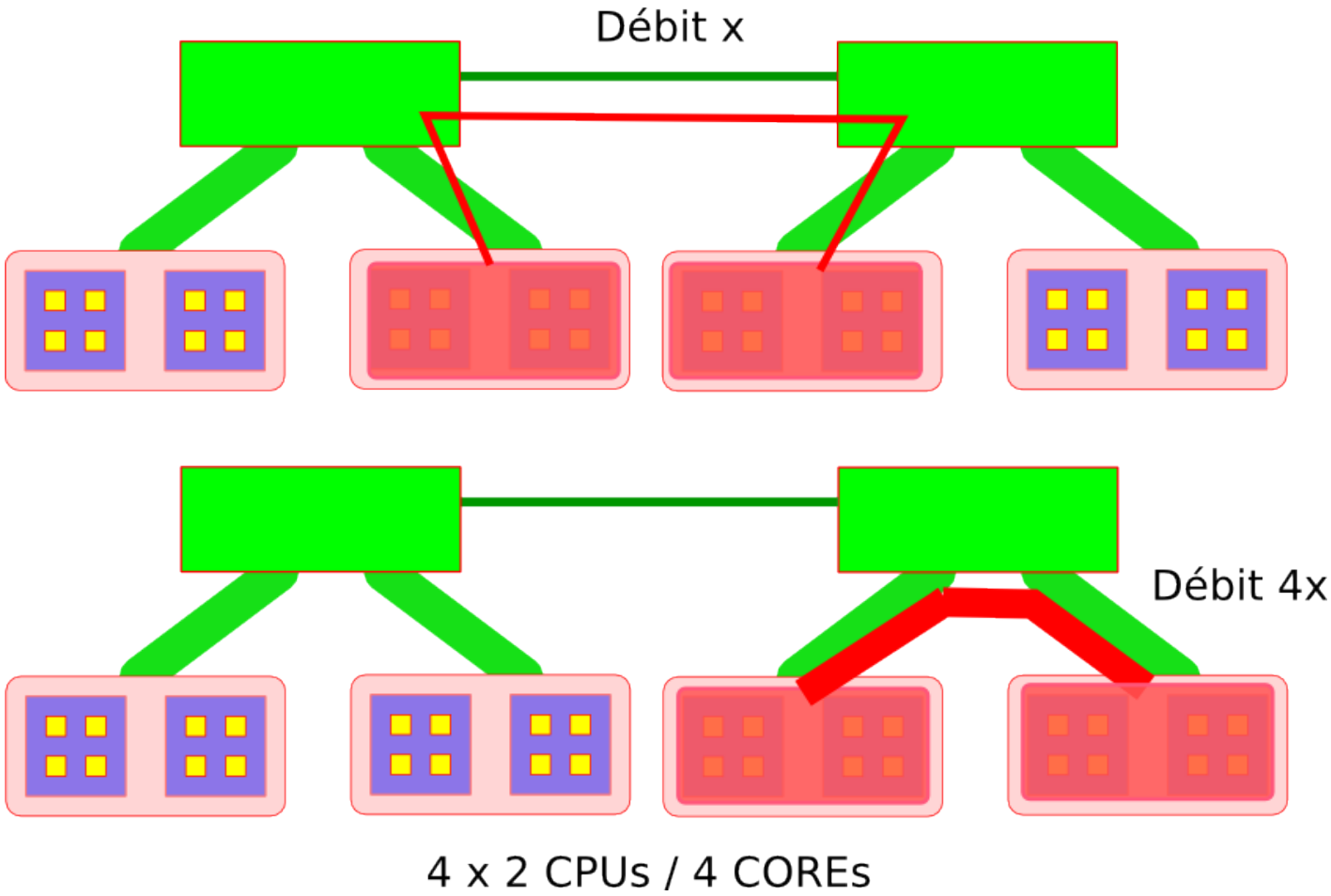
- switch/node/cpu/core : Hierarchical architecture
- NUMA host/ BlueGene host : 2D grid, 3D or hybrid ...



4 x 2 CPUs / 4 COREs

Hierarchical topological constraints

Problem with parallel applications that are network-bandwidth sensitive.

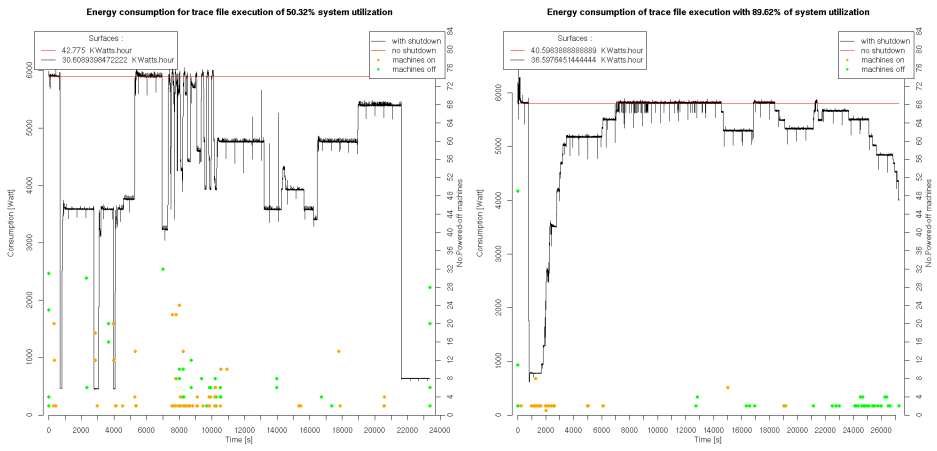


Hierarchical topological constraints

- Hierarchy in the requests :
  oarsub -l switch=1/nodes=2/cpu=2/core=2 ./my-app
  = 1x2x2x2 = 8 cores
- Linux cpusets management (be aware of the cpu affinity inside a cpuset)

Energy saving

- **We can shutdown nodes when not used (on-demand wake up)**
- Timetable priorities
- Work done during *Google Summer Of Code 2008* (Gsoc'08)
  - Parametrical job type : **powersaving** + options (cpufreq, disk shutdown, video ..., special policy)
  - Ex Job BestEffort → lowest CPU frequency.



▶ ...

Conclusion

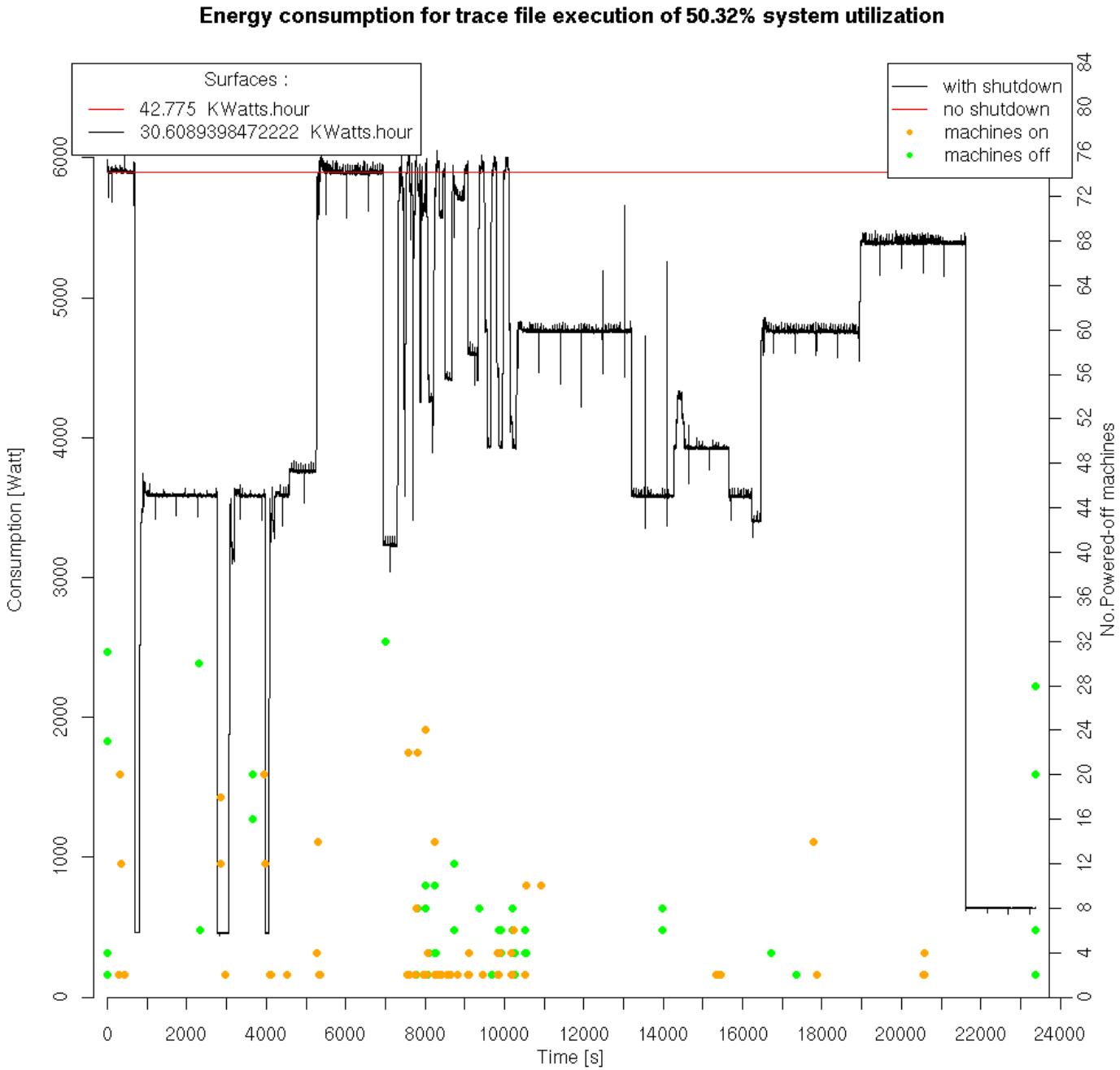
- Management of Resources and Jobs for high performance computing in modern architectures has become a complex procedure with interesting research issues.
- Opensource and commercial RJMS have been evolving to provide efficient exploitation of the infrastructures along with quality of services to the users.
- OAR versatile and customizable distributed RJMS.

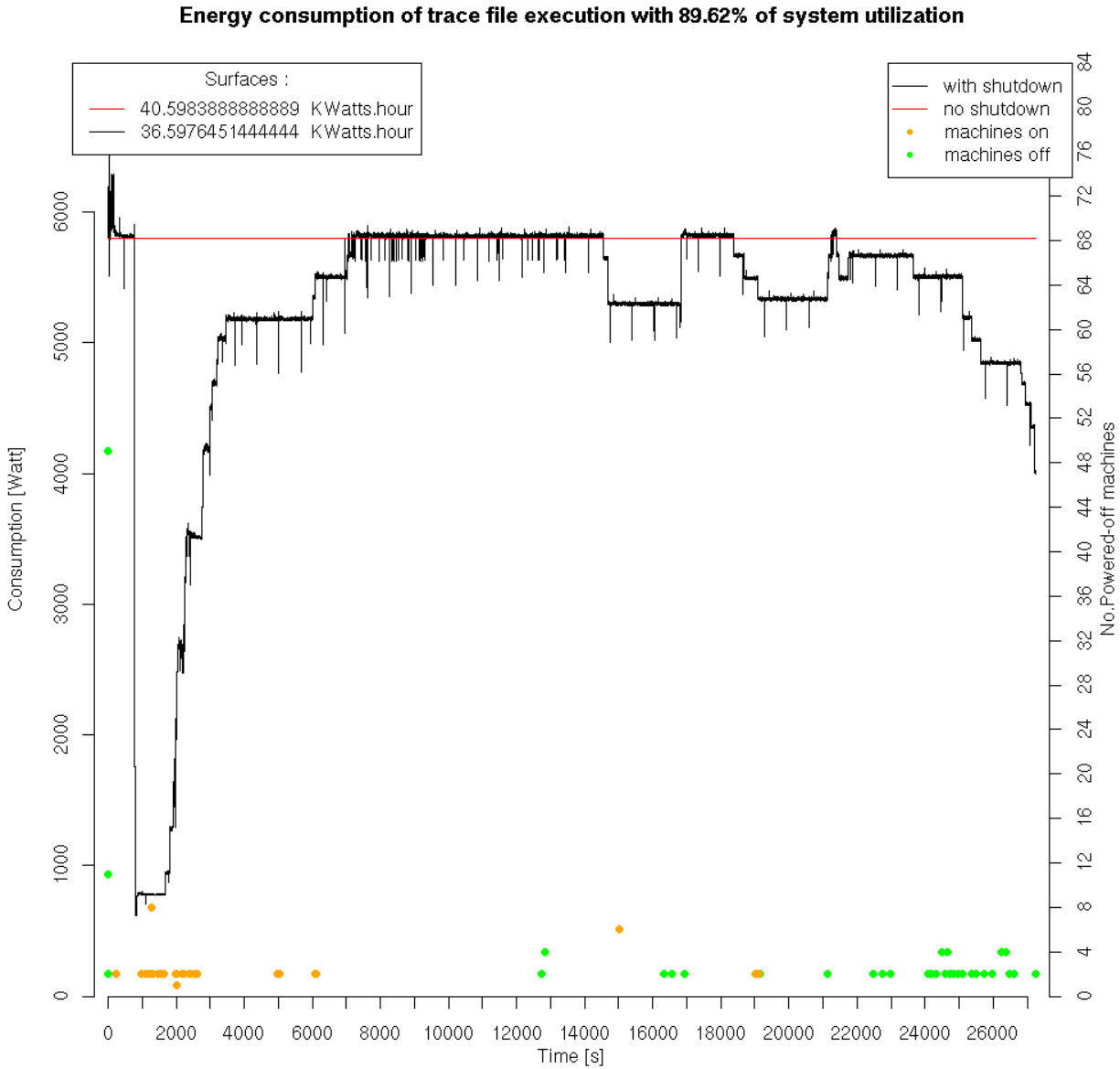
Questions ?

http ://oar.imag.fr/

Liens

- Condor
  http ://www.cs.wisc.edu/condor/
- Sun Grid Engine (SGE)
  http ://gridengine.sunsource.net
- TORQUE/MAUI
  http ://www.clusterresources.com/
- SLURM
  www.llnl.gov/linux/slurm/
- LSF
  http ://www.platform.com
- OAR
  http ://oar.imag.fr


Energy consumption for trace file execution of 50.32% system utilization
Surfaces :
42.775 KWatts.hour
30.6089398472222 KWatts.hour
with shutdown
no shutdown
machines on
machines off
Consumption [Watt]
No.Powered-off machines
Time [s]
0
1000
2000
3000
4000
5000
6000
0
4
8
12
16
20
24
28
32
36
40
44
48
52
56
60
64
68
72
76
80
84
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
24000


Energy consumption of trace file execution with 89.62% of system utilization
Surfaces :
40.5983888888889 KWatts.hour
36.5976451444444 KWatts.hour
with shutdown
no shutdown
machines on
machines off
Consumption [Watt]
No.Powered-off machines
Time [s]
0
1000
2000
3000
4000
5000
6000
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
24000
26000
0
4
8
12
16
20
24
28
32
36
40
44
48
52
56
60
64
68
72
76
80
84