

High Performance Computing

Resource and Job Management for HPC

Yiannis Georgiou

LIG / BULL

SC-CAMP - 13/07/2011



Summary

Introduction

- Cluster Computing
- Grid Computing
- Cloud Computing

About Resource and Job Management Systems

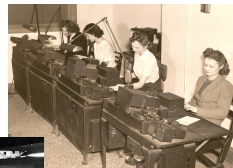
OAR a highly configurable RJMS

- Concepts and architecture
- Scheduling
- Interfaces

Conclusions and RJMS Research Challenges

Introduction

- ▶ Computational Science
 - ▶ Began during World War II
 - ▶ with Manhattan Project and design of nuclear weapons



- ▶ ... from Calculators to Computers



- ▶ ... and the first supercomputers



Introduction

High Performance Computing is defined by:

Infrastructures:

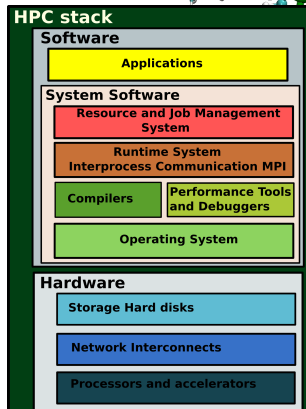
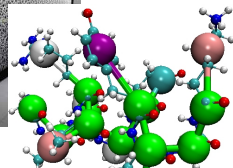
- ▶ Supercomputers, Clusters, Grids, Peer-to-Peer Systems and lately Clouds

Applications:

- ▶ Climate Prediction, Protein Folding, Crash simulation, High-Energy Physics, Astrophysics, Animation for movie and video game productions

System Software

- ▶ System Software: Operating System, Runtime system, Resource Management, I/O Systems, Interfacing to External Environments



High Performance Computing Evolutions

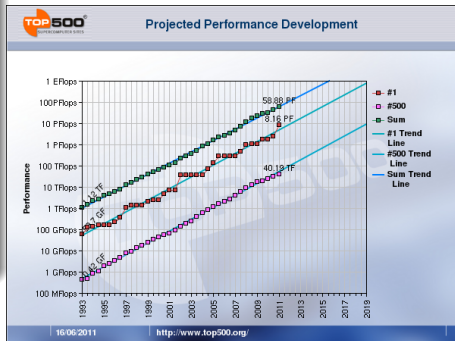
Computational infrastructures

Advances in computer architecture (Top 500 supercomputer sites)

- ▶ multiprocessor technologies: 85% quadcore (2010) up from 1% dualcore (2005)
- ▶ Use of Clusters 80% (2010) up from 42% (2003)
- ▶ Peak performance: 1st position From GigaFlops (1997) to Petaflops (2008)
- ▶ increase of energy consumption: average 397 kWatt (2010) up from 257 kwatt (2008)

Scientific Applications

- ▶ Need for more computing power
- ▶ quality of services
- ▶ deep parallelism
- ▶ fault tolerance



Plan

Introduction

Cluster Computing

Grid Computing

Cloud Computing

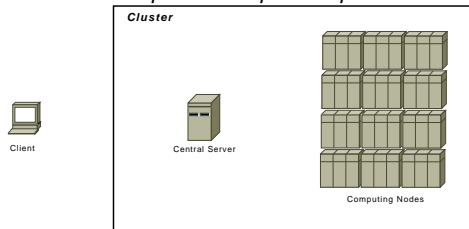
About Resource and Job Management Systems

OAR a highly configurable RJMS

Conclusions and RJMS Research Challenges

Definition Cluster

In our context (HPC) a cluster is a set of n compute *nodes* that are interconnected in a way that allows simultaneous execution of several *sequential* or *parallel jobs*. A cluster is also called *parallel supercomputer*.



Concepts Cluster Computing

Process

Processes are running 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.

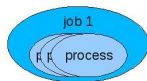


Concepts Cluster Computing

Jobs

Processes can be grouped 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.

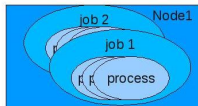


Concepts Cluster Computing

Nodes

Jobs are running on nodes.

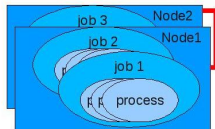
- ▶ A *node* is a computer having a set of p CPU, an amount of memory, one or several network interfaces and that may have a storage unit (local disk).



Concepts Cluster Computing

Computing network

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

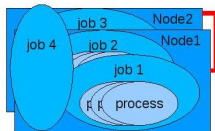


Concepts Cluster Computing

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



Concepts Cluster Computing

Types of jobs

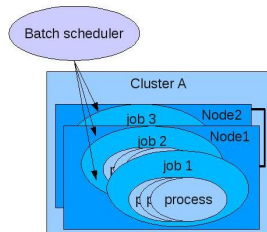
The Jobs can come at the form of Batch or Interactive type.

- ▶ A *batch* job is a script. A shell script is a given list of shell commands to be executed in a given order
- ▶ Today's shells are so sophisticated that you can create scripts that are real programs with some variables, control 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.



Plan

Introduction

Cluster Computing

Grid Computing

Cloud Computing

About Resource and Job Management Systems

OAR a highly configurable RJMS

Conclusions and RJMS Research Challenges

The grid concept



- ▶ Comes from the "power grid" concept
- ▶ In a power grid, there are several energy sources and the ending user consumes a part of that energy without knowing exactly where it has been produced.
- ▶ In a computing grid, there are several computing hosts and the ending user launches tasks that will run on some of them without knowing exactly where.

The grid concept

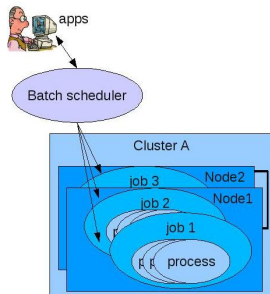


- ▶ Well...
- ▶ Computing tasks are a bit more complicated than a simple electrical flow :-)
 - ▶ Application code dependency
 - ▶ Input data dependency
 - ▶ I/O data amount
 - ▶ Duration
 - ▶ Type of code: parallel/sequential
 - ▶ ...

From the process to the grid

Job submission

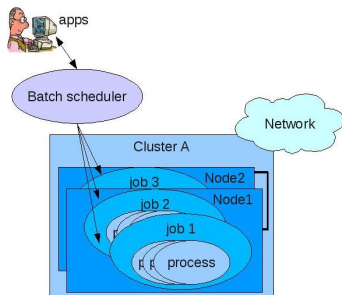
Users submit jobs to the batch scheduler.



From the process to the grid

Public network

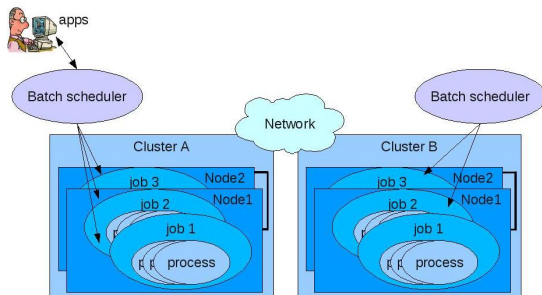
A cluster frontend maybe connected to a public network, generally not the same network as the private computing network.



From the process to the grid

Public network

Clusters frontend maybe interconnected.

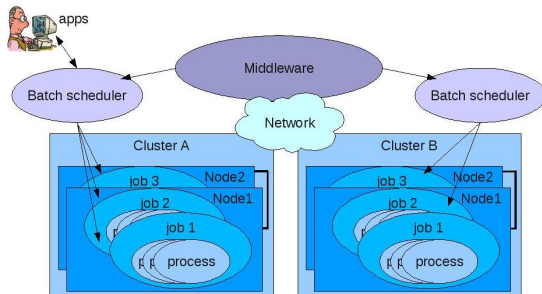


From the process to the grid

Computing Grids

They are often composed by multiple loosely coupled and geographically dispersed clusters with different administrative policies.

- ▶ Specialised software, termed as *grid middleware*, are used for the monitoring, discovery, and management of resources in order to promote the application execution upon the grid.
- ▶ At this level a collaboration between the local cluster resource and job management system and the grid middleware is needed.



From the process to the grid

Grid middleware

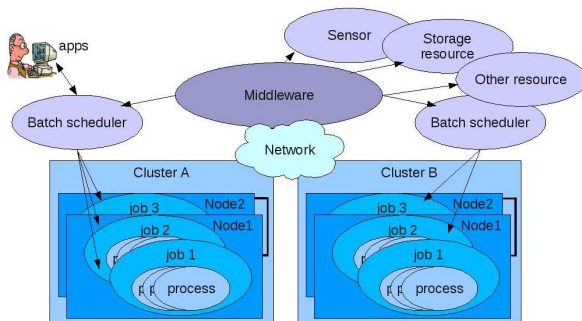
The middleware is the component that acts between the different grid resources and the users' applications.

- ▶ It may be very complex and composed of elements with a specific global interaction for the given grid.
- ▶ The middleware gives a uniform access to heterogeneous grid resources
- ▶ It manages and allocates the grid resources (clusters availability, load and properties, storage services,...)
- ▶ It manages with authentication and confidentiality
- ▶ It may offer visualization and monitoring tools
- ▶ Examples: Globus, UNICORE, gLite, CiGri...

From the process to the grid

Grid middleware

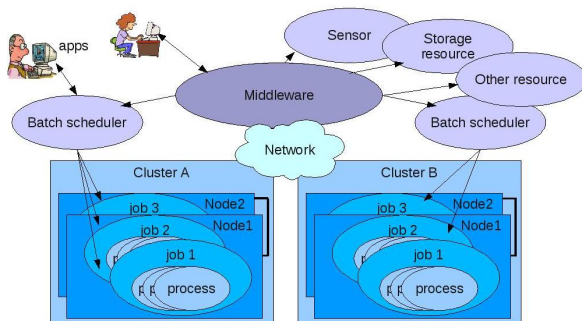
The middleware may be responsible of the communication with other external elements to the grid: sensors, storage, etc



From the process to the grid

Grid job submission

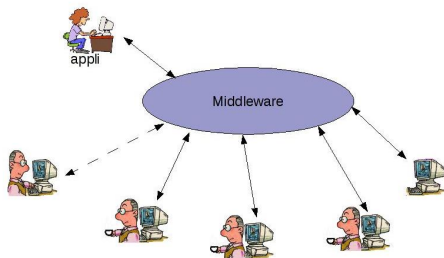
The users interact with the grid through the middleware, for submitting grid jobs for example.



Alternative Computing Grids

Desktop/volunteer computing

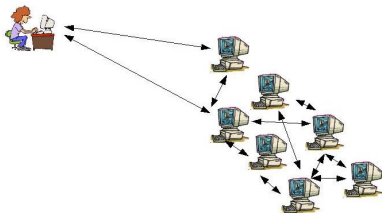
But a grid may also look like this...



Alternative Computing Grids

Peer-to-peer grid

...or like this...



Grid definitions

Wikipedia

"Grid computing (or the use of computational grids) is the combination of computer resources from **multiple administrative domains** applied to a **common task**, usually to a scientific, technical or business problem that requires a **great number of computer processing cycles** or the need to process **large amounts of data**."

Grid definitions

The Grid, I. Foster, C. Kesselman, 1998

"A computational grid is a **hardware and software** infrastructure that provides dependable, consistent, pervasive, and **inexpensive** access to high computational capabilities."

Grid definitions

The CERN dream: The grid

"[...] Now imagine that all of these computers can be connected to form **a single, huge and super-powerful computer!** This huge, sprawling, global computer is what many people dream "The Grid" will be."

Middleware examples

- ▶ The GLOBUS Toolkit <http://www.globus.org> (EGEE, National Virtual Observatory,...)
- ▶ gLite: Globus based (EGEE)
- ▶ UNICORE (DEISA)
- ▶ Oargrid, kadeploy and... ssh (Grid5000)
- ▶ CiGri (CIMENT)
- ▶ Boinc (*@home)
- ▶ CONDOR-G: globus based
- ▶ ARC: Globus based (Nordunet)
- ▶ eMule
- ▶ XtremOS
- ▶ XWHEP

Plan

Introduction

Cluster Computing

Grid Computing

Cloud Computing

About Resource and Job Management Systems

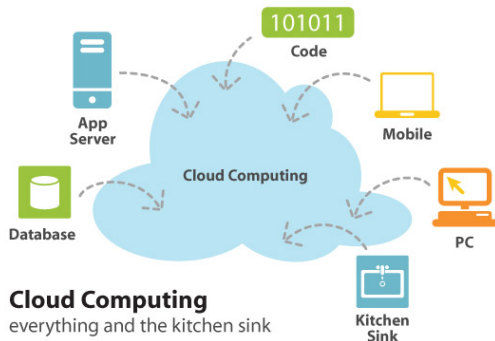
OAR a highly configurable RJMS

Conclusions and RJMS Research Challenges

Cloud Definition

Cloud computing

is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).



Cloud computing Concepts

A cloud service has three distinct characteristics:

- ▶ It is sold **on demand**, typically by the minute or the hour;
- ▶ it is **elastic** – a user can have as much or as little of a service as they want at any given time;
- ▶ and the service is fully **managed by the provider** (the consumer needs nothing but a personal computer and Internet access).

Cloud computing

- ▶ The idea is that you can use an application or manage data through services without knowing where they are (somewhere in the cloud)
- ▶ It's related to an economical model where clients pay for services without worrying about the infrastructure
- ▶ Directly related to **grid computing** and **virtualization** (you may rent an OS running somewhere in the cloud)
- ▶ Also related to scalability: the infrastructure adapts to exactly what you need

Cloud computing examples

- ▶ Amazon EC2 (virtual hosts) and S3 (online storage web service)
- ▶ GoGrid
- ▶ iCloud (free!)
- ▶ Google apps
- ▶ eyeOs

Plan

Introduction

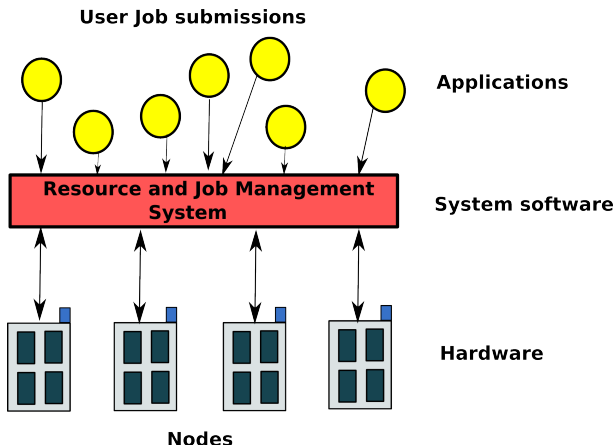
About Resource and Job Management Systems

OAR a highly configurable RJMS

Conclusions and RJMS Research Challenges

Resource and Job Management

The goal of a Resource and Job Management System (RJMS) is to satisfy users' demands for computation and assign user jobs upon the computational resources in an **efficient manner**.



RJMS Importance

Strategic position but complex internals:

- ▶ **Direct and constant knowledge** of resources and jobs
- ▶ **Multifacet procedures** with complex internal functions

Concepts Resource and Job Management System

This assignment 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.

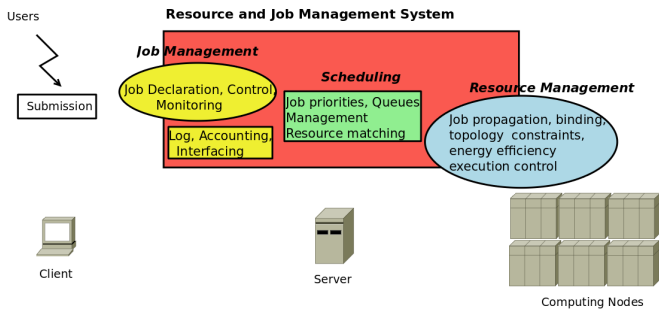
In this sense, the work of a RJMS can be decomposed into three main subsystems: Job Management, Scheduling and Resource Management.

RJMS principal concepts

RJMS subsystems	Principal Concepts	Advanced Features
<u>Resource Management</u>	<ul style="list-style-type: none"> -Resource Treatment (hierarchy, partitions,..) -Job Launching, Propagation, Execution control -Task Placement (topology, binding,..) 	<ul style="list-style-type: none"> - High Availability - Energy Efficiency - Topology aware placement
<u>Job Management</u>	<ul style="list-style-type: none"> -Job declaration (types, characteristics,..) -Job Control (signaling, reprioritizing,..) -Monitoring (reporting, visualization,..) 	<ul style="list-style-type: none"> - Authentication (limitations, security,..) - QOS (checkpoint, suspend, accounting,..) - Interfacing (MPI libraries, debuggers, APIs,..)
<u>Scheduling</u>	<ul style="list-style-type: none"> -Scheduling Algorithms (builtin, external,..) -Queues Management (priorities, multiple,..) 	<ul style="list-style-type: none"> - Advanced Reservation - Application Licenses

RJMS General organization

- ▶ A central host
- ▶ Client programs (command line) to interact 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
- ▶ ...

Used Resource and Job Management Systems

Open Source RJMS

- ▶ SLURM
- ▶ TORQUE
- ▶ MAUI
- ▶ OAR
- ▶ CONDOR
- ▶ SGE (before Oracle)

Commercial RJMS

- ▶ Loadleveler
- ▶ LSF
- ▶ MOAB
- ▶ PBSPro
- ▶ OGE (Oracle Grid Engine)

Used Resource and Job Management Systems

Open Source RJMS

- ▶ SLURM
- ▶ TORQUE
- ▶ MAUI
- ▶ OAR
- ▶ CONDOR
- ▶ SGE (before Oracle)

Commercial RJMS

- ▶ Loadleveler
- ▶ LSF
- ▶ MOAB
- ▶ PBSPro
- ▶ OGE (Oracle Grid Engine)

RJMS Comparison study

- ▶ Quantifiable Functionalities Evaluation of opensource and commercial RJMS

RJMS Quantifiable Functionalities Comparison

Quantifying Functionalities support by RJMS

- ▶ Resource Management
 - ▶ Resources Treatment, Job Launching, Task Placement, High Availability,...
- ▶ Job Management
 - ▶ Job declaration, Job Control, Monitoring, Interfacing, Quality of Services,...
- ▶ Scheduling
 - ▶ Scheduling Algorithms, Queues Management, Advanced Reservations,...

Overall Evaluation / RJMS Software	SLURM	CONDOR	TORQUE	OAR	MAUI	LSF
Resource Management (/10)	7.1	5.2	5.2	6.9	1.9	6.9
Job Management (/10)	5.1	6.5	5.1	5.5	3.1	6.8
Scheduling (/10)	6	5.3	3	5.7	5.5	5.7
<i>Overall Evaluation Points (/10)</i>	6.2	5.7	5.1	6	3.4	6.4

RJMS Quantifiable Functionalities Comparison

Quantifying Functionalities support by RJMS

- ▶ Resource Management
 - ▶ Resources Treatment, Job Launching, Task Placement, High Availability,...
- ▶ Job Management
 - ▶ Job declaration, Job Control, Monitoring, Interfacing, Quality of Services,...
- ▶ Scheduling
 - ▶ Scheduling Algorithms, Queues Management, Advanced Reservations,...

Overall Evaluation / RJMS Software	SLURM	CONDOR	TORQUE	OAR	MAUI	LSF
Resource Management (/10)	7.1	5.2	5.2	6.9	1.9	6.9
Job Management (/10)	5.1	6.5	5.1	5.5	3.1	6.8
Scheduling (/10)	6	5.3	3	5.7	5.5	5.7
Overall Evaluation Points (/10)	6.2	5.7	5.1	6	3.4	6.4

Plan

Introduction

About Resource and Job Management Systems

OAR a highly configurable RJMS

- Concepts and architecture



- Scheduling

- Interfaces

Conclusions and RJMS Research Challenges

Objectives

OAR has been designed with **versatility** 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).

The logo for CIMENT, featuring the word "CIMENT" in white capital letters on a dark blue rectangular background, with a small graphic of a brick or stone to the right.

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 *CiGr*)
- ▶ **Multiple task types** (besteffort, deploy, timesharing, idempotent, power, cosystem ...) (customizable)
- ▶ **Moldable tasks**
- ▶ **Energy saving**

Plan

Introduction

About Resource and Job Management Systems

OAR a highly configurable RJMS

- Concepts and architecture

- Scheduling

- Interfaces

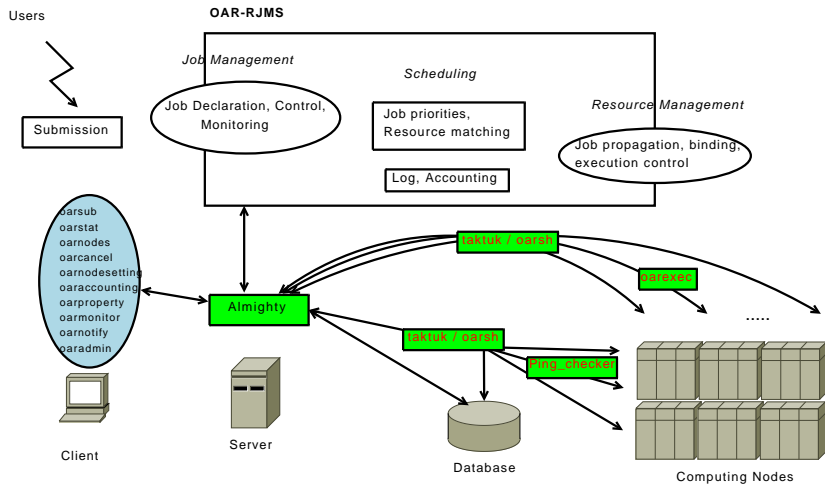
Conclusions and RJMS Research Challenges

OAR: design concepts

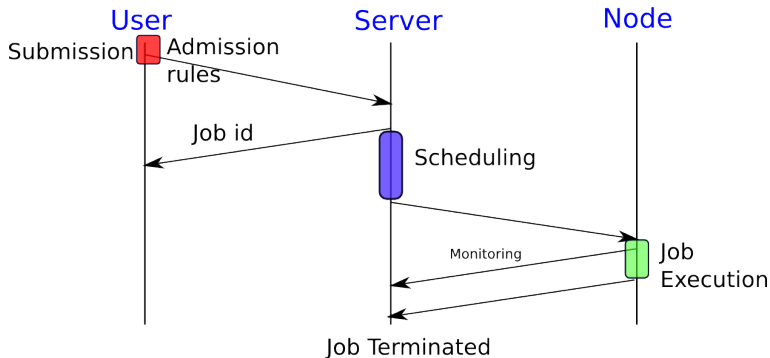
High level components use

- ▶ **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 structures (lists, hash tables, sort...)
 - ▶ Short development cycles
- ▶ **Other components**: SSH, CPUSSETS, Taktuk
 - ▶ **SSH, CPUSSET** (isolation, cleaning)
 - ▶ **Taktuk** adaptative parallel launcher

OAR : general organisation



Job life cycle

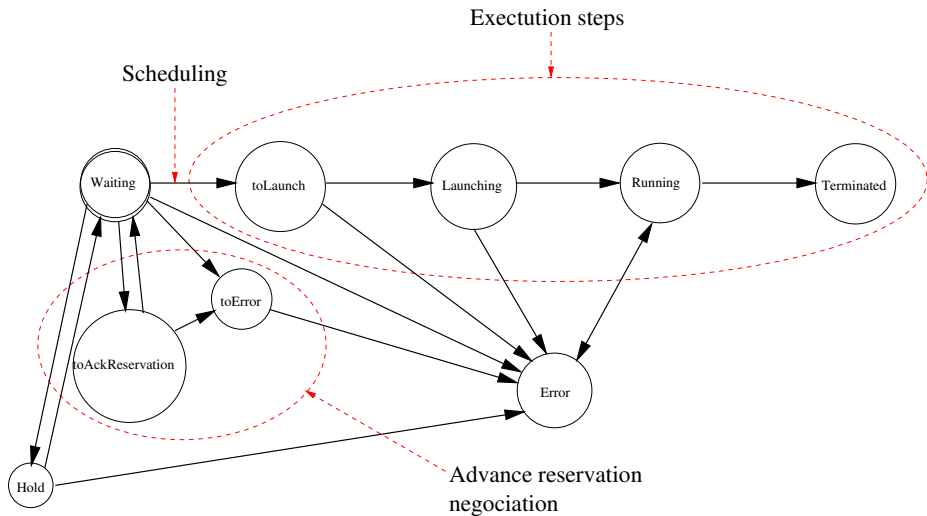


Admission rules

Very important customization point

- ▶ High *customization* offered to the administrator
- ▶ manage the requests' scope
- ▶ default values fixing: walltime, queue, number of resources,...
- ▶ access control (users, groups, timetable...)

Task status diagram



Submission examples: OAR

Interactive task: ¹

- ▶ **oarsub -l 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:

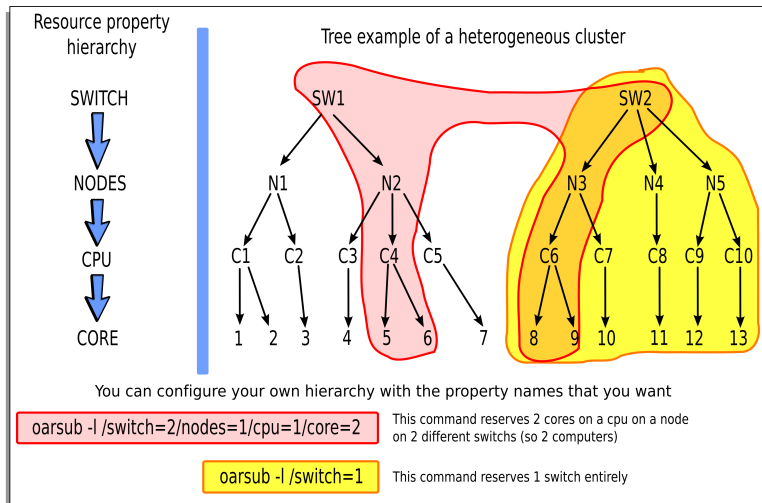
- ▶ **oarsub -r "2008-04-27 11:00" -l nodes=12**

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

- ▶ **oarsub -C 154**

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

Hierarchical Resources



Plan

Introduction

About Resource and Job Management Systems

OAR a highly configurable RJMS

- Concepts and architecture

- Scheduling

- Interfaces

Conclusions and RJMS Research Challenges

Scheduling

Scheduling is the step ² at which the system selects the **ressources to give** to tasks **and starting dates**.

Scheduling is made following a **policy** that is defined by a **scheduling algorithm**.

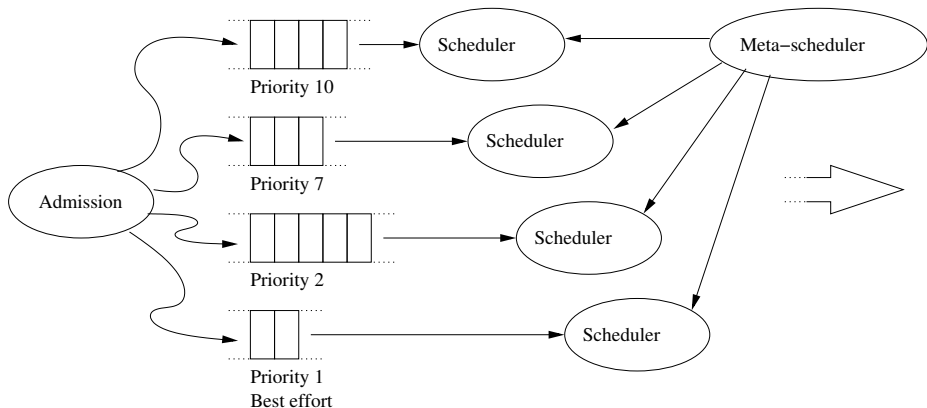
Furthermore numerous **parameters** are used to guide and scope the allocations and priorities.

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 has a priority level
- ▶ each queue obey to a scheduling policy



Resource matching

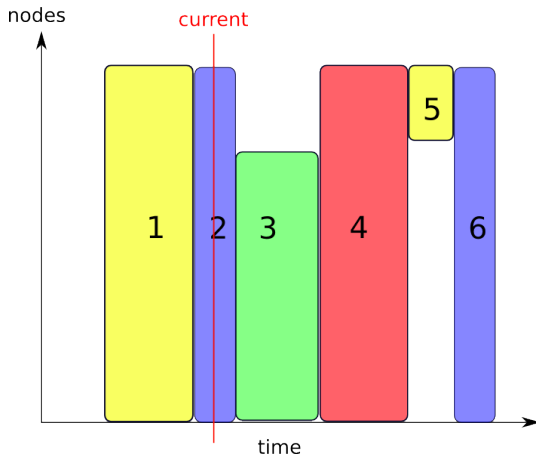
A preliminary step to scheduling

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

Scheduling policies

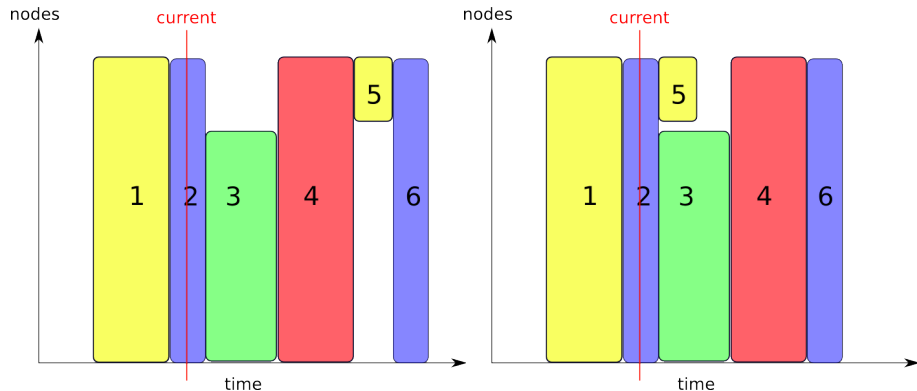
- ▶ FIFO (First-In First-Out)
- ▶ First-Fit (Backfilling)
- ▶ FairSharing
- ▶ Timesharing

FIFO: First-In First-Out



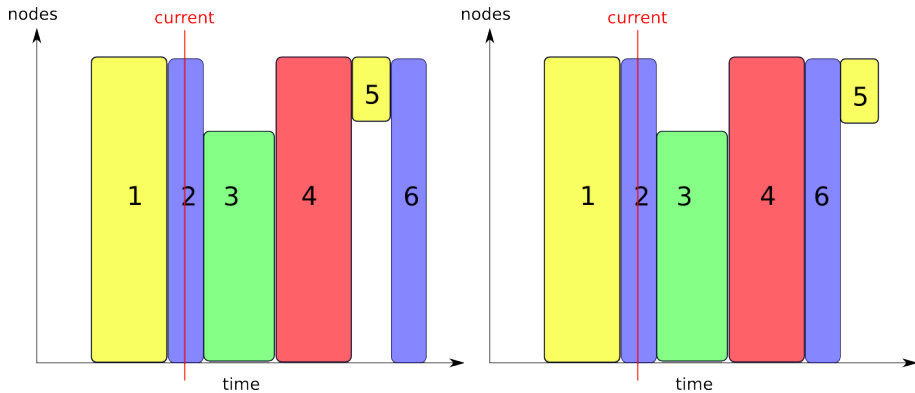
First-Fit (Backfilling)

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

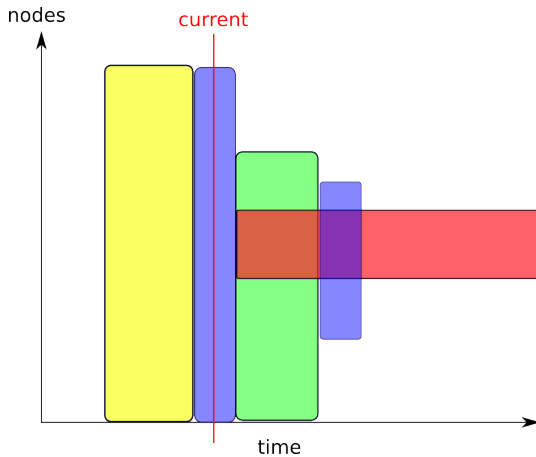


FairSharing

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.

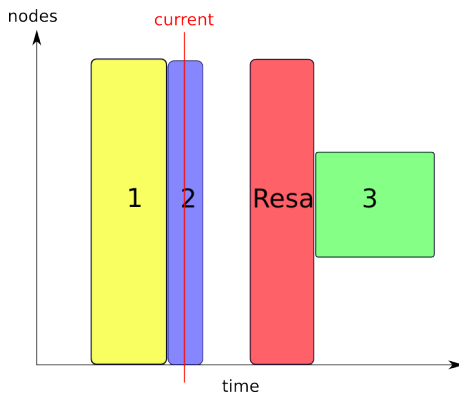


TimeSharing



Advanced 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!)



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

Plan

Introduction

About Resource and Job Management Systems

OAR a highly configurable RJMS

- Concepts and architecture

- Scheduling

- Interfaces

Conclusions and RJMS Research Challenges

OAR: Monika

OAR Cluster nodes

<i>default summary</i>			
	Free	Busy	Total
network_address	4	15	32
resource_id	32	120	256

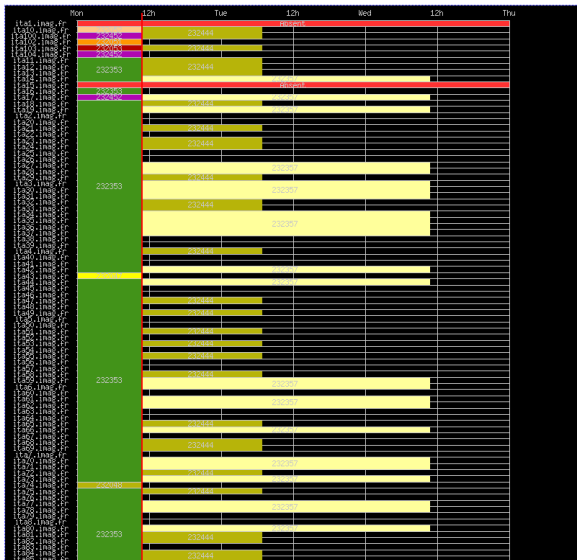
Reservations:

Reservations for property iru=0:

r10n0	182270	182270	182270	182270	182270	182270	182270	182270
r10n1	182270	182270	182270	182270	182270	182270	182270	182270
r10n2	Free	Free	Free	Free	Free	Free	Free	Free
r10n3	Free	Free	Free	Free	Free	Free	Free	Free
r10n4	182267	182267	182267	182267	182267	182267	182267	182267
r10n5	Free	Free	Free	Free	Free	Free	Free	Free
r10n6	182271	182271	182271	182271	182271	182271	182271	182271
r10n7	182271	182271	182271	182271	182271	182271	182271	182271
r10n8	182271	182271	182271	182271	182271	182271	182271	182271
r10n9	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy
r10n10	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy	StandBy
r10n11	182294	182294	182294	182294	182294	182294	182294	182294
r10n12	182282	182282	182282	182282	182282	182282	182282	182282

OAD: Gantt diagramm

Origin 2005 Oct 17 00:00 Range 3 days BestEffort Draw Default



Plan

Introduction

About Resource and Job Management Systems

OAR a highly configurable RJMS

Conclusions and RJMS Research Challenges

Conclusion

- ▶ **Versatile and customizable**
- ▶ Basic Features like other RJMS
- ▶ Advanced Features like besteffort jobs, hierarchical resources, etc...
- ▶ Current stable version: **2.5.0** (.tgz, .deb, .rpm)
- ▶ **Resource and Job Management** for HPC is a continuously evolving domain.



Launching and Scheduling

- ▶ Launching and propagation techniques through socket communications or ssh based approaches... optimizations by using **dynamically adapted deployment** based upon tree structures.
- ▶ The scheduler is the central point of intelligence of the RJMS system. The more advanced features are supported upon the RJMS the more complex will be the process of scheduling. Network topology characteristics, energy efficient resource management, fault-tolerance techniques.
- ▶ **Advanced Scheduling policies** may optimize system utilization under specific contexts

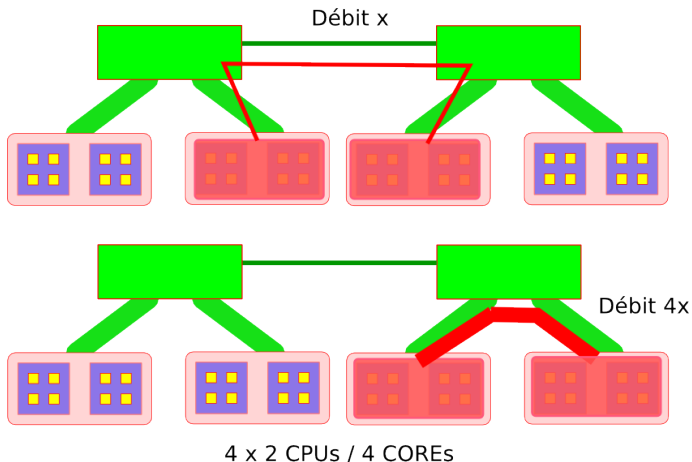
Internal node Topology constraints

- ▶ Internal node topology increasingly complex
- ▶ Applications using OpenMP, MPI or hybrid models (OpenMP+MPI) have to be carefully placed upon the machines so that hardware affinities are efficiently handled for optimal performance.
- ▶ Shared-memory or synchronization between tasks benefits from shared caches, while intensive memory access benefits from local memory allocations.
- ▶ Avoid **Internal fragmentation** which is the phenomenon that results into idle processors when more CPUs are allocated to a job, that it requests.
- ▶ Placement and binding of tasks upon cores for best performance

Hence, ideally the RJMS: 1) **automatic optimal** task placement techniques 2) **specific placement** considering the application profiling

Network Hardware Topology constraints

- ▶ Different kind of topologies: Hierarchical, 2D grid, 3D or hybrid
- ▶ Problem with parallel applications that are network-bandwidth sensitive.
- ▶ **Bisection bandwidth** constraints



Energy Efficient Resource Management

Automatic actions for unutilized resources

- ▶ shutdown, standby techniques
- ▶ Dynamic Voltage and Frequency Scaling techniques

Thermodynamic controls

- ▶ temperature and sensors data
- ▶ selection of particular nodes based upon those information

Workload Prediction for intelligent shutdown and power on mechanisms

Questions ?



<http://oar.imag.fr/>

Links



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>