

# Computing grids

SC-CAMP

17 august 2010



# Grid computing

## 1 Definitions

- The grid concept
- From the process to the grid
- Grid definitions

## 2 Grid classification and examples

- by objective
- by infrastructure
- middlewares
- around the grids

## 3 CIGRI Lightweight Grid

- CIGRI grid system : Principal Concepts
- The CIGRI middleware
- CIGRI Fault Treatment

# History

- Grid... a fashion ? No more, now the fashion is "cloud" computing :-)
- The term "grid computing" was introduced by Ian Foster in the early 1990
- Very popular in late 1990 with Seti@home and Napster
- In France : ACI grid started in 2001

# 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

## Process

Processes are running on CPUs.



# From the process to the grid

## Jobs

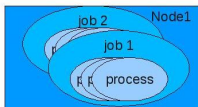
Processes can be grouped into jobs.



# From the process to the grid

## Nodes

Jobs are running on nodes. Nodes are computers (one or several cpus, a shared memory space, and i/o device).

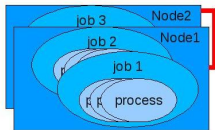




# From the process to the grid

## Computing network

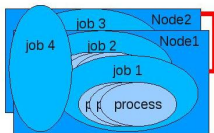
Several nodes are connected to a computing network, generally low latency network (Myrinet, Infiniband, Numalink,...)



# From the process to the grid

## Parallel jobs

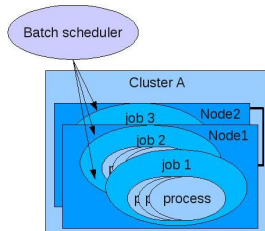
Jobs may be "parallel" or "sequential". A parallel job runs on several nodes, using the computing network to communicate.



# From the process to the grid

## Clusters

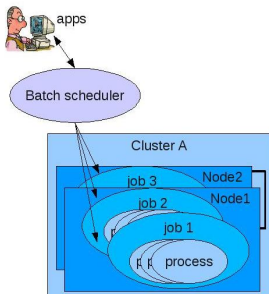
A batch scheduler is managing jobs and nodes. We have a cluster.



# From the process to the grid

## Job submission

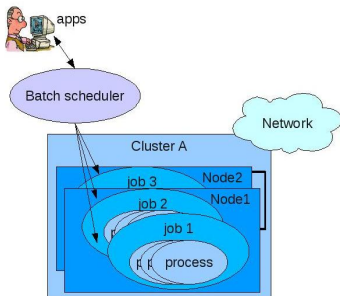
Users submit jobs to the batch scheduler.



# From the process to the grid

## Public network

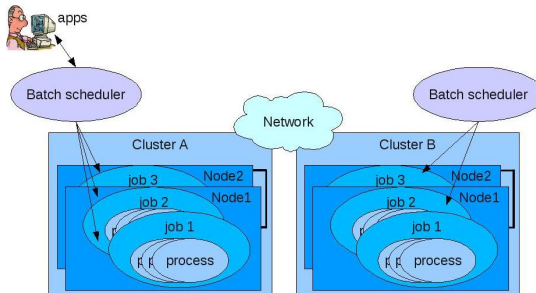
A cluster frontend maybe connected to a public network, generally not the same network as the private computing network.



# From the process to the grid

## Public network

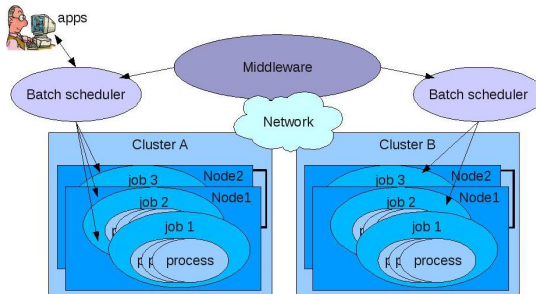
Clusters frontend maybe interconnected.



# From the process to the grid

## Computing Grids

...and then we have a computing grid



# 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

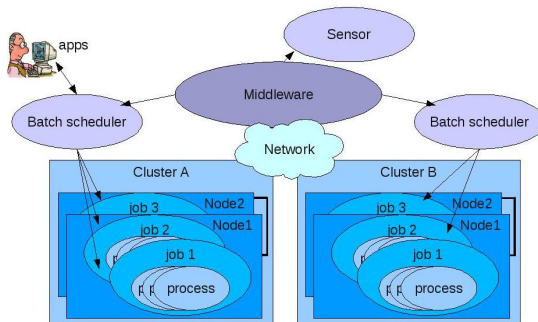
The middleware is the component that acts between the different grid resources and the users' applications.

- 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
- Exemples : Globus, UNICORE, gLite, CiGri...

# From the process to the grid

## Grid middleware

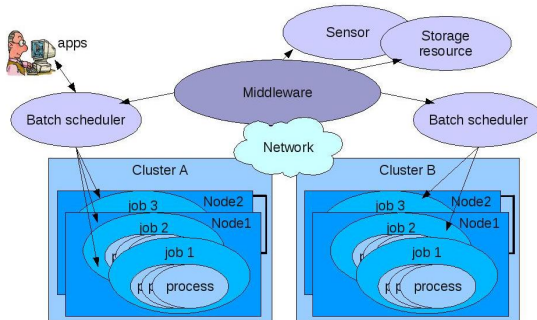
The middleware may be responsible of the communication with other external elements to the grid : sensors



# From the process to the grid

## Grid middleware

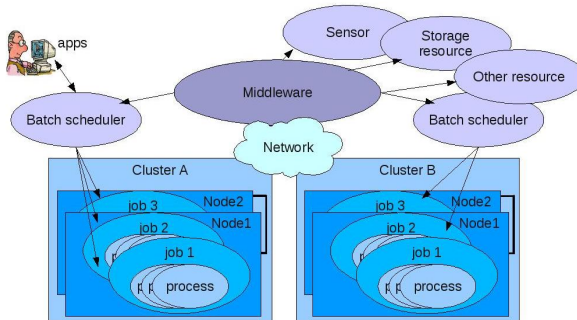
The middleware may be responsible of the communication with other external elements to the grid : sensors, storage



# From the process to the grid

## Grid middleware

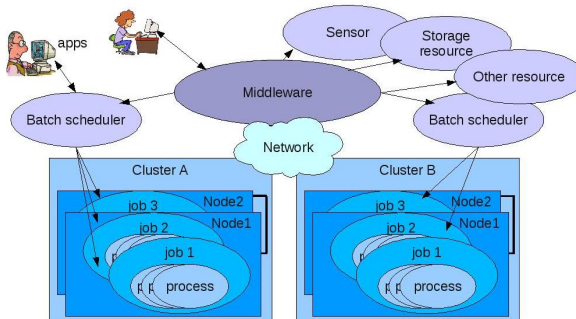
The middleware may be responsible of the communication with other external elements to the grid : sensors, storage, etc



# From the process to the grid

## Grid job submission

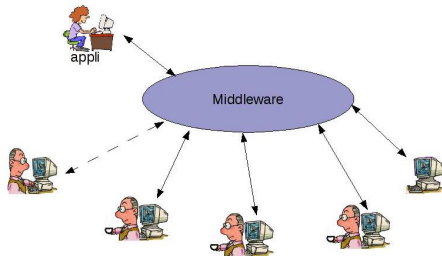
The users interact with the grid through the middleware, for submitting grid jobs for example.



# Alternative Computing Grids

## Desktop/volunteer computing

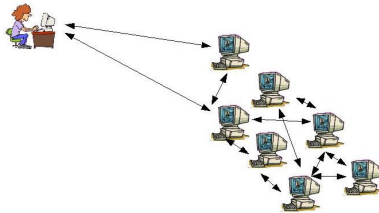
But a grid may also look like this...



# Alternative Computing Grids


## Peer-to-peer grid

...or like this...



# Grid definitions

## Common mix-up

- *Grid*  $\neq$  *Cluster*
- *Grid*  $\neq$  Cluster of clusters (a grid can't be constructed by simply nesting batch schedulers) 



# 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."

## So... When ?

Regarding this panel of point of views,

- when you need more resources than what you can have in one unique place (because of power, conditioned air, area, administrative reasons,...)
- when you want to optimize computers or supercomputers that are not used all the time
- when you have an application that has several paralelism levels and that we easily imagine to naturally use several supercomputers (**code coupling**)
- when it is the cheapest solution for the same service

a grid may be anything you can imagine for performing large scale distributed computations !

## What for ? (usage examples)

- Physics : Data analysis of experiments upon a scientific instrument like LHC (Large Hardon Collider). Huge volume of data nearly 15Petabytes per year.
- Health : Data base and analysis of millions of mammogrammes distributed upon various sites.
- Industry : Optimization of production control based upon particular algorithms : creation of solutions through millions of values and multiple parameters to take into account
- Environment : Climate modelization, weather prediction
- Astrophysics, Chemistry, Biology...
- ...

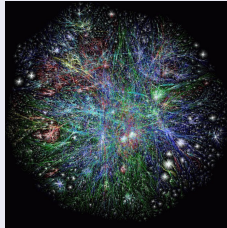
# Research Challenges

- Standards (OGSA, SAGA, DRMAA, GLUE, GRIDRPC,...)  
Open Grid Forum (<http://www.ogf.org>)
- Heterogeneous resources
- Data management (amount, synchronization, distribution,...)
- Security and privacy (authentication, encryption,...)
- Availability (monitoring, redundancy,...)
- Sharing (priority, accounting, fairsharing,...)
- Networking performance
- Organization : who is the administrator of the grid ?
- Applications : how to gridify an application ? or how to make the grid transparent to the applications ?

# classification by objectives

## Information grid

To share knowledge



Examples :

- World Wide Web
- Virtual observatory <http://www.france-ov.org>

# classification by objectives

## Data storage grid

High scale data storage



Examples :

- LCG (15 Petabytes/an, over EGEE)  
<http://lcg.web.cern.ch/LCG/>
- eMule (eDonkey)
- Bittorrent (the grid and it's middleware are merged...)



# classification by objectives

## Computing grid

Computing power aggregation

Examples :

- EGEE <http://www.eu-egee.org>
- DEISA <https://www.deisa.org>
- CIMENT Grid <https://ciment.ujf-grenoble.fr>
- SETI@home, Folding@home, Decryphon

# classification by objectives

## Experimentation grids

Distributed computing research

Examples :

- Grid5000 <http://www.grid5000.fr>
- PlanetLab <http://www.planet-lab.org>
- DAS3 <http://www.cs.vu.nl/das3>
- XtremLab <http://xtremlab.lri.fr>
- NAREGI [http://www.naregi.org/index\\_e.html](http://www.naregi.org/index_e.html)

# classification by infrastructure

## Institutional grid

Generally grids of clusters, with stable and secured nodes.

Examples :

- EGEE <http://www.eu-egee.org>
- Grid5000 <http://www.grid5000.fr>
- CIMENT Grid <https://ciment.ujf-grenoble.fr>

# classification by infrastructure

## Desktop and volunteer computing

A lot of nodes (millions), volatiles and not secured

Examples :

- XtremLab
- SETI@home, Folding@home, Decryphon
- eMule, bittorrent
- Computemode
- XWHEP

# 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

# Cloud computing

- A more recent term to design something less specific than grids
- 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
- Often related to virtualization (you may rent an OS running somewhere in the cloud)
- Also related to scalability : the infrastructure adapts to exactly what you need
- Critics : you lose the control of your own data (Stallmann) ; it's a sexy word to sell something that exists since a long time (Oracle)

# Cloud computing

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

# Green grid computing

- Turning off unused nodes
- Best-effort / volunteer computing at low frequency
- ...



# Lightweight grid computing

- Using the free cpu cycles for parametric computation when the local users don't need them.
- Zero priority jobs are immediately killed when a local user needs the resources (best-effort type of jobs).

# CIGRI Motivations and Related Work

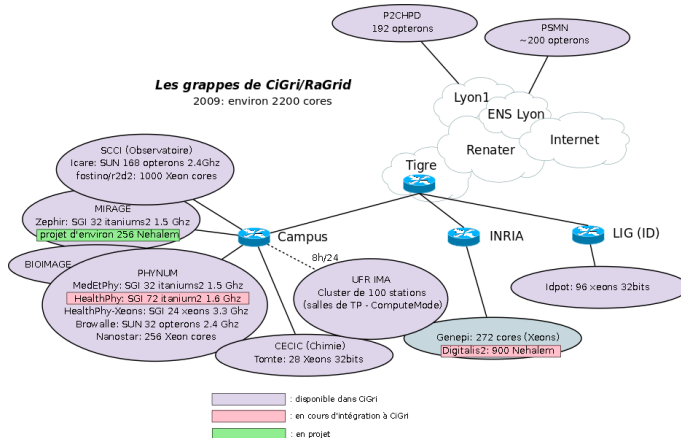
CIMENT project : **Mutualise the computing power** of private laboratories **cluster resources** from different disciplines (environment, chemistry, physics, astronomy, biology, ...) to effectuate **larger scale computations**.

- Option of **Globus** : complicated, expensive (ex. Condor-G, Nimrod-G,...)
- Emergence of **desktop grid** systems and the idea of **cycle stealing** technologies provided the good bases for the CIGRI approach
- Alternative grid solutions : **Condor** (..low security,.. parallel applications), **OurGrid** (..high security, ..BoT applications)

# CIGRI approach for grid computing

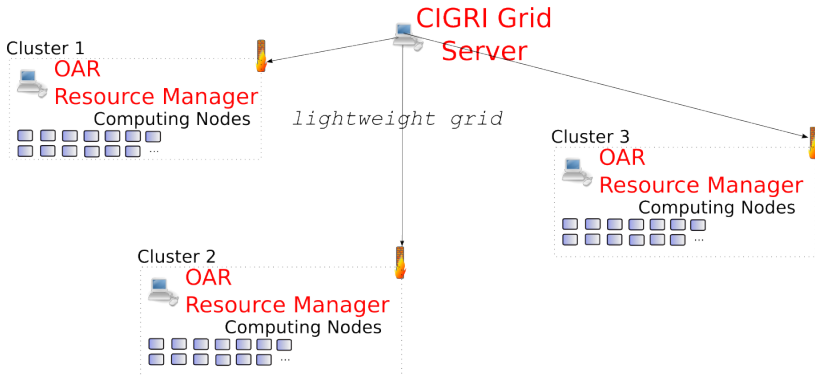
- **CIGRI** : simpler, **lightweight approach** of grid platform (very low security,..only BoT applications)
- Using the method of aggregation of idle cluster resources
- Platform **focuses** on research and development of problems that come along with the execution of tasks (scheduling, fault tolerance,...)
- **Doesn't deal** with important classic grid issues like security, authentication mechanisms , resource location...

# The ciment "platforms"



# CIGRI : Lightweight grid

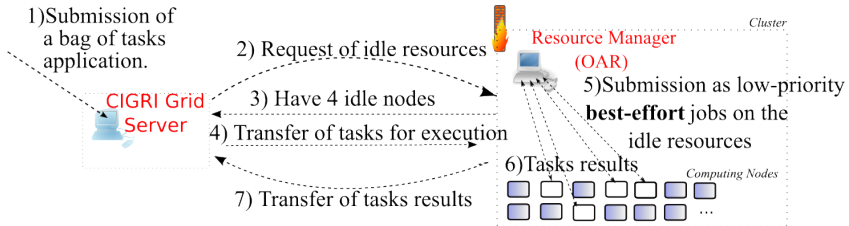
- Homogenisation of services and administration procedures between clusters



# CIGRI : Cluster utilisation policy

- Works **discreetly** along with the interconnected clusters : **No specific** CIGRI software installed on clusters
- **Besteffort jobs** : job type provided by the cluster resource management system (introduced in OAR)
  - Lowest priority jobs submitted only if there is a free resource
  - Killed when local cluster job requests the resource

# CIGRI : Cluster utilisation policy



# The CiGri middleware



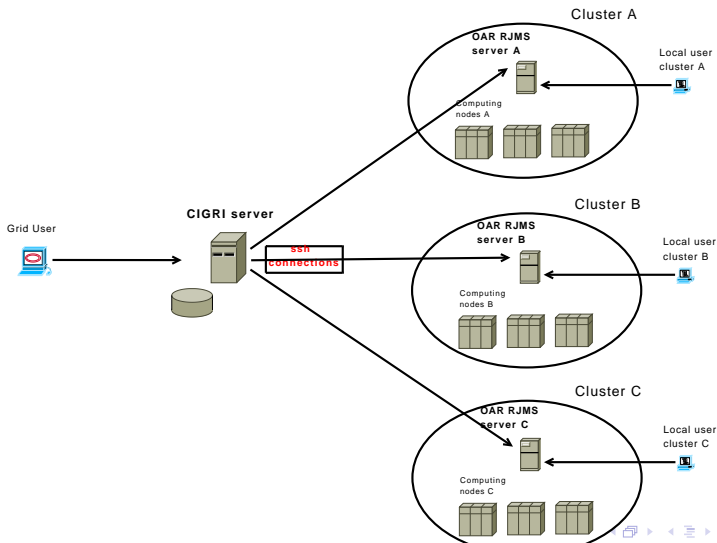
- A central CIGRI host
- Uses an SQL database as the core model
- Communicates with clusters via ssh
- Non intrusive for local production sites
- Submits jobs into the OAR batch scheduler (maybe coupled with another bs)
- Uses the "best effort" concept of the OAR batch scheduler (next slide)



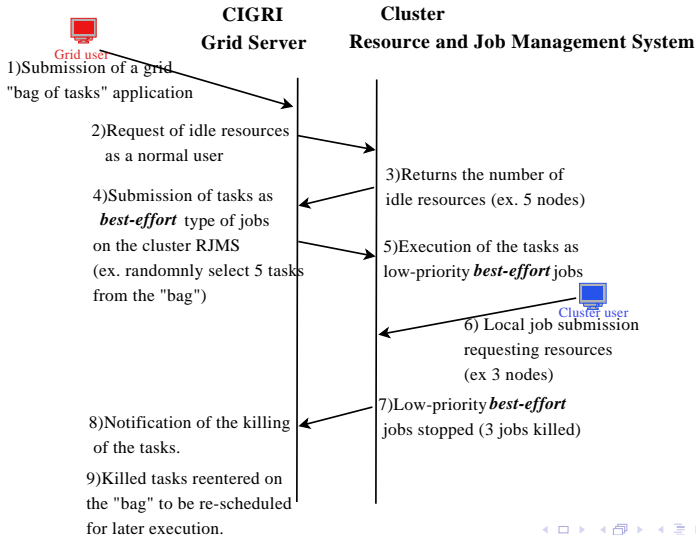
# CIGRI Global Architecture

- **High level components** : MySQL Database, Perl programming language
- No specific software installed on clusters : Based on linux **system commands** (bash, ssh, scp, tar, rsync, ...)
- Integrated to function with OAR resource management system (can be easily integrated with other resource management systems)
- **Modular Architecture** : Easy for development and research

# CIGRI Global Architecture

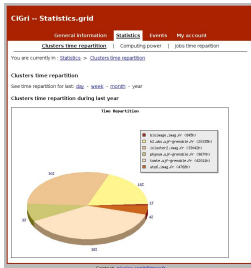


# CIGRI Job execution



# CIGRI Functionalities

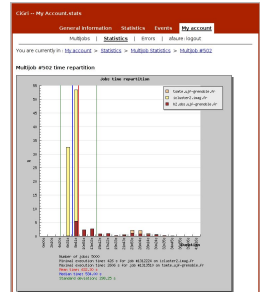
- **Web Portal** for grid monitoring with statistics
- **Collection of results** of the executed jobs from the clusters on a centralised server
- Automatic **Data synchronization** of clusters



Details of the Multijobs 171

Parameters in execution

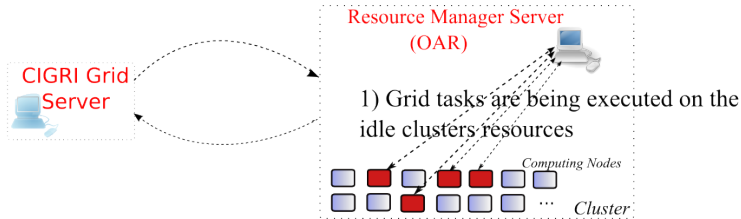
Primary key	Cluster Name	Job Name	Job ID	Job Status	Job Type	Job Size	Job Time	Job Date	Job User	Job Group	Job Priority	Job Queue	Job Count	Job Error	Job Remark
171	cluster1	job1	171	Running	Normal	1000	1000	2010-18-08	user1	group1	1	1	1	0	
172	cluster2	job2	172	Running	Normal	1000	1000	2010-18-08	user2	group2	1	1	1	0	
173	cluster3	job3	173	Running	Normal	1000	1000	2010-18-08	user3	group3	1	1	1	0	
174	cluster4	job4	174	Running	Normal	1000	1000	2010-18-08	user4	group4	1	1	1	0	



# CIGRI Fault Treatment

- **Robustness** : Fault treatment mechanisms highly important for CIGRI grid approach
- CIGRI Fault Treatment Implementation
  - **Locate, log** and **categorize** the different errors that are possible to occur
  - Error handling with rules : either automatic treatment or "request for help mechanism"
- Error classification :
  - **Abnormal behaviour** on the grid like : network communication errors, system command errors,...
  - **Interference failures** : killing of a grid job due to request of the resource by a local job

# Fault treatment (1/2)



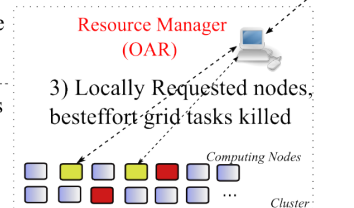
## Fault treatment (2/2)

4) CIGRI Grid Server is notified for the killing of the tasks

5) Tasks reentered on the bag of tasks to be scheduled for re-execution

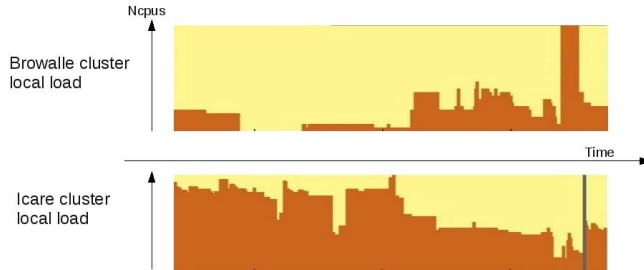
6) Random selection of the tasks by the scheduler when exist free resources on any of the clusters

2) Local user Job submission  
(ex. 2nodes)



- CIGRI guarantees the complete execution of the application

# Cigri efficiency (1/2)



The load of the clusters is not constant and peaks are often not at the same time...



## Cigri efficiency (2/2)



CiGri uses the idle cpus

# Why a grid is not a cluster of clusters?

- A cluster scheduler that has the vision of all the resources (cpus) is not a grid but an heterogeneous multi-cluster
- A special scheduler having an aggregated view of clusters resources and a system to send jobs to underlying (local) batch schedulers is a grid middleware and not a simple batch scheduler so, not a cluster; a grid submission results in a job submission, not in an execution on a node.

◀ Back.