Corso di formazione per neoassunti nelle attività di Computing

4–7 Mar 2024
LNF

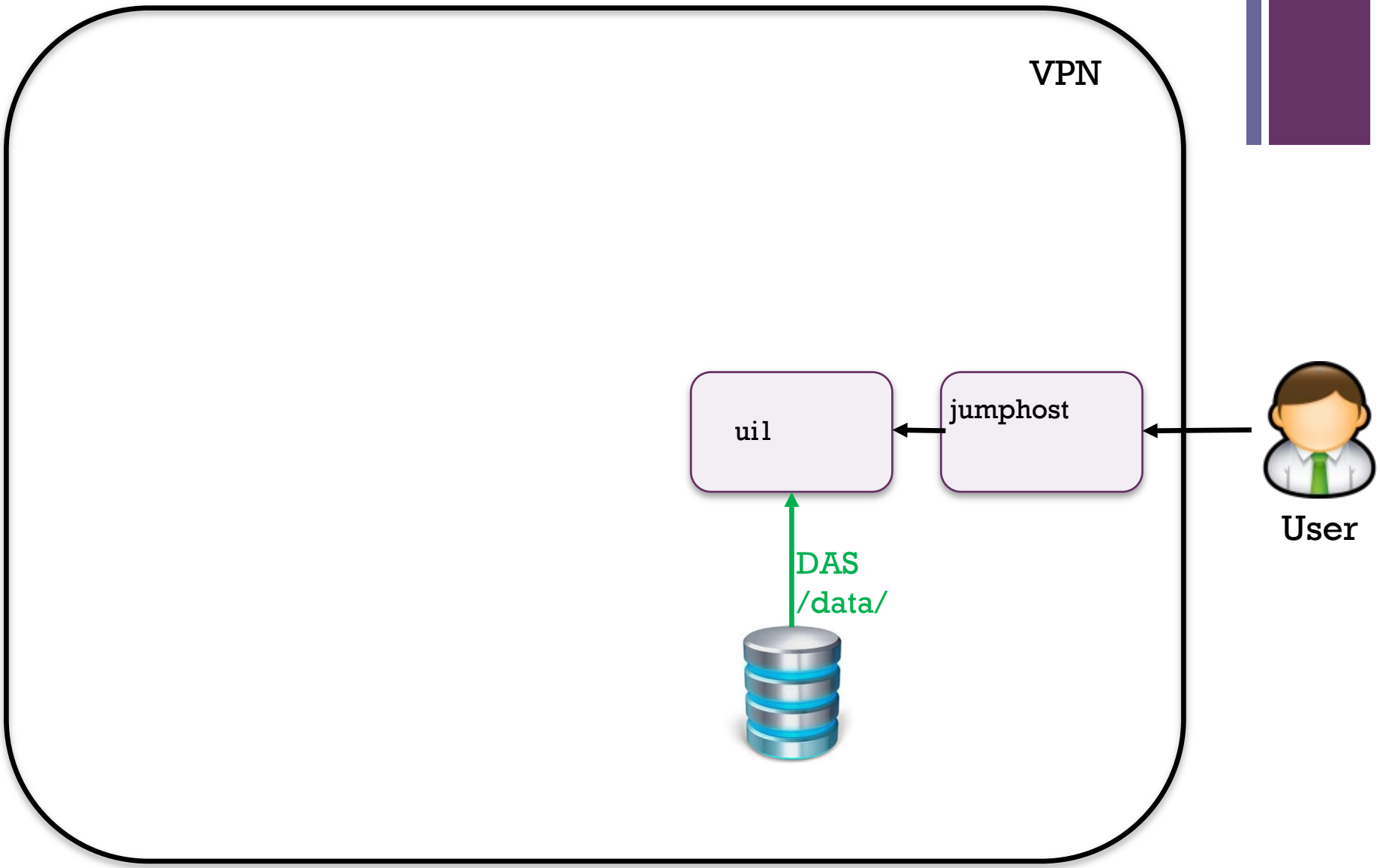
Dal laptop al supercalcolo

Alessandro Costantini, Daniele Cesini – INFN-CNAF
alessandro.costantini <at> cnaf.infn.it

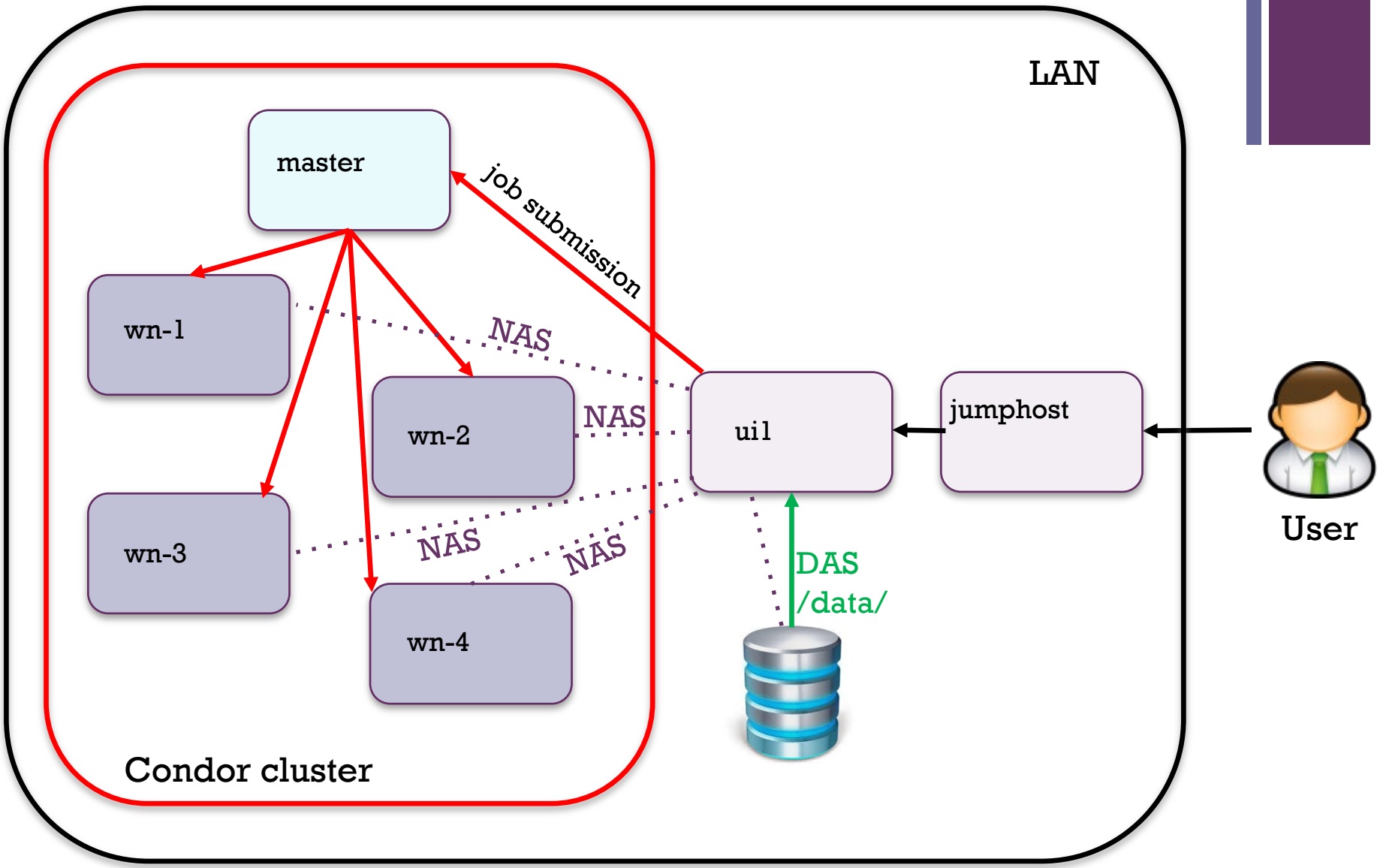
This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license

Computing and Job scheduling

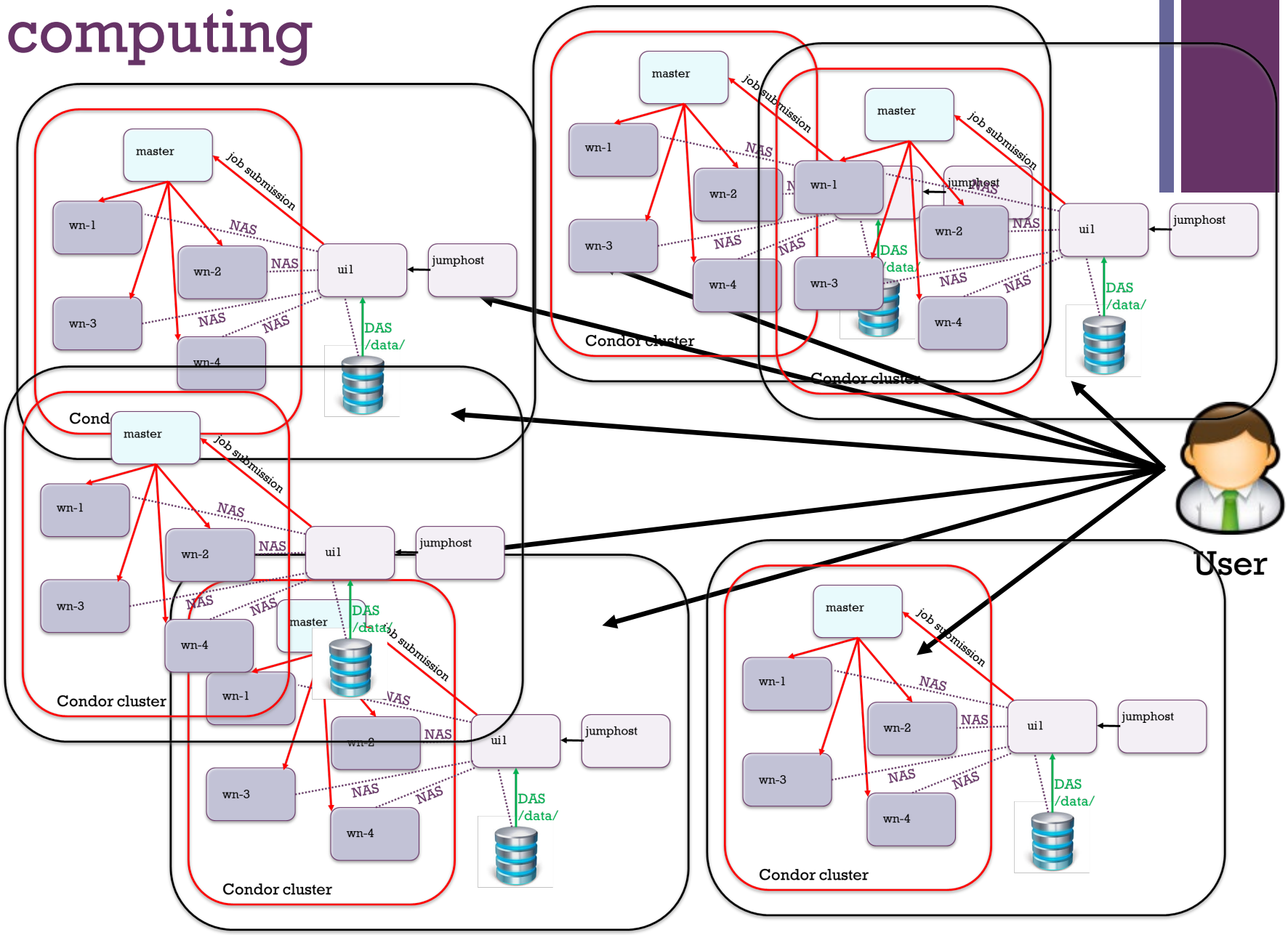
Computing on a server



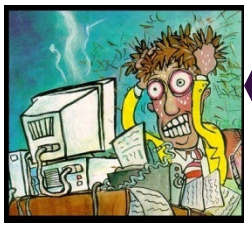
Computing on a datacenter

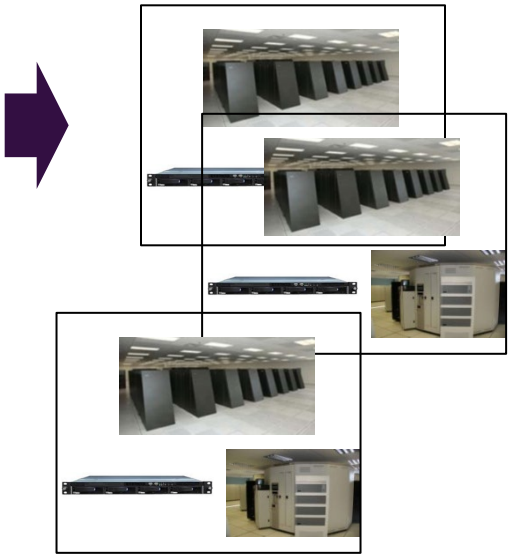


Computing on multi-site: distributed computing



Job Scheduling in distributed environments





Managing the Concurrent Access

- Typically in a datacenter multiple users share the same resources
  - Optimize resource usage and avoid waste of computing power
  - Different users could have paid different shares
  - Different users could have different priorities

Managing the Concurrent Access

- Typically in a datacenter multiple users share the same resources
  - Optimize resource usage and avoid waste of computing power
  - Different users could have paid different shares
  - Different users could have different priorities

Managing the Concurrent Access

- Typically in a datacenter multiple users share the same resources
  - Optimize resource usage and avoid waste of computing power
  - Different users could have paid different shares
  - Different users could have different priorities

- The batch system (or batch scheduler) allows to manage the concurrent access to the resources, respecting priorities and shares

Batch system

- Is a computer application for controlling unattended background program execution of jobs(execution tasks)
  - PBS/MAUI
  - LSF
  - MOAB
  - LoadLeveler
  - SLURM
  - UNIVA Grid Engine
  - OpenLava
  - HTCondor
- Execution of non-interactive jobs is often called batch processing
- Provides a single point of control for jobs submitted to the CPU farm
- Often organize the submission through queues

Batch system queues

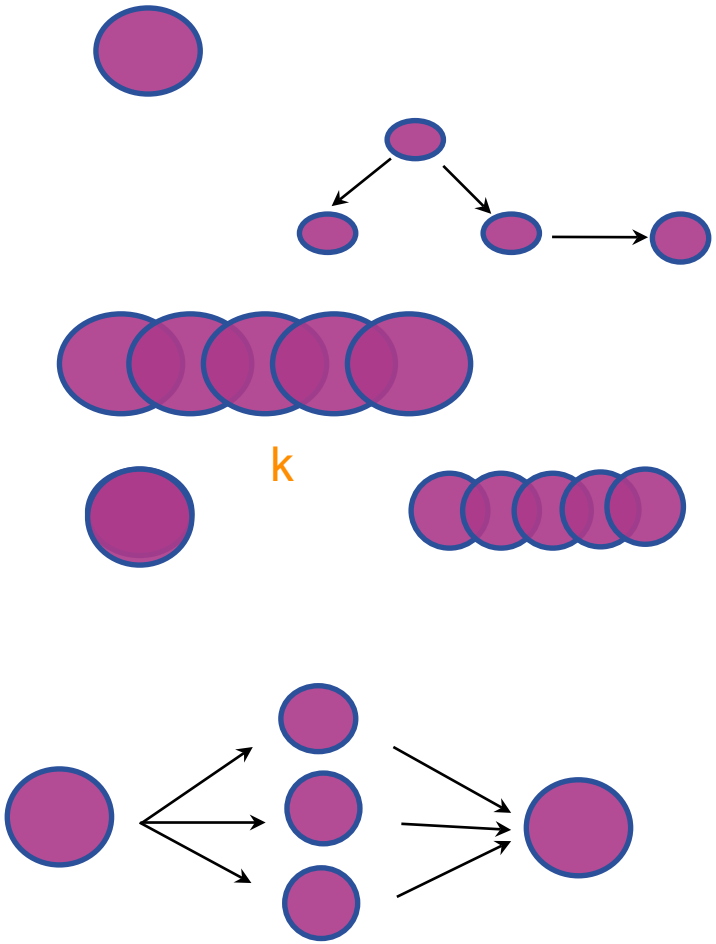

```
[root@ui-tier1 ~]# bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
dteam           200  Open:Active       -    -    -    -     0     0     0     0
ops             200  Open:Active       -    -    -    -    63     0    63     0
cms             100  Open:Active       -    -    -    -    13     0    13     0
ams             100  Open:Active     800    -    -    -   137     0   137     0
atlas_test       60  Open:Active      60    -    -    -    17    17     0     0
mcore            60  Open:Active    7200    -    -   32   656   472   184     0
mcore7           60  Open:Active    7200    -    -   40   904   672   232     0
hnsc_mcore7      60  Open:Active    7200    -    -    8  1224   800   424     0
atlas            50  Open:Active       -    -    -    -  2563     0  2563     0
atlas_bari       50  Open:Active       -    -    -    -   129     0   129     0
atlas7           50  Open:Active       -    -    -   30  1034     0  1034     0
test             40  Open:Active     100    -    -    -     0     0     0     0
generic          40  Closed:Inact    100    -    -    -     0     0     0     0
infngrid         40  Closed:Inact      -    -    -    -     0     0     0     0
icarus           40  Open:Active     100    -    -    -     0     0     0     0
argo             40  Closed:Inact      -    -    -    -     0     0     0     0
auger            40  Open:Active    1500    -    -    -   132   131     1     0
auger_db         40  Open:Active     450    -    -    -  2092  1925   167     0
aug_hm_long      40  Closed:Inact      8    -    -    -     0     0     0     0
ccube            40  Open:Active     300    -    -    -    23     0    23     0
juno             40  Open:Active     400    -    -    -     8     0     8     0
famu             40  Open:Active     100    -    -    -     0     0     0     0
panda            40  Open:Active     300    -    -    -     0     0     0     0
cdf              40  Open:Active       -    -    -    -     0     0     0     0
magic            40  Open:Active       -    -    -    -     0     0     0     0
opera            40  Open:Active     300    -    -    -     0     0     0     0
gerda            40  Open:Active     100    -    -    -     0     0     0     0
rdfa             40  Open:Active     100    -    -    -     0     0     0     0
cms_mcore        40  Open:Active    9000    -    -   32  8649  6472  2177     0
cms_mcn          40  Open:Active    9000    -    -   72  7848  6816  1032     0
virgo            40  Open:Active    8000    -    -   64   205     0   205     0
virgo_h12        40  Open:Active    2500    -    -   16     0     0     0     0
cta              40  Open:Active       -    -    -    -  2791   500  2291     0
```

Scheduling Policies

- Various schemes are used to decide which job to run when a resource (job slot) is available
- Parameters that might be considered include:
  - Job/User priority
  - Compute resource availability
  - License key if job is using licensed software
  - Execution time allocated to user
  - Number of simultaneous jobs allowed for a user
  - Estimated execution time
  - Elapsed execution time
  - Availability of peripheral devices
  - Occurrence of prescribed events
  - Job dependency
  - File dependency
  - Operator prompt dependency

Job types

- Single batch job
- DAG workflow
- Collection
- Parametric
- Parallel

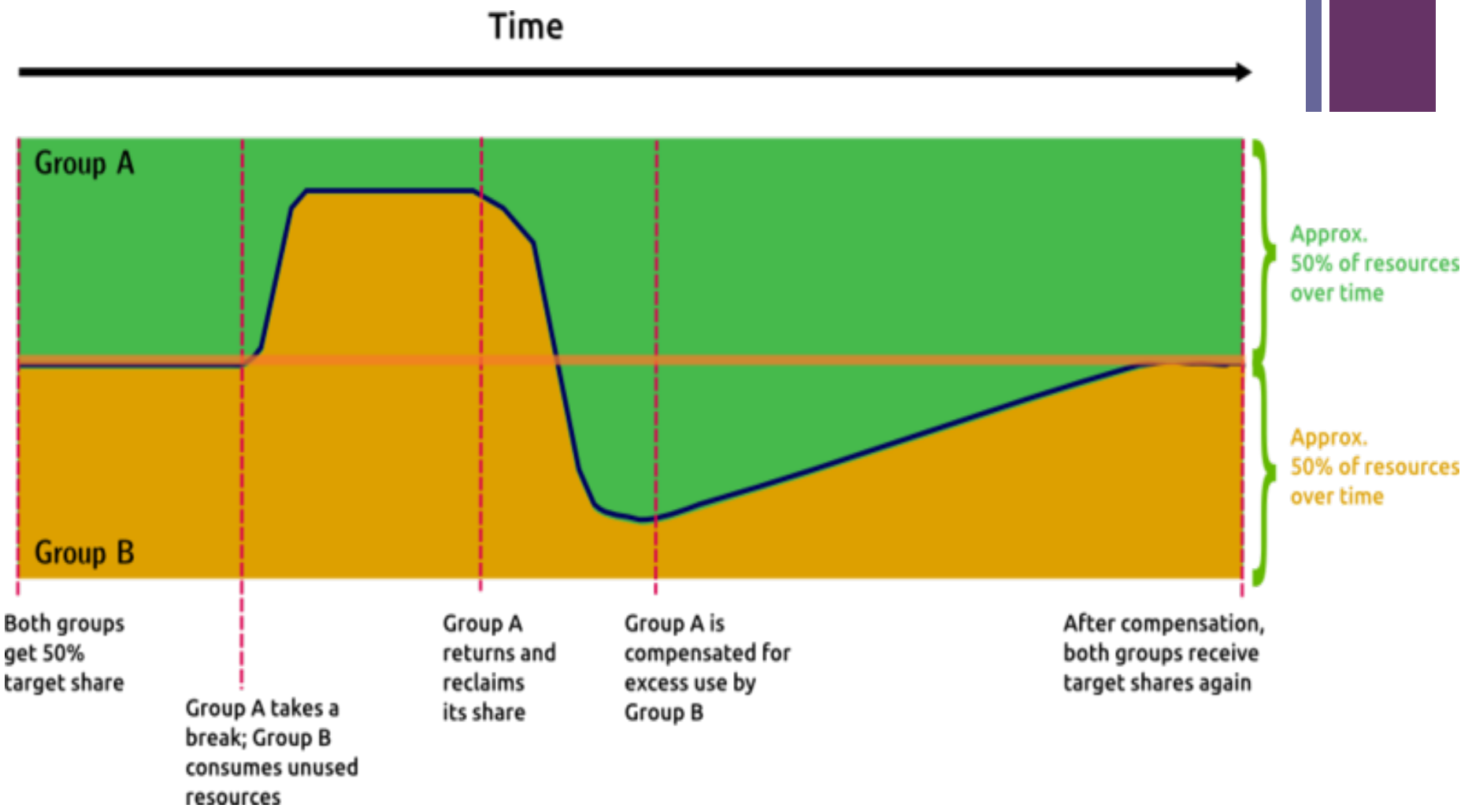


compound

Job types

- Single batch job
  - a task occupying a single slot (normally one core)
- DAG workflow:
  - Directed Acyclic Graph workflow - a series of jobs whose dependencies can be described by a **graph** without cycles (a cycle is a complete circuit). When following the **graph** from node to node, you will never visit the same node twice
    - Used to concatenate automatically jobs that need as input the output of other jobs
- Collection
  - A set of equal jobs with no dependencies that can be submitted at the same time
    - Normally acting on different input data
- Parametric
  - A collection that can be easily defined by a parameter
- Parallel
  - Jobs needing more that a single job slot to run (i.e. MPI or OpenMP jobs)

Fair share scheduling



Fair share scheduling

- Normally, by default, batch systems consider jobs for dispatch in the same order as they appear in the queue.
  - This is called first-come, first-served (FCFS) scheduling.
- Fairshare scheduling divides the processing power of the cluster among users and queues to provide fair access to resources
  - so that no user or queue can monopolize the resources of the cluster and no queue will be starved.
- Fairshare scheduling controls how the resources should be shared by competing users
- Fairshare is not necessarily equal share: you can assign a higher priority to the most important users. If there are two users competing for resources, you can:
  - Give all the resources to the most important user
  - Share the resources so the most important user gets the most resources
  - Share the resources so that all users have equal importance

© IBM LSF:
https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/fairshare_about_lsf.html

Fair share scheduling

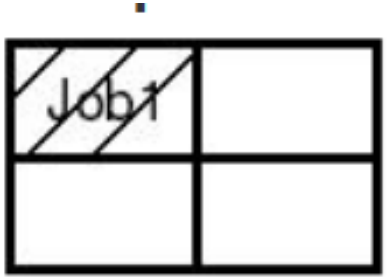
- The most important parameter considered in fairshare is the dynamic priority of the user who submitted the job.
- When fairshare scheduling is used, the batch system tries to place the first job in the queue that belongs to the user with the highest dynamic priority
- Can be set for single user or group of users
- Can be set at the level of queue or at the level of the entire farms

In example…

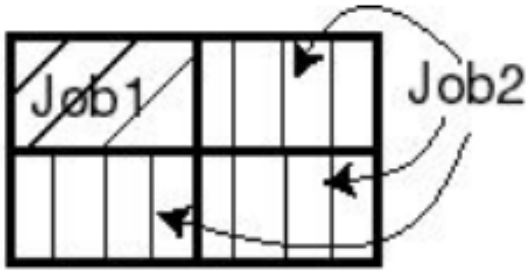
dynamic priority = *number_shares* / (*cpu_time** CPU_TIME_FACTOR + *run_time* * RUN_TIME_FACTOR + (1 + *job_slots*) * RUN_JOB_FACTOR + (1 + *fwd_job_slots*) * FWD_JOB_FACTOR + *fairshare_adjustment**FAIRSHARE_ADJUSTMENT_FACTOR)

© IBM LSF:
https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/fairshare_about_lsf.html

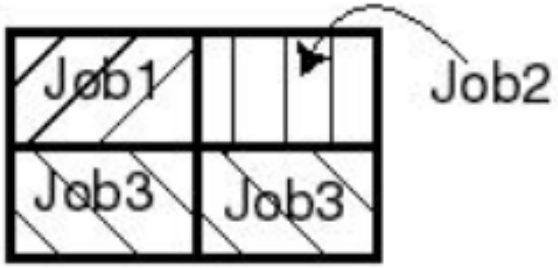
Reservation / Backfill



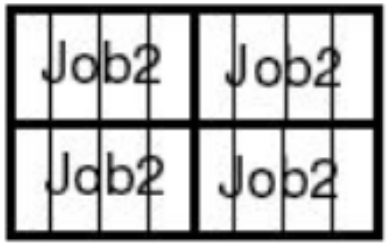
(a) Job1 started at 8:00 am.
Will finish at 10:00 am.



(b) Job2, submitted but can't start since it needs 4 processors.
Remaining 3 reserved by Job2.



(c) At 8:30 am Job3 submitted.
Job3 backfills Job2.



(d) At 10:00 am, Job2 starts.

© IBM LSF https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/backfill.html

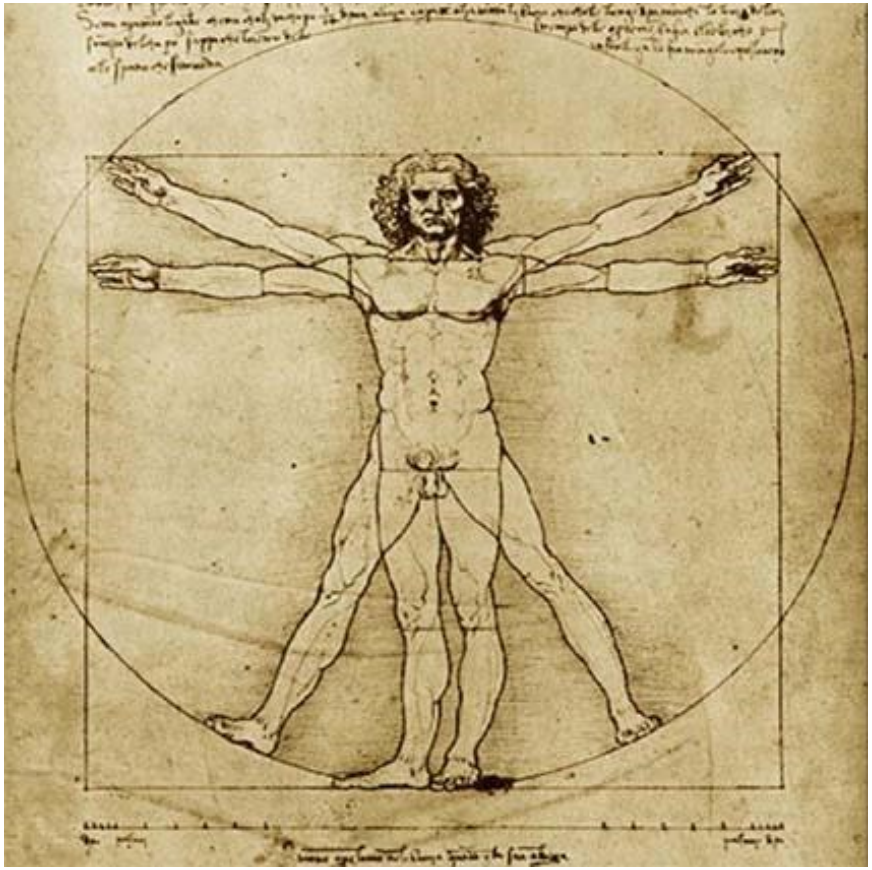
Reservation / Backfill

- In a busy cluster, processor reservation helps to schedule large parallel jobs sooner
- by default, reserved processors remain idle until the large job starts. This degrades the performance of the farm because the reserved resources are idle while jobs are waiting in the queue
- Backfill scheduling allows the reserved job slots to be used by small jobs that can run and finish before the large job starts. This improves the performance of farm because it increases the utilization of resources
  - It is possible only if the user declares the job duration
    - The job is killed after that duration

Take away messages (Computing)

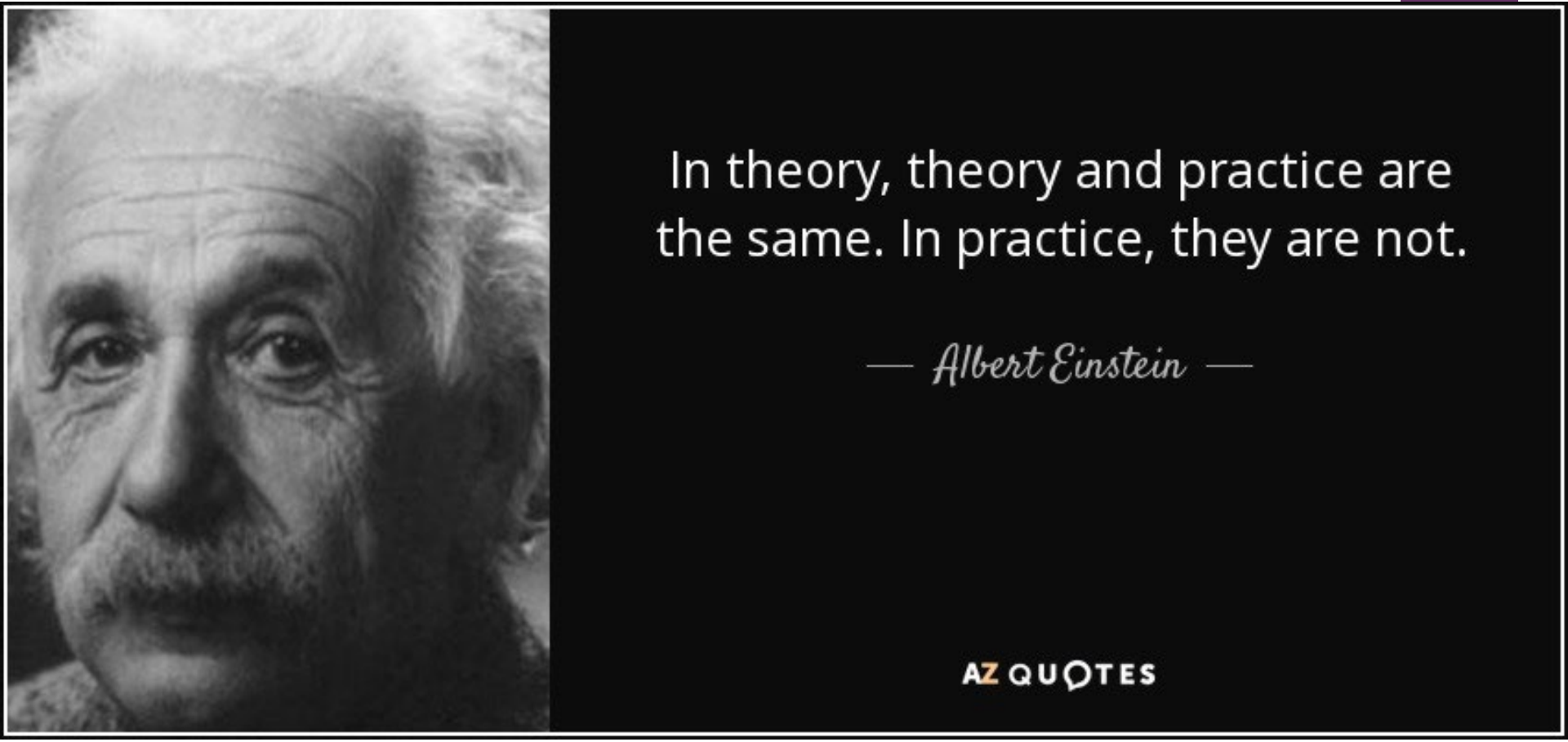
- Concurrency in a datacenter is handled by a Batch System
- Often the user submit unattended background program execution of jobs (execution tasks) -> Batch computing
- Fairshare uses a dynamic priority to ensure that agreed quotas are respected
- The dynamic priority may depend on a huge number of factors
- The batch systems tries to minimize the waiting time in the queues and maximize at the same time the cluster utilization and hence its efficiency

Anatomy of a batch system job



- JOB Type
- Prologue
- Input SandBox
- Requirements
- Executable
- Std Output/Error
- Output SandBox
- Epilogue
- Error Recovery

Let's put it into practice



Workload Management System

Workload Management System

- Workload management is a process for determining the proper workload distributions in order to provide optimal performance for applications and users
  - Resource Matchmaking, Selection and Rank
  - Job distribution
- It provides the capacity to control and manage where each work request is run in order to maximize workload throughput and enhance performance by making sure that no single processing node/site is overtaxed while others are underutilized
- Handles job status, monitoring, information retrieving
- It can handle Input/Output Sandboxes

Scheduling strategies

Scheduling jobs in distributed infrastructures is a challenging problem

- Eager scheduling ("push" model)
  - The job is bound to a resource as soon as possible
  - Once the decision has been taken on which resource will be used, the job is passed to the selected resource for execution

- Lazy scheduling ("pull" model)
  - The job is held by the Workload Management System until a resource becomes available
  - When this happens the resource is matched against the submitted job
  - The user submit to a queue and jobs are pulled from that queue

Which is the best?

Workload Management System

- Can be developed in house for simple workload
- Better to use community developed services in all other cases

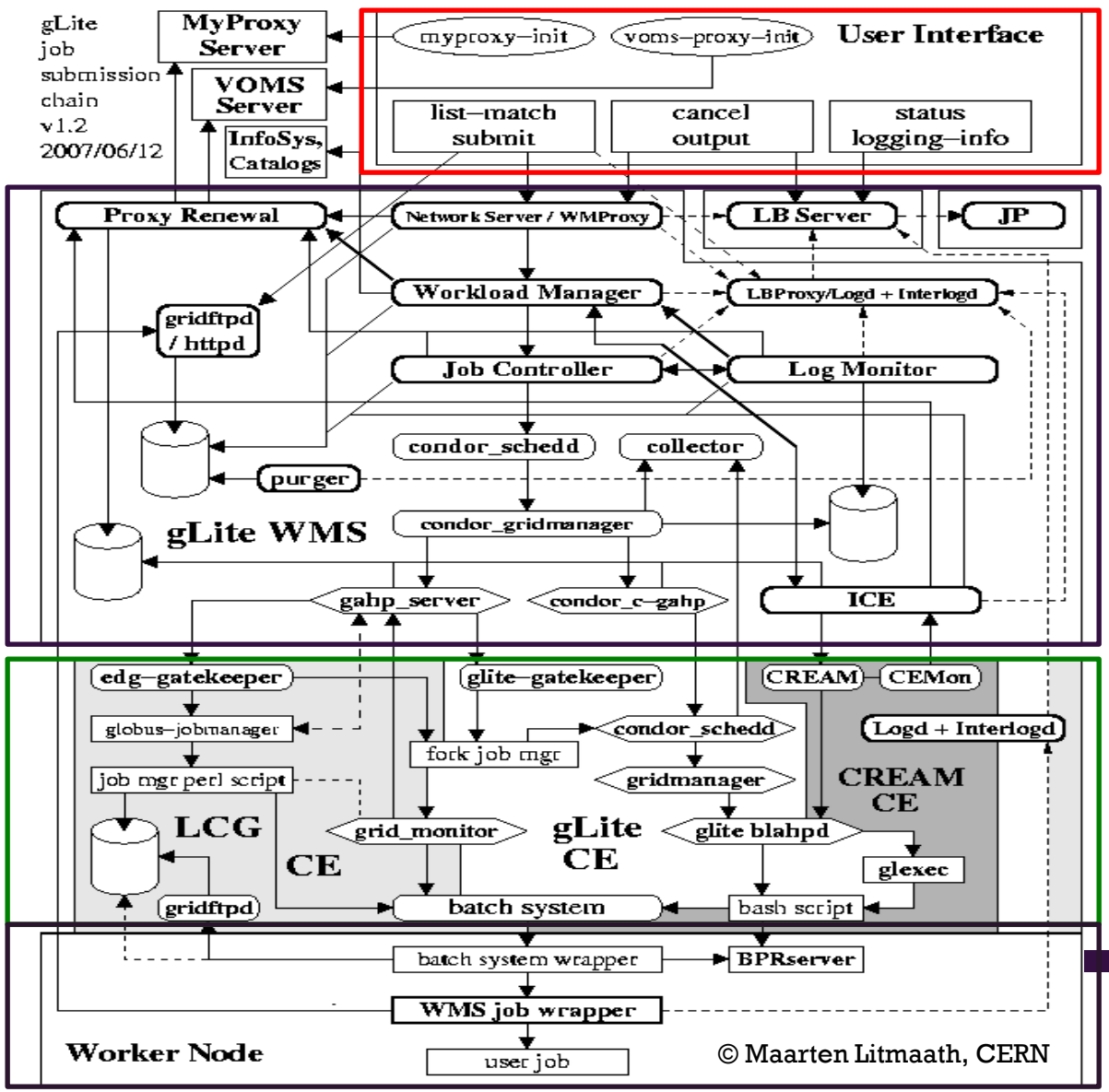
Overall architecture example

UI

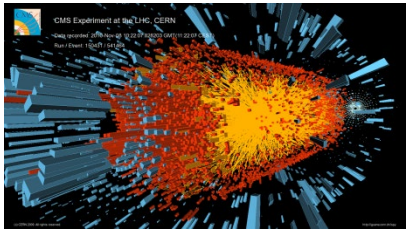
WMS

CE

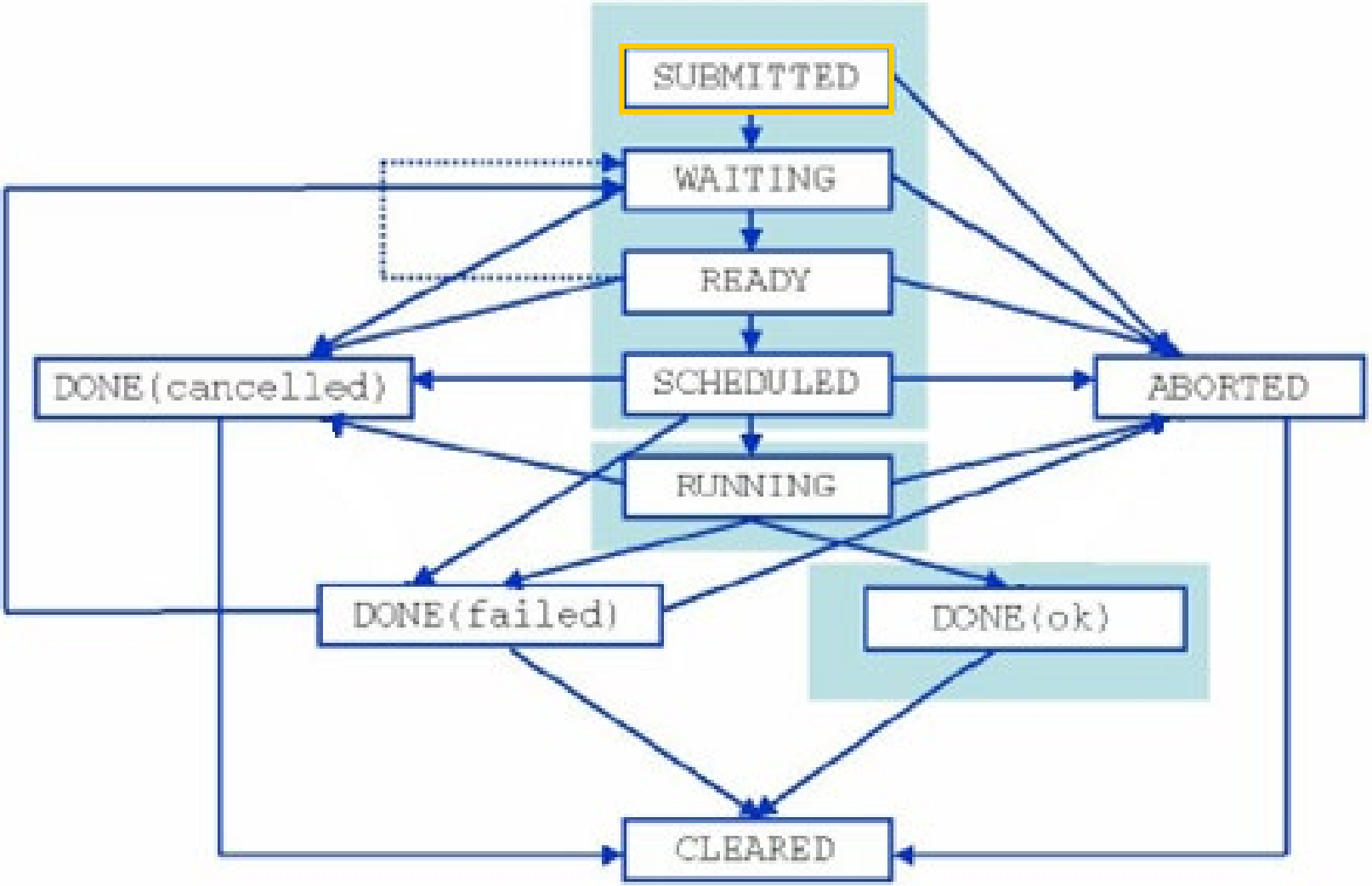
WN



SE



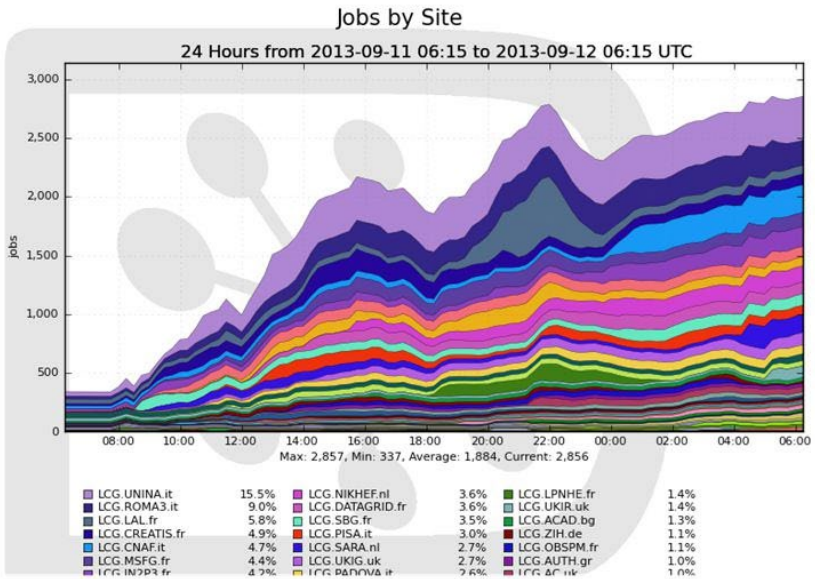
Jobs State Machine



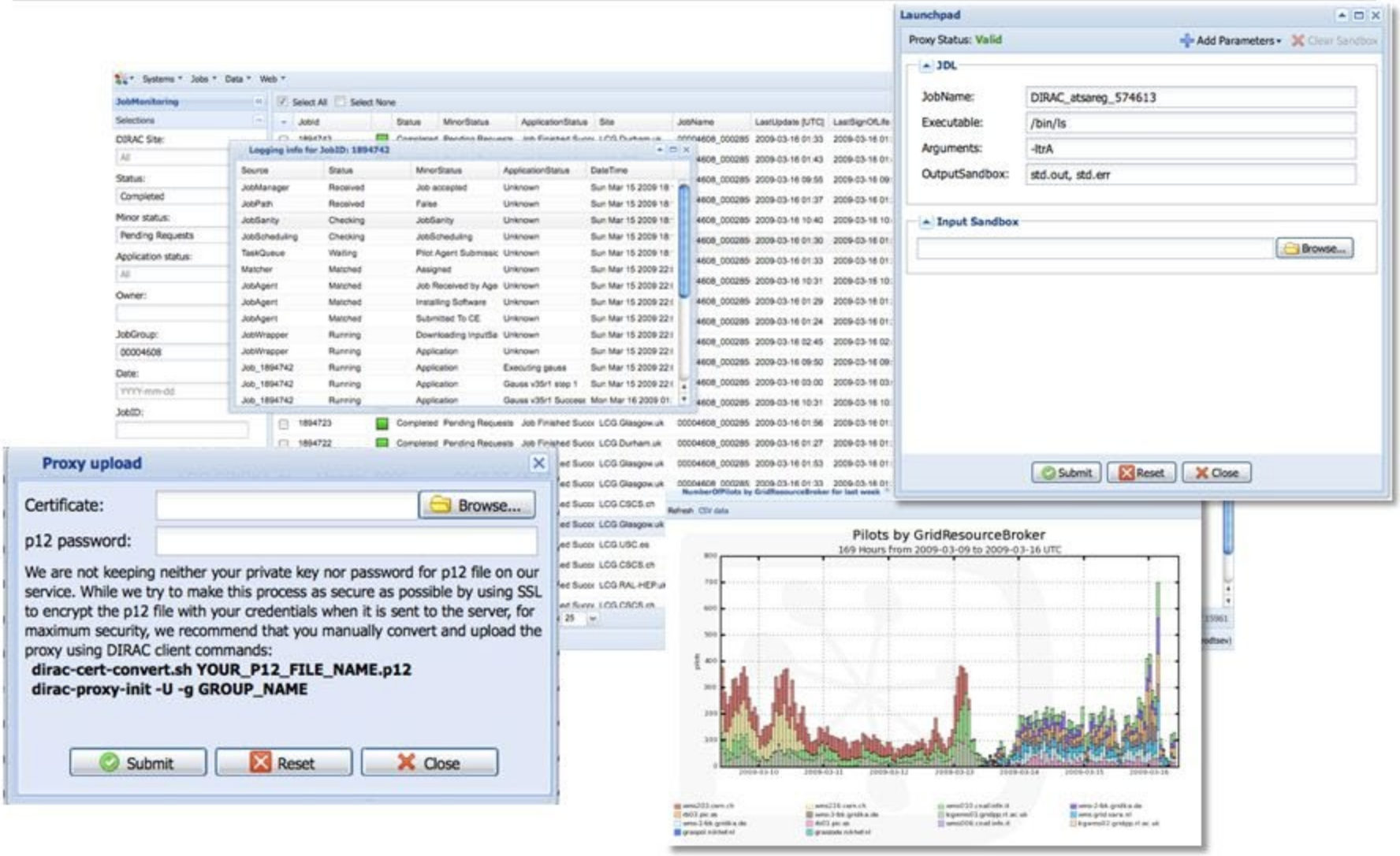
Monitoring capabilities

DIRAC THE INTERWARE

Today FG-DIRAC snapshot



Web Portal: example interfaces

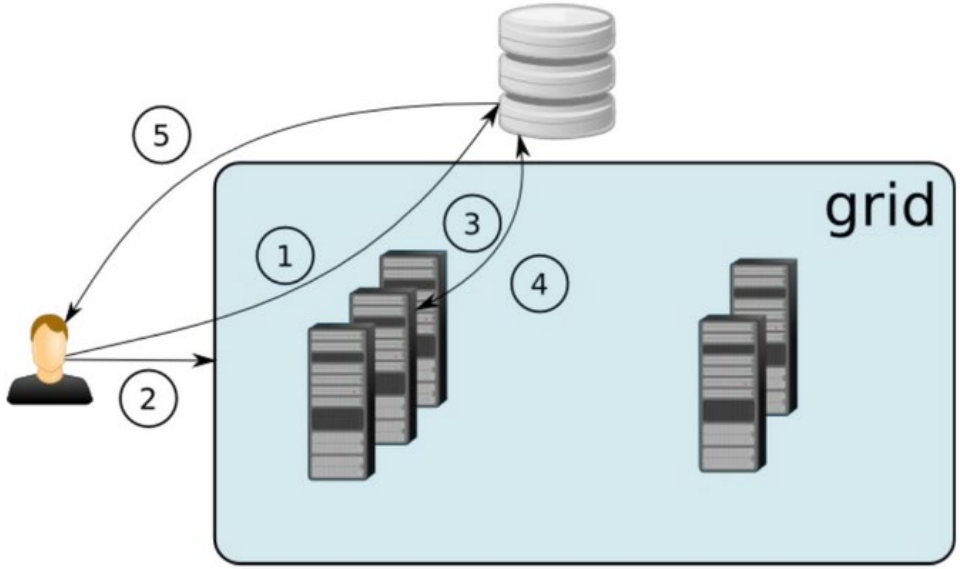


Job Submission Modes

- Submission through the Workload Management System
  - The responsibility of the outcome of the job is passed to a service – the WMS
    - which provides value-added capabilities and instrumented to always know the whole picture trough the Information System of the infrastructure
- Direct submission to computing resources
  - A Workload Management System is bypassed
  - Users cannot have a global view of the whole picture
  - The responsibility of the job remains to the user
- A simple WMS can be built on top of Direct submission

Pilot Job Submission Mode

- (1), the user uploads work to the central database, he then (2) submits jobs just containing the application to the Grid, which handles retrieving from (3) and updating of (4) the job database.
- This process continues until all the work present in the database has been done, the application has crashed or the job has run out of time on the computational resource.
- When all work has been done, the user retrieves (5) the results from the database.



Pilot Job Submission Mode

- It is a way to implement a pull submission system
- Special jobs – the Pilots – are submitted a priori to all the available resources
  - Execute no real tasks
  - Check the environment, hw and sw
  - Pull jobs from a central task queue
  - Terminate if no tasks are available in the queue

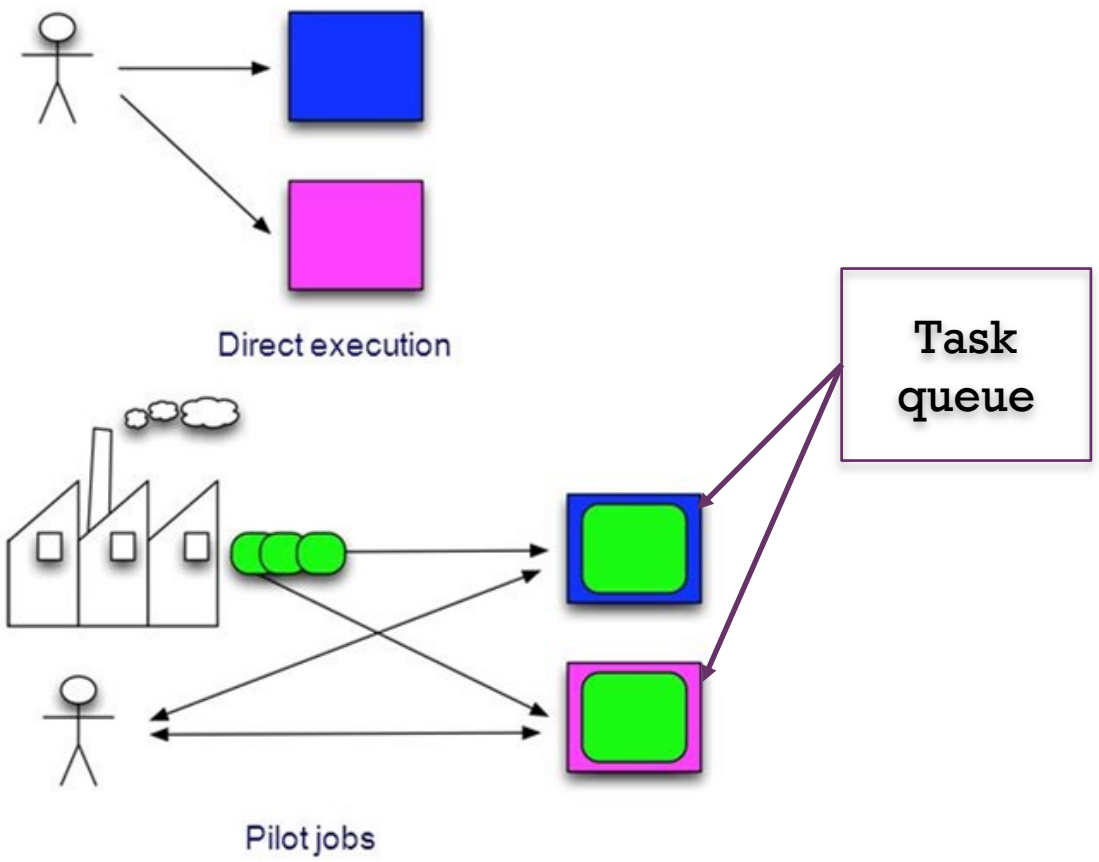
Pilot Job Submission Mode

- Advantages
  - Jobs land on resources that have been already checked and tested by the pilots
  - Allow to bypass the priority set by the site admins (at a certain extent)
  - Make it easier to optimize the usage of the site resources

- Disadvantages
  - Easily destroys any system of a-priori data distribution and data pre-placement
  - The central queue is a single point of failure if not correctly implemented

Direct execution vs Pilot jobs

- Pilots
  - Separate user job from grid job
  - Have some overhead
  - Can perform some initial test
  - Uniform (enhanced) environment
  - Delayed scheduling



SandBoxes

- InputSandBox and OutputSandBox indicate the set of files that "travel" with the Job when submitted

! USE SMALL SANDBOXES !

(there is a server handling your sandboxes we should avoid to overload that server)

- Use the data management tools offered by the Infrastructure to transfer big data files

Handling ISB & OSB

OUTPUT SANDBOX (OSB)

JOB

INPUT SANDBOX (ISB)

ISB & OSB

- Can travel with the job through the WMS
  - MUST BE SMALL (~5MB) !!!
- Can be remote Any size
  USE DATA MANAGEMENT COMMANDS

Job Prologue and Epilogue

- Prologue
  - Prepare the environment to execute jobs
    - Install libraries
    - Transfers data and executables
    - Check that sw and hw available at the sites are inline with what is needed to the job
- Epilogue
  - Run post executions scripts
    - Transfer output data
    - Update databases
    - Compute checksums
    - Perform final checks

Sometimes the job cannot be modified by the user and only Prologue and Epilogue can be customized

Distribution strategies and Computing models

Data distribution strategies

- a-priori push: we know the resources that will be used and we distribute the data before launching the jobs
  - Distribution strategies in general depend on the types of application
- Job pull: the first operation that a job (or its prologue) performs is fetching the input data
- No distribution is possible: someone else distributed the data, we cannot move them
  - Privacy, Security, Size, Policy, Funds
- Output data
  - Keep where produced
  - Move somewhere else
    - Final step of a job
    - Done by prologue

Data distribution strategies

- When distributing data (both input and output) consider:
  - Backup strategies
    - Use different Quality-of-Services
      - Disk in one site, tape in another
  - Failover strategies
    - Use more than one site if possible
  - Access policies
    - i.e. restrict write access to important data (RAW)
  - Load distribution

REMEMBER that everything that could go wrong will

Policy driven Data Management

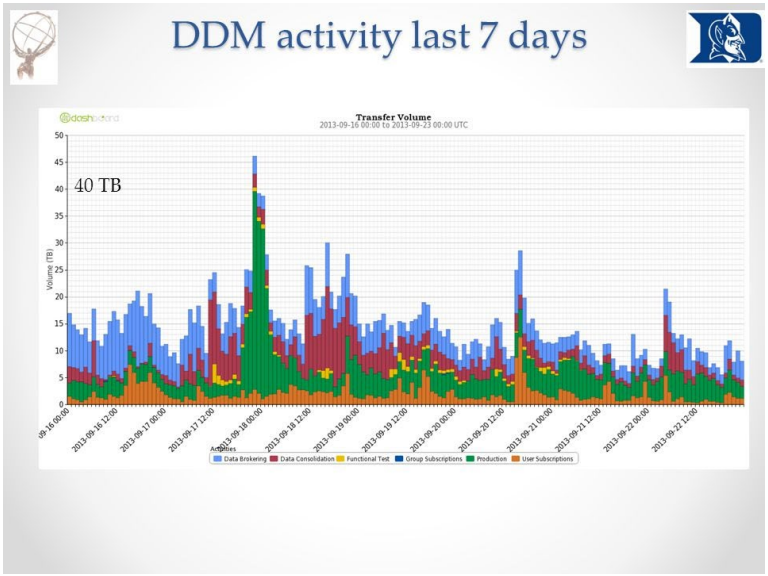
- Dataset Distribution
  - A typical workflow
    - Initially the data will be stored on low latency devices for fast access
    - To ensure data safety, the data will be replicated to a second storage device and will be migrated to custodial systems, which might be tape or S3 appliances
    - Eligible users will get permission to restore archived data if necessary
    - After a grace period, Access Control will be changed from "private" to "open access"

User Software and Applications

- In general, should be treated as any other Input data
  - Needs distribution strategies!!
    - Could depend on licensing
    - Sometimes cannot be distributed
    - Etc…
  - Could drive the job distribution model
    - i.e applications could be dockerized
      - Need docker support

Data Distribution and Data Management Systems

- As for the job distribution, also Data Management can be delegated to third party services
  - Handle replicas
  - Maintain policies over time
  - Executes data transfers as batch jobs
  - Set Access Control Lists



Data-driven computing model

- The data location and availability "drives" the selection of resources and sites that will run my jobs
- Not always data can be moved
  - because of their size
  - because of the policies of the data owners
  - because there is not enough bandwidth
- The matchmaking process needs to take into account the data availability at sites

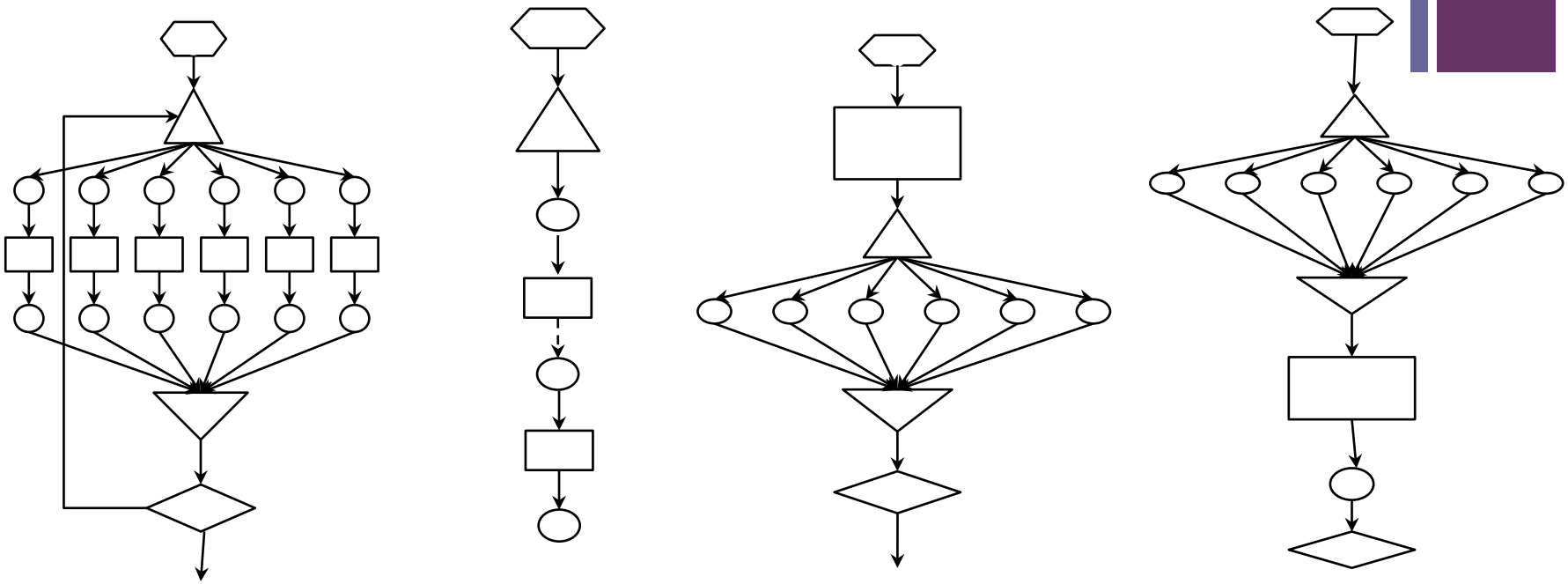
CPU-driven computing models

- Jobs are sent where the computing power is available and data are transferred accordingly
  - A-priori distribution
  - By the job itself
    - (or by the Prologue/Epilogue)
- Resources must be selected according to the application needs
  - Scalar vs parallel
  - RAM needed
  - Queues duration
  - User Priority

Data and CPU driven models

- Complex workflows can be a linear combination of CPU and Data driven models
- In particular when different computations of the workflow need very different hw architectures
  - i.e. HTC and HPC workflows

HTC/HPC Workflows



Arrows indicates logical dependencies, but also data transfers, that can be local or remote on the case of multi-site infrastructures

HPC run (multi/manycores)

Scalar pre&post Processing or can be data sources

HTC and HPC - definition

- High Throughput Computing (HTC)
  - The focus is on the execution of many copies of the *same program* at the *same time*
    - not in the speedup of individual jobs
  - Many copies of the same program run *in parallel* or *concurrently*
  - Maximize the **throughput**
- High Performance Computing (HPC)
  - speed up the individual job as much possible so that results are achieved more quickly
- HTC infrastructures tend to deliver large amounts of computational power over a long period of time.
  - In contrast, High Performance Computing (HPC) environments deliver a tremendous amount of compute power over a short period of time.
- The interest in HTC is in how many jobs complete over a long period of time instead of how fast an individual job can complete.

GFLOPS definition

- FLOPS: Floating Point Operations per second
  - FLOPS on an HPC-system

$$\text{FLOPS} = \text{racks} \times \frac{\text{nodes}}{\text{rack}} \times \frac{\text{sockets}}{\text{node}} \times \frac{\text{cores}}{\text{socket}} \times \frac{\text{cycles}}{\text{second}} \times \frac{\text{FLOPs}}{\text{cycle}}.$$

  - FLOPS on a 1 CPU system

$$\text{FLOPS} = \text{cores} \times \frac{\text{cycles}}{\text{second}} \times \frac{\text{FLOPs}}{\text{cycle}}.$$

Computer performance

Name	Unit	Value
kiloFLOPS	kFLOPS	10^3
megaFLOPS	MFLOPS	10^6
gigaFLOPS	GFLOPS	10^9
teraFLOPS	TFLOPS	10^{12}
petaFLOPS	PFLOPS	10^{15}
exaFLOPS	EFLOPS	10^{18}
zettaFLOPS	ZFLOPS	10^{21}
yottaFLOPS	YFLOPS	10^{24}
ronnaFLOPS	RFLOPS	10^{27}
quettaFLOPS	QFLOPS	10^{30}

Crunching Factor or Speedup

- Ratio between the time needed to completed the challenge on a single machine and the time needed on the distributed infrastructure
  - It depends on the number of available resources
  - On the priority of the user/vo
  - On the congestion of the infrastructure
    - Queue waiting times
  - Highly influenced by data transfers times
  - Highly influenced by errors/resubmission
  - Very unstable over time

HTC - infrastructures

- PC clusters
- Server clusters
- Distributed systems
- Grids

Grids and distributed systems

- What is a Grid?
- Grid types
- Anatomy of a Grid
- Accessing a Grid

© Grant Faint

What is a Grid? - Early definition

Ian Foster

I.Foster, C.Kesselman: The Grid: Blueprint for a New Computing Infrastructure", 1998

Carl Kesselman

"A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities"

What is a computational Grid? the 3 points checklist

A Grid is a system that…..

1) **Coordinates resources that are not subject to centralized control**

2) **Uses standard, open, general-purpose protocols and interfaces**

3) **Delivers nontrivial qualities of service** (Ian Foster, 2002)

[1] Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 259-278
[2] Ian Foster, Carl Kesselman, and Steven Tuecke. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. High Perform. Comput. Appl. 15, 3 (August 2001), 200-222. DOI=10.1177/109434200101500302
[3] What is the Grid? A Three Point Checklist. I. Foster, GRIDToday, July 20, 2002.

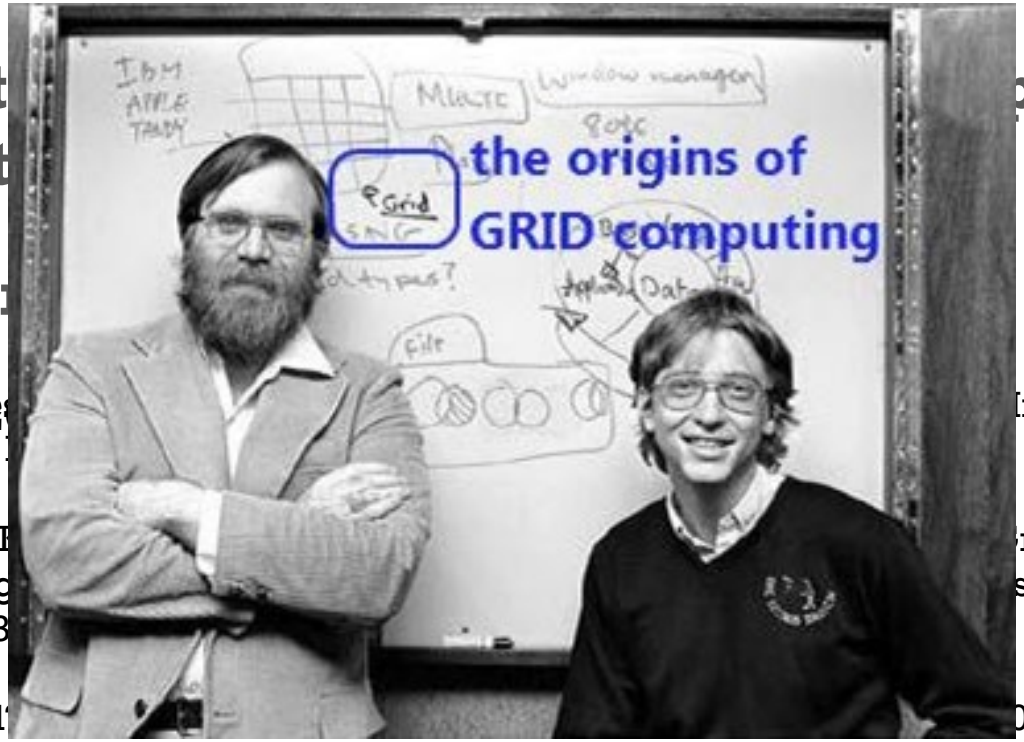
What is a computational Grid? the 3 point checklist

A Grid is a system that.....

1) **Coordinates resources that are not subject to centralized control**

2) **Uses st** **protocols and int**

3) **Deliver**



(Ian Foster, 2

[1] Foster, I. and Ke Infrastructure, Morgan Kaufmann,

[2] Ian Foster, Carl K rid: Enabling Scalable Virtual Org st 2001), 200-222. DOI=10.1177/10943

[3] What is the Grid? 02.

Paul Allen and Bill Gates in 1982.

Grid Users

- **Virtual Organizations: a set of individuals and/or institutions that share resources under certain rules**
  - *Sharing is highly controlled,* resource providers and consumers define clearly and carefully what is shared, who is allowed to share, and the conditions under which sharing occurs

1st Law of the Grid

95% of the Grid is… agreement

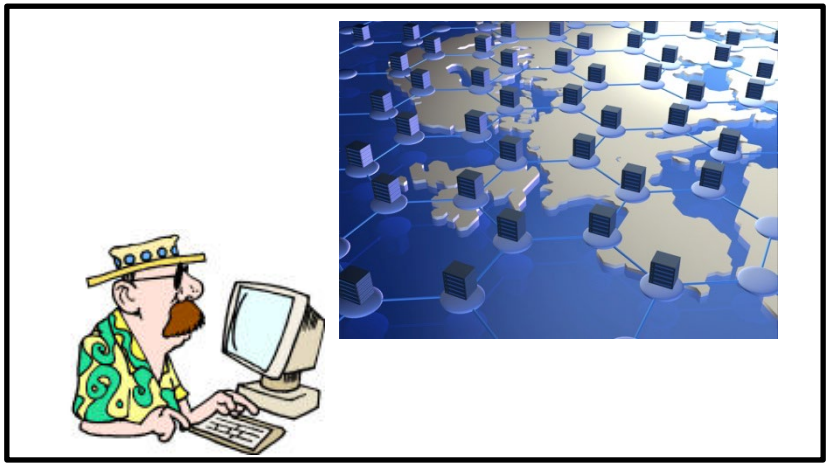
- Key terms
  - Coordination
  - No centralized control
  - Standards
  - Protocols
  - Interfaces

- Standards, protocols, interfaces,… aim at providing common abstractions of different implementations of similar services

Grid: No centralized control

The user in general has full ownership of a desktop workstation.

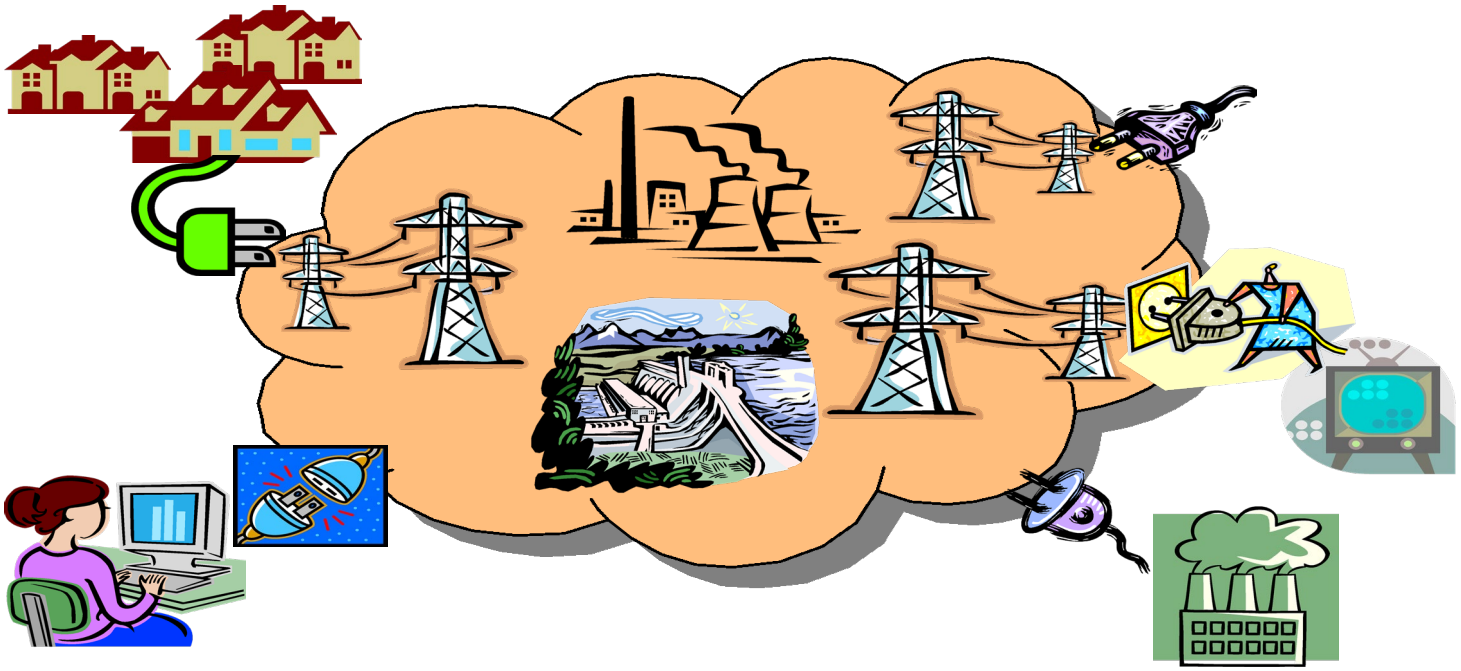
A Cluster is a shared resource – Only the administrator has full control of the system
The physical layer is still well defined



I submit my jobs to "the GRID" and they get processed: somehow, somewhere, after some time.

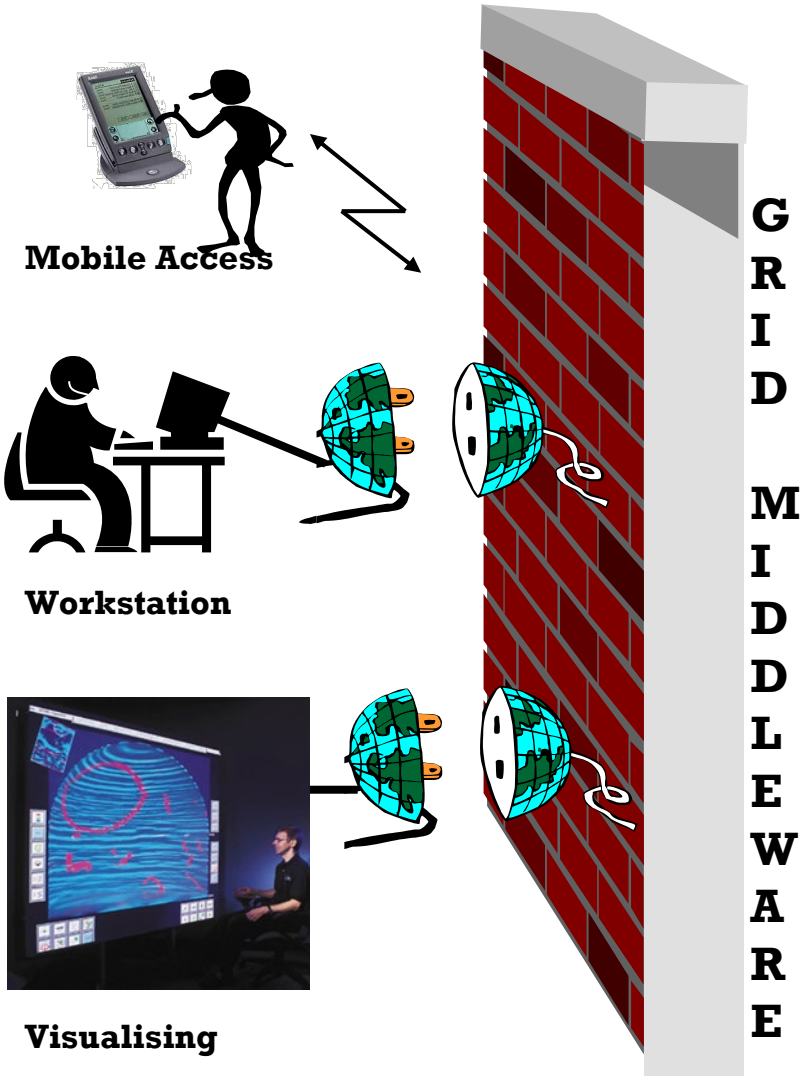
There is no GRID owner!

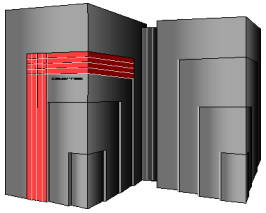
Power Grid Similarity

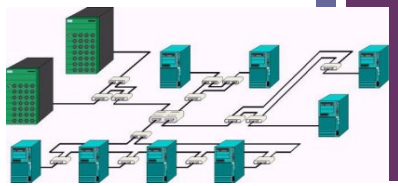


"We will probably see the spread of computer utilities, which, like present electric and telephone utilities, will service individual homes and offices across the country" (Len Kleinrock, 1969)

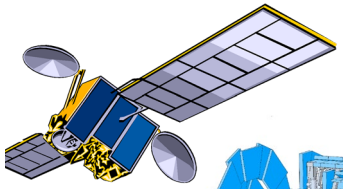
The Grid Paradigm

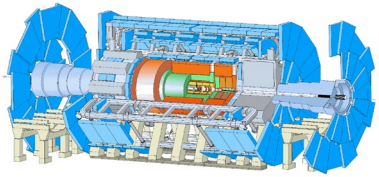




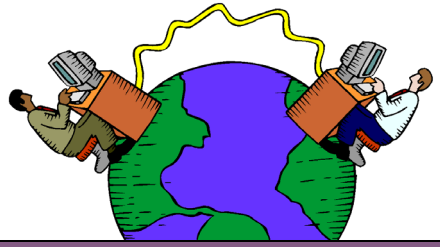


Supercomputer, PC-Cluster





Data-storage, Sensors, Experiments



Internet, networks

Grid Types

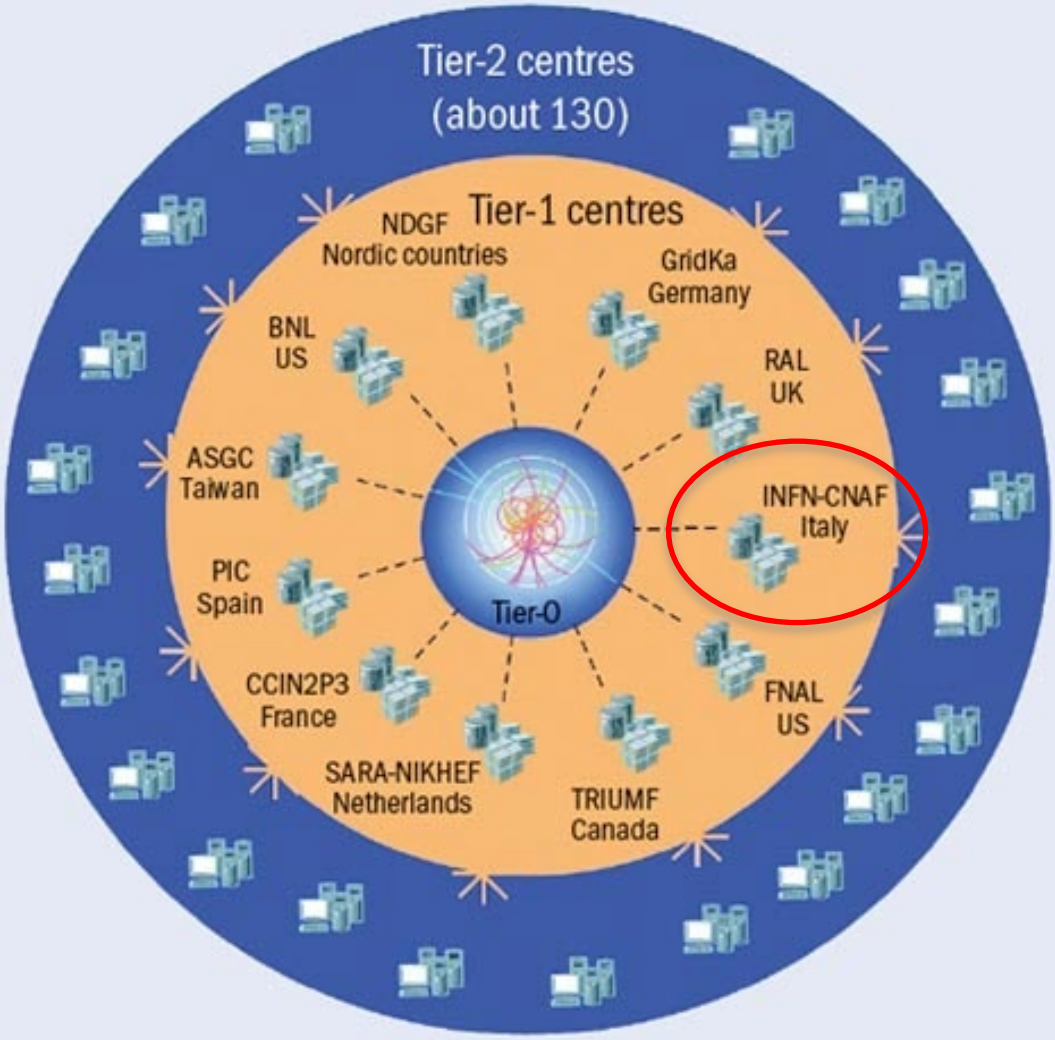
Supercomputer Based Service Grid

Cluster Based Service Grid

Volunteer Desktop Grid

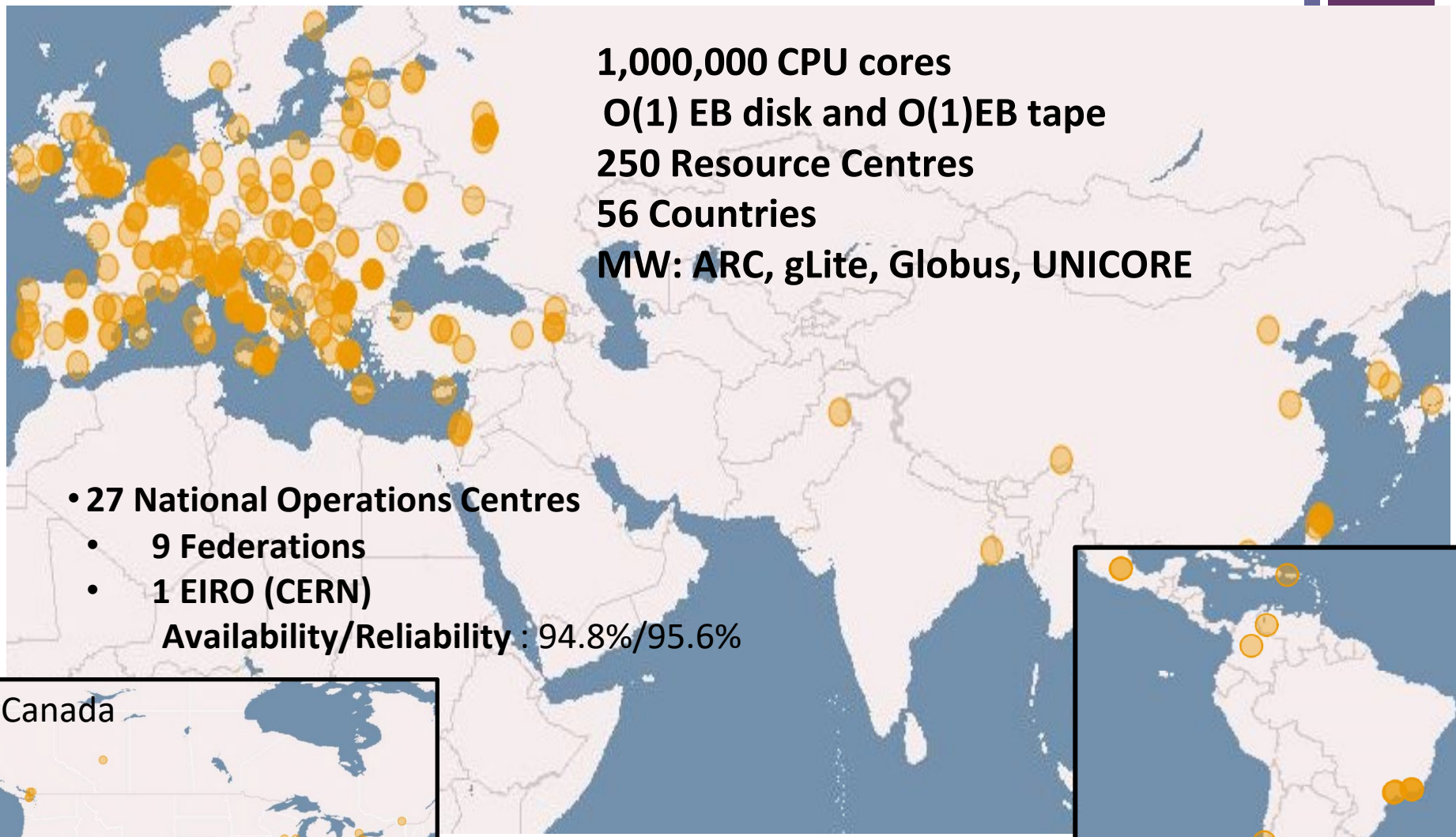
Grid of Clusters – WLCG Service Grid

High Throughput Computing



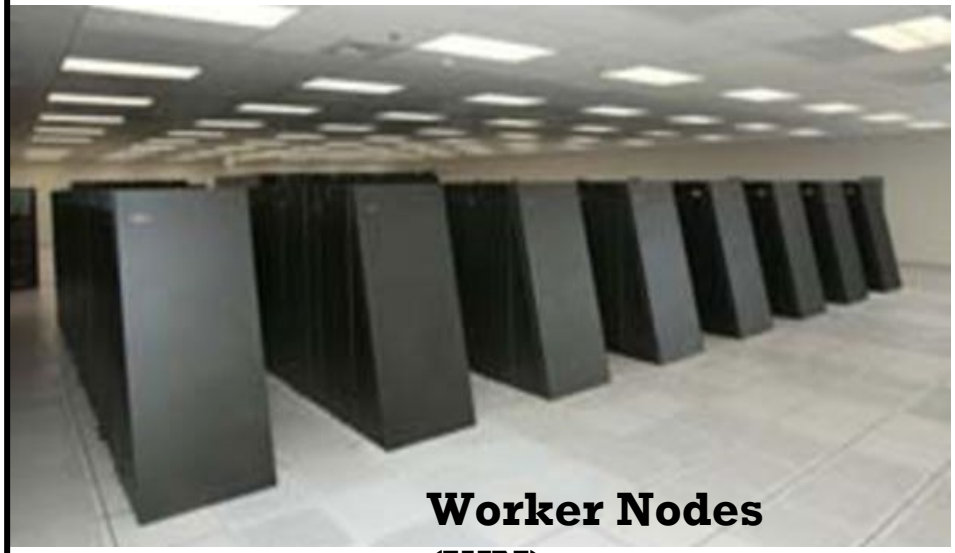
- **Worldwide LHC computing Grid**
- Service GRID for the LHC high energy physics experiments
- Tiered structure
- Part of the European grid Infrastructure (EGI)
  - O(1M) logical CPUs
  - O(1) EB disk
  - O(1) EB tape

The European Grid Infrastructure



EGI job flow in real-time

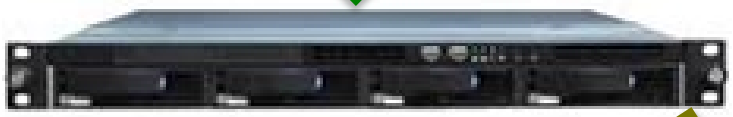
A typical Grid Site



Worker Nodes (WN)

Storag e

Batch System (PBS,LSF…)

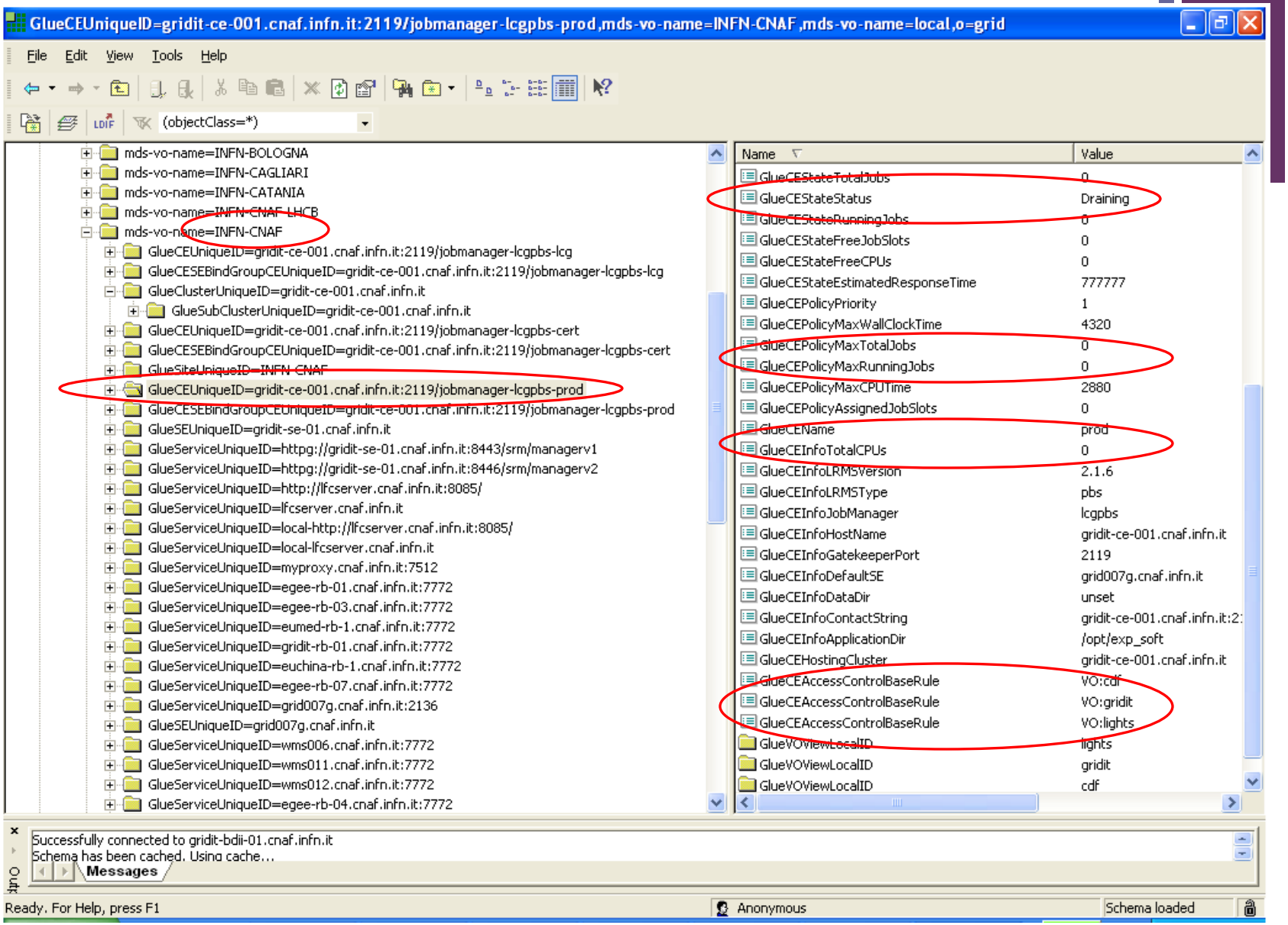


Computing Element (CE)

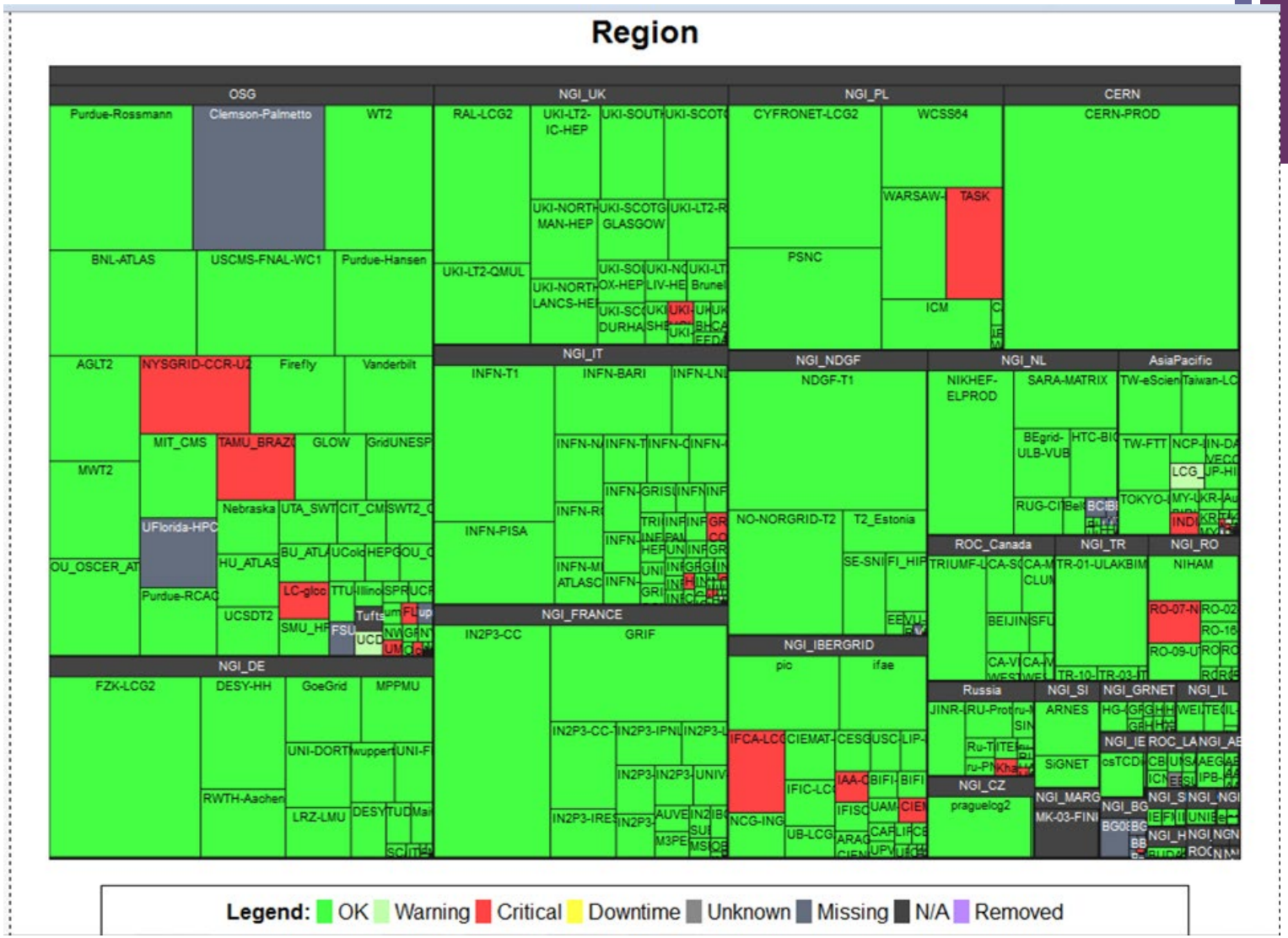
Storage Element (SE)

Site BDII – Information system

Standard Interface: The Information System



Operations Tools: Monitoring



Anything that can go wrong, will

- In simple and complex environment errors will happen
- Some errors are preventable, some are manageable by the infrastructure, some can only be managed by the user

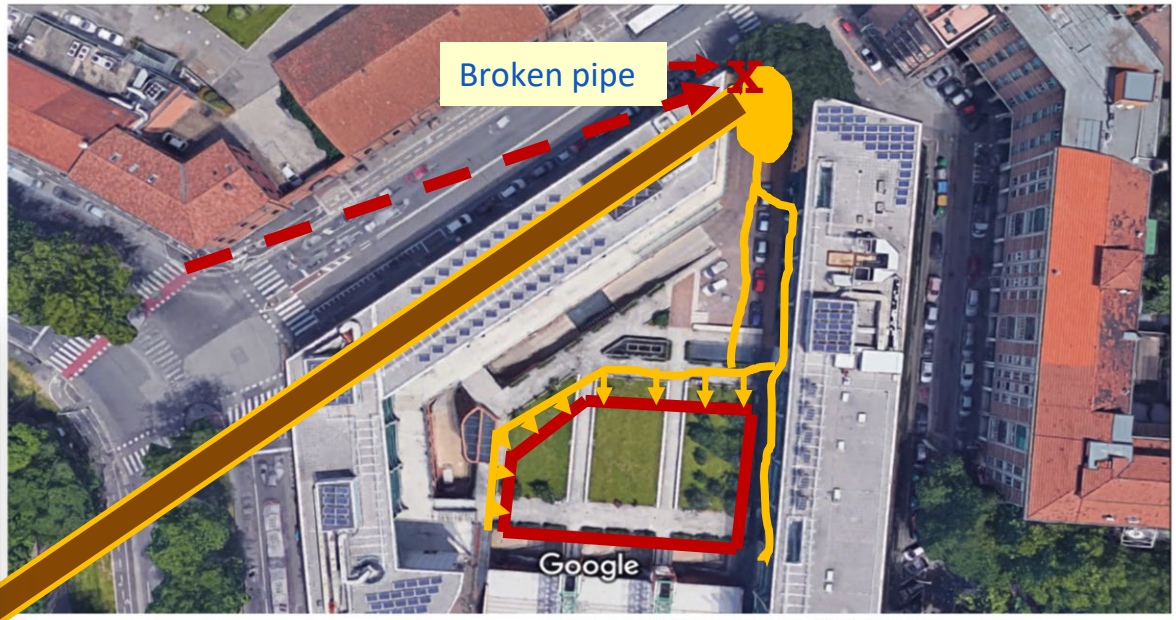
“A distributed system is one in which the failure of a computer you didn‘t even know existed can render your own computer unusable.”
Leslie Lamport

Expect the Unexpected

- When services / servers don't respond or return an invalid status/ message;
- When the air-conditioning / power fails (again & again & again);
- When disks fail and you have to recover from backup – but the tapes have been overwritten;
- When a service engineer puts a Coke into a machine to 'warm it up'
- When Oracle returns you someone's else data
- When a fishing trawler cuts a trans-Atlantic network cable;
- When a Tsunami does the equivalent in Asia Pacific;

All these things really happened ©Jamie Shiers
2008 J. Phys.: Conf. Ser. 119 052030

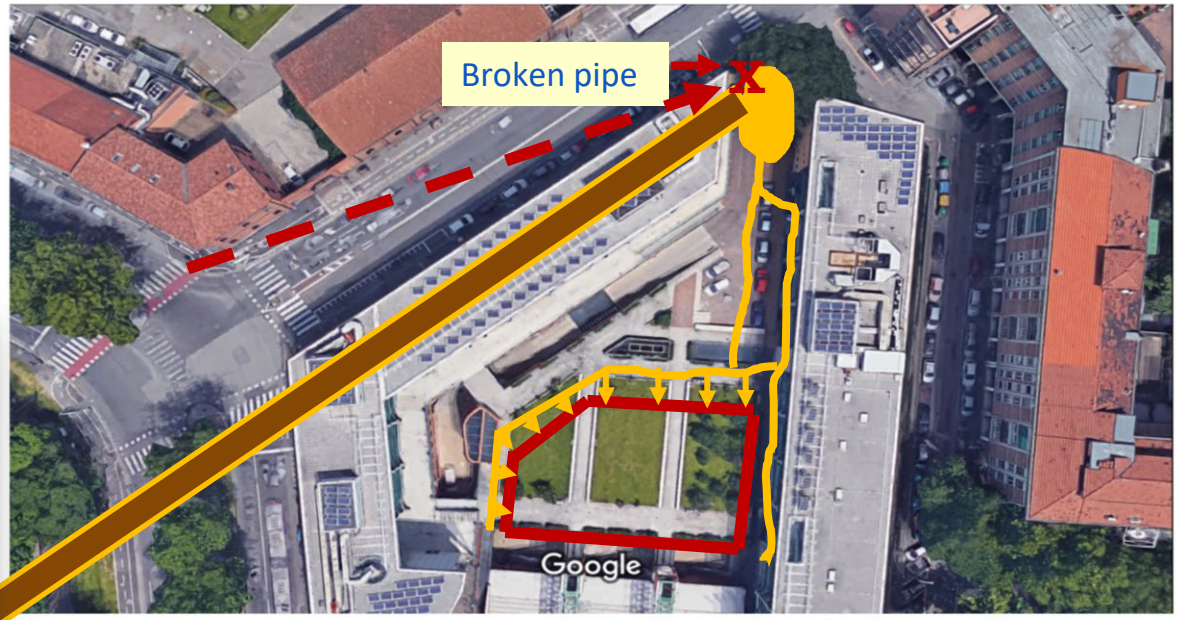
9/11 2017…in Bologna….



The broken pipe

9/11 2017…in Bologna…

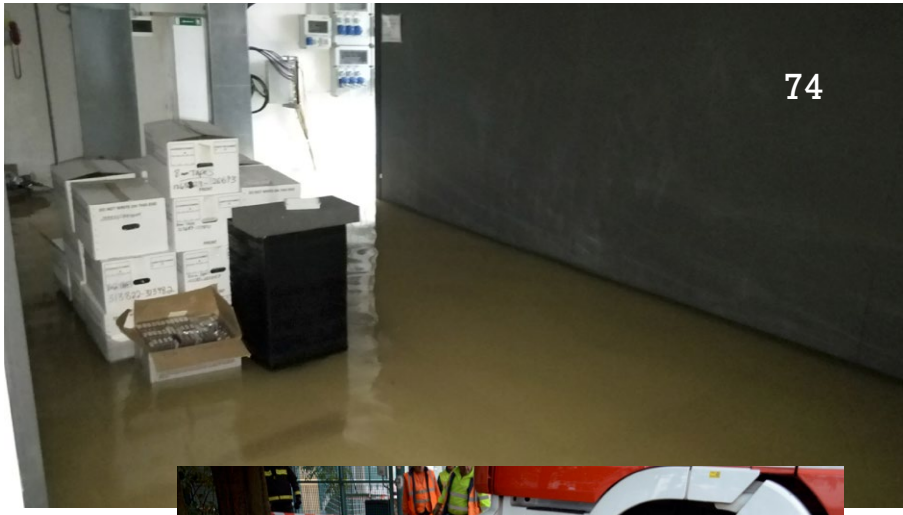
- The flood happened on November 9 early in the morning
  - Breaking of one of the main water pipelines in Bologna
  - CNAF was flooded

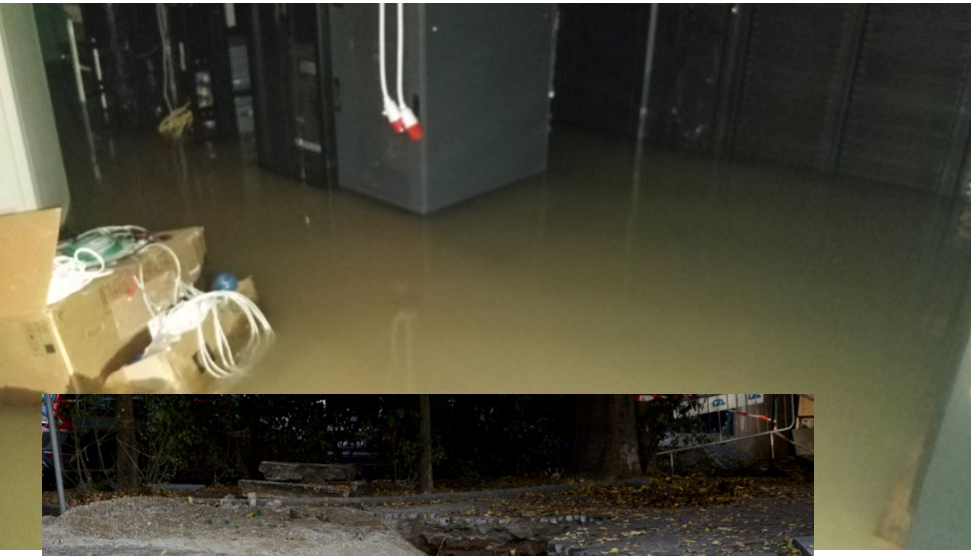


The broken pipe

The Tier-1 entrance that morning

All Tier-1 doors are watertight
Height of water outside: 50 cm
Height of water inside: 10 cm (on floating floor) for a total volume of ~500 m^3






115

Grid Security: Virtual Organizations

- Resources are accessible to members of VOs
- The owner of the resources decides which VOs are authorized

Grid Security

- ***Single sign on***. *Users must be able to "log on" (authenticate) just once and then have* access to multiple Grid resources, without further user intervention.

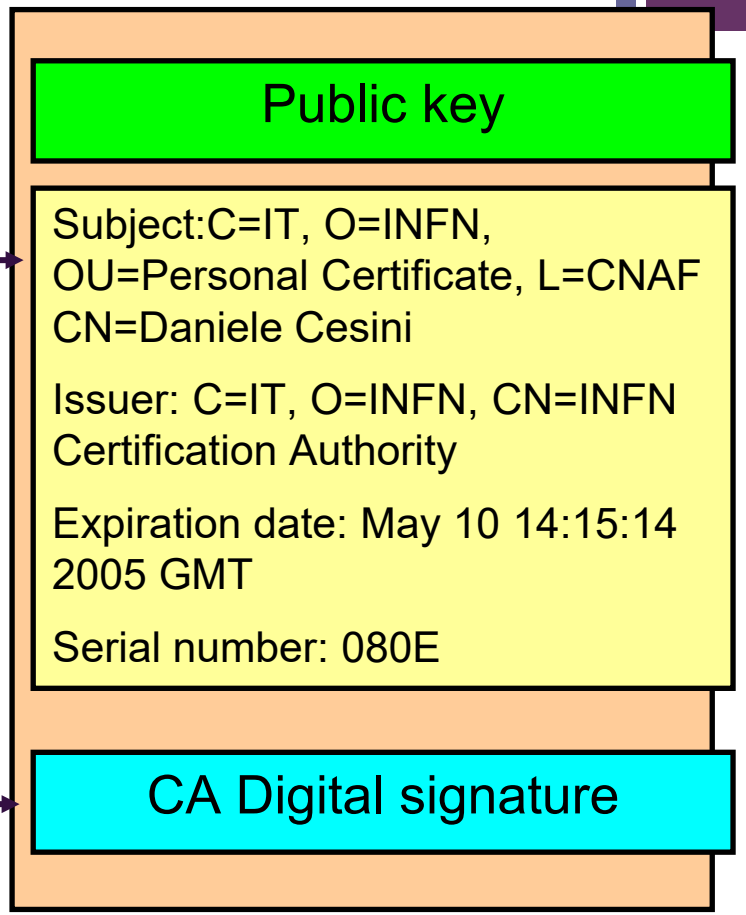
- ***Delegation***. *A user must be able to endow a program with the ability to run* on that user's behalf, so that the program is able to access the resources on which the user is authorized.

Grid Security: X509 Certificates

An X.509 Certificate contains:

- owner's public key;
- identity of the owner;
- info on the CA;
- time of validity;
- Serial number;
- digital signature of the CA

Structure of a X.509 certificate

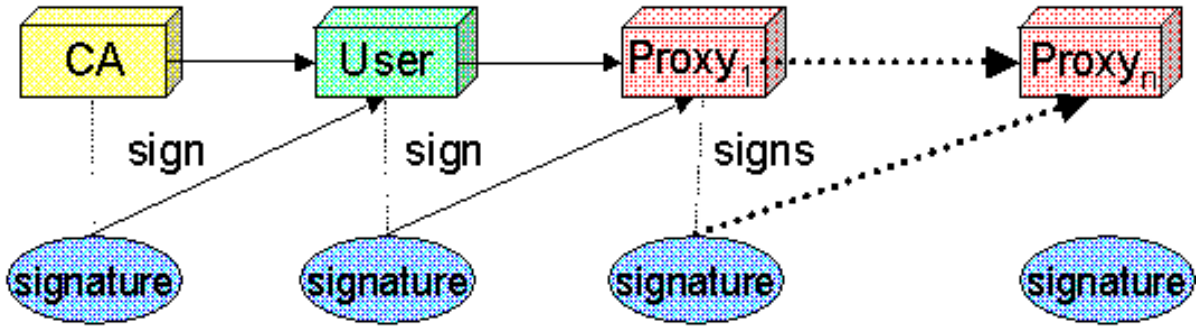


Grid Security: Proxy Certificates

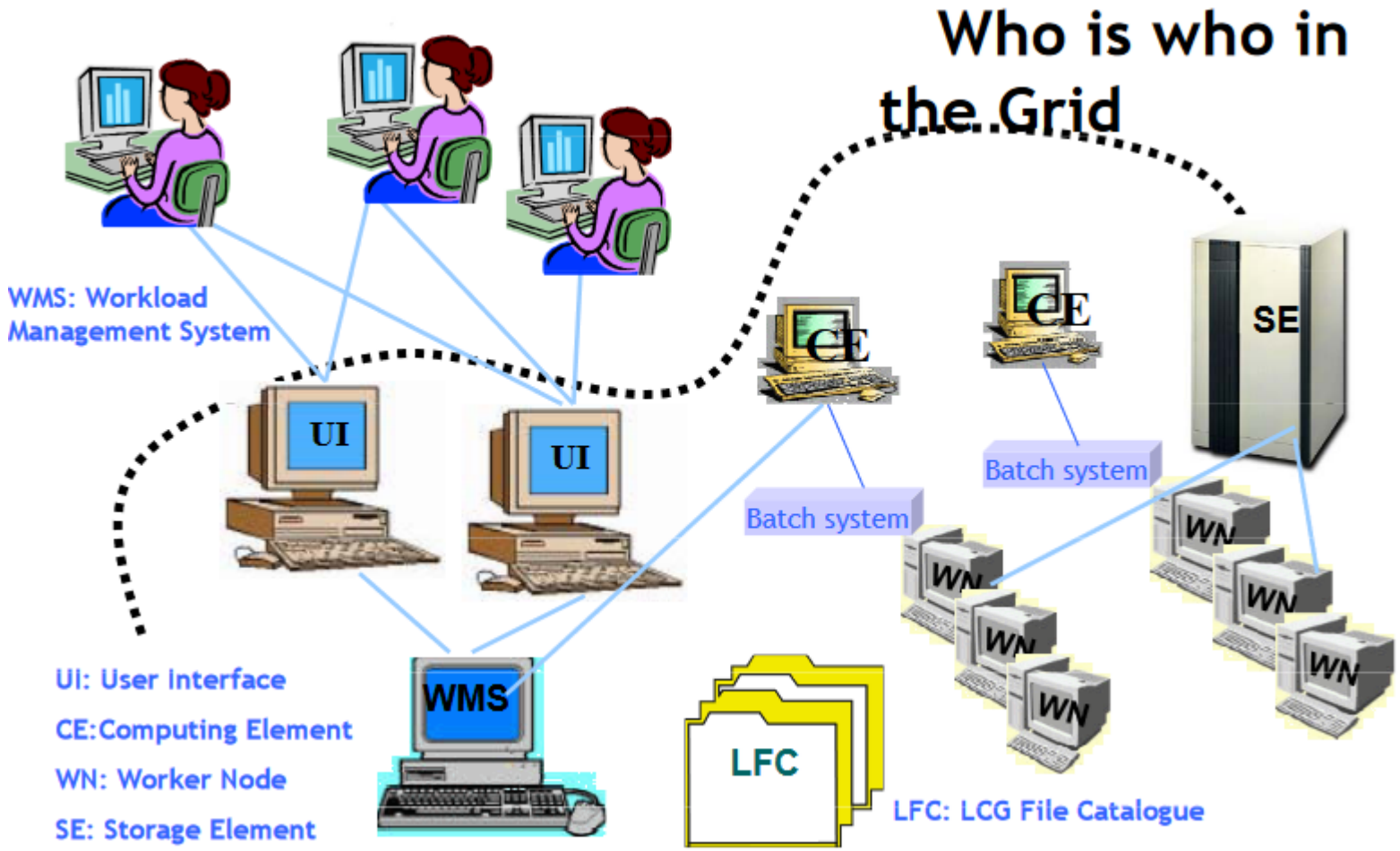
X.509 Proxy Certificate $\equiv$ GSI extension to X.509 Identity Certificates

- Has a limited lifetime
- Is signed by the normal end entity certificate or by another proxy
- Delegation = remote creation of a (second level) proxy credential

Allows remote process to authenticate on behalf of the user



The Job Submission Actors



The single batch Grid Job

JOB Type	JobType = “Normal";
Prologue	Prologue = "prologue.sh";
Input SandBox	InputSandbox = {"test.sh", "fileA"};
Requirements	Requirements = false;
Executable	Executable = "test.sh";
Std Output/Error	StdOutput = "std.out"; StdError = "std.err";
Output SandBox	OutputSandbox={"std.out", "std.err"};
Epilogue	Epilogue = "compress.sh";
Error Recovery	RetryCount = 1; ShallowRetryCount = 2;

The Grid JOB

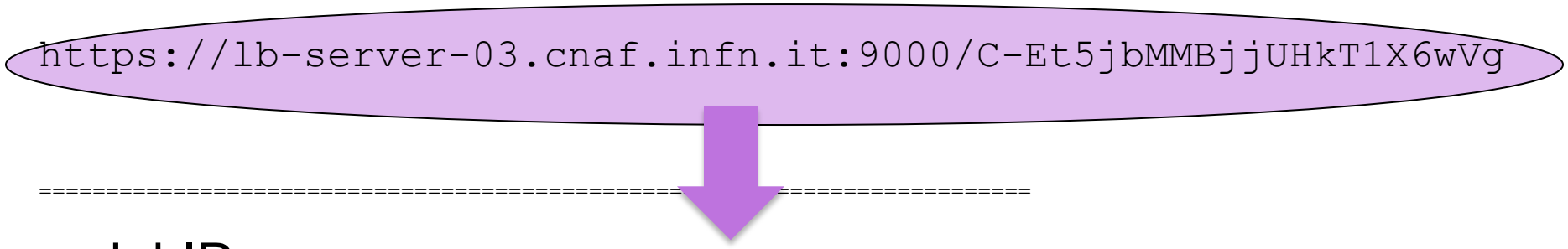

```
====================== glite-wms-job-submit Success ======================


The job has been successfully submitted to the WMProxy

Your job identifier is:


https://lb-server-03.cnaf.infn.it:9000/C-Et5jbMMBjjUHkT1X6wVg


==========================================================================
```



JobID:

- Upon submission each job is assigned a unique, virtually non-recyclable job identifier in an URL form.
- The first part is the url of the server that accepted the job
- The remainder is a random generated sequence: the Grid is a highly decentralized system, characterized by lack of unified control no serial numbering is possible

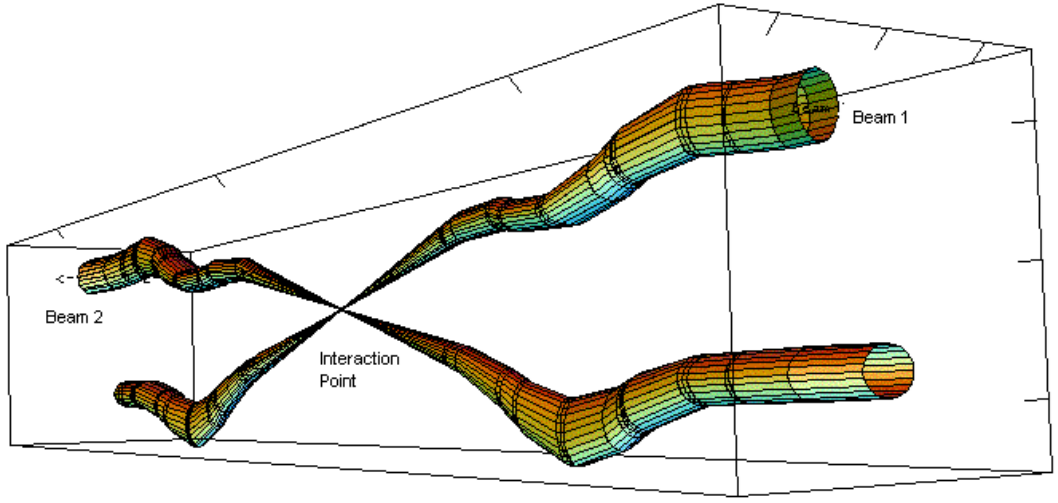
Which Applications for the HTC?

- Computing
  - embarrassingly parallel jobs
    - Single, Parametric, DAG, Collections
  - "small" MPI/OPENMP jobs
  - Opportunistic Usage

- Data
  - Flat Files
  - Write Once, Read Many
  - Need to be shared among many organizations

High Energy Physics

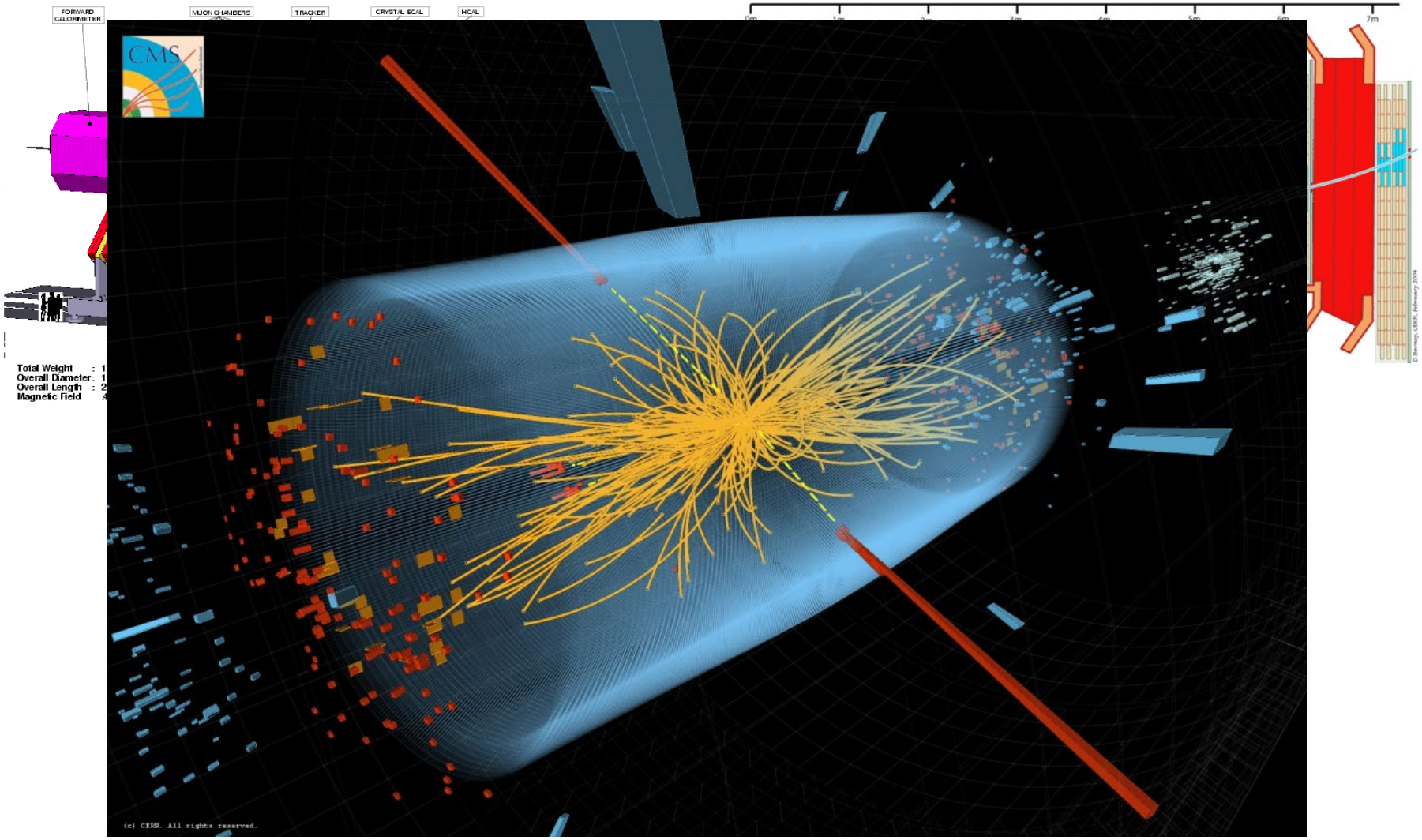
Large Hadron Collider (LHC)
60 PB /year



Relative beam sizes around IP1 (Atlas) in collision

40MHz peak crossing rate (25ns)

600 million inelastic events per second.


FORWARD CALORIMETER
MUON CHAMBERS
TRACKER
CRYSTAL ECAL
HCAL
CMS
Total Weight
Overall Diameter
Overall Length
Magnetic Field
(c) CERN. All rights reserved.

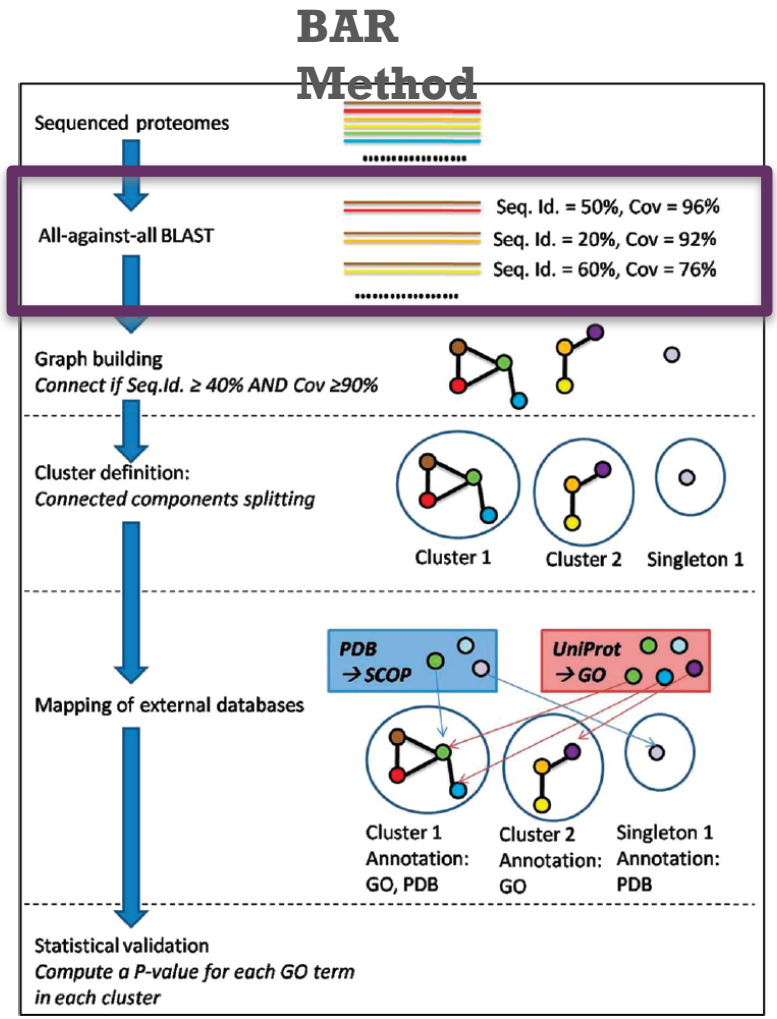
Protein Annotation

Protein annotation is essential for understanding diseases at the molecular level

Bologna Annotation Resource (BAR) method (*) uses BLAST for aligning the sequence: the number of computations is the cartesian product of the number of the input sequences.

$\sim 17 \times 10^{13}$

Problem: the amount of time to do these alignments is estimated in seven years on the local resources of the group that run the research



The alignment step was griddified

(*) J PROTEOME RES. 2009 SEP; 8(9); 4362-71. - R. Casadio et al. **"The Bologna Annotation Resource: A Non Hierarchical Method For The Functional And Structural Annotation Of Protein Sequences Relying On A Comparative Large-scale Genome Analysis."**

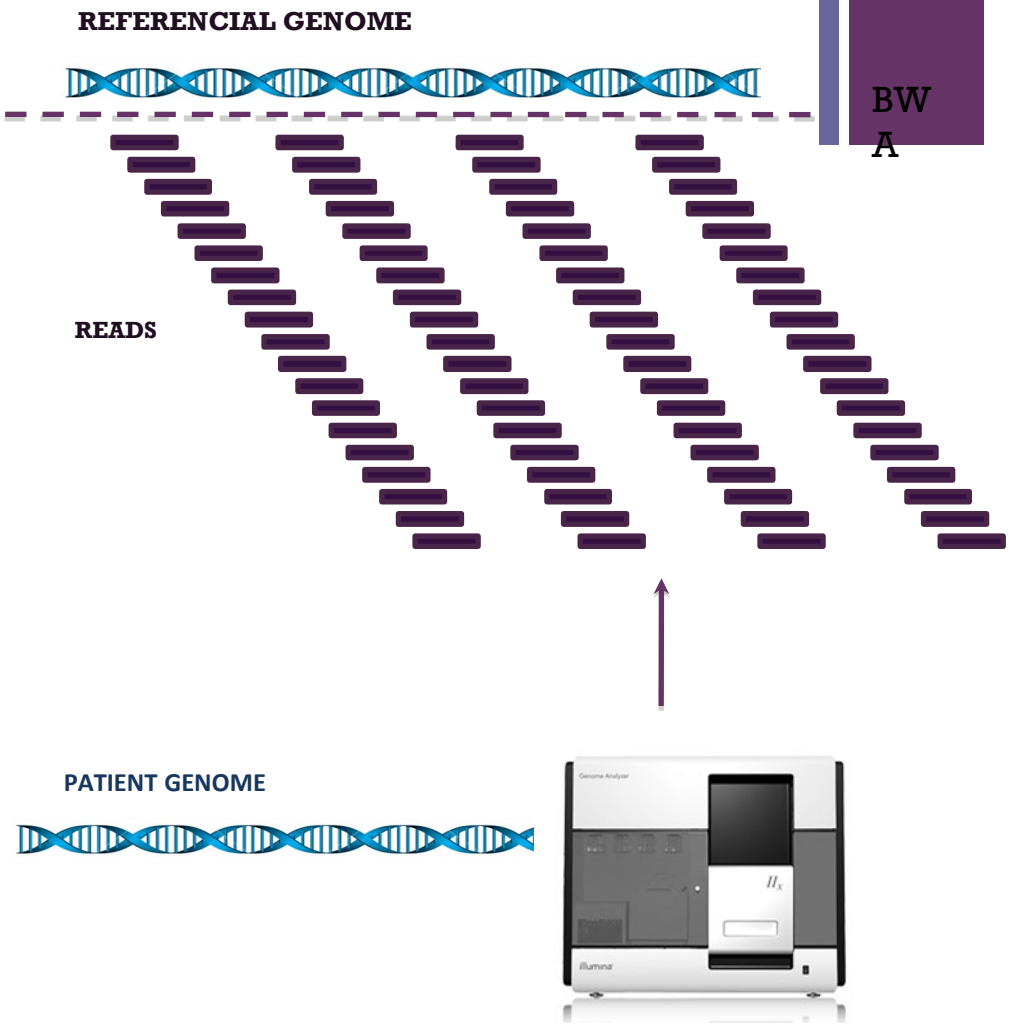
Massively Genome Sequencing

Used in the study of cancer Diseases

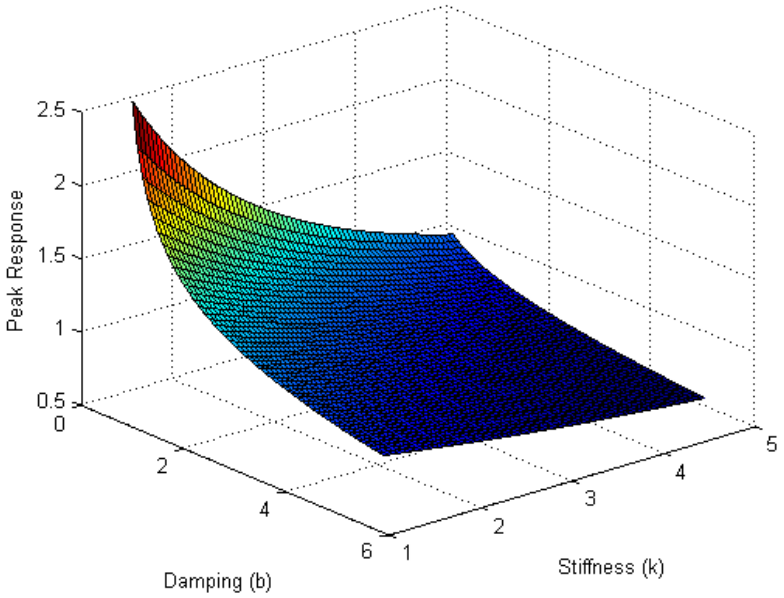
Allows massive amount of DNA or RNA fragments to be sequenced in a single experiment.

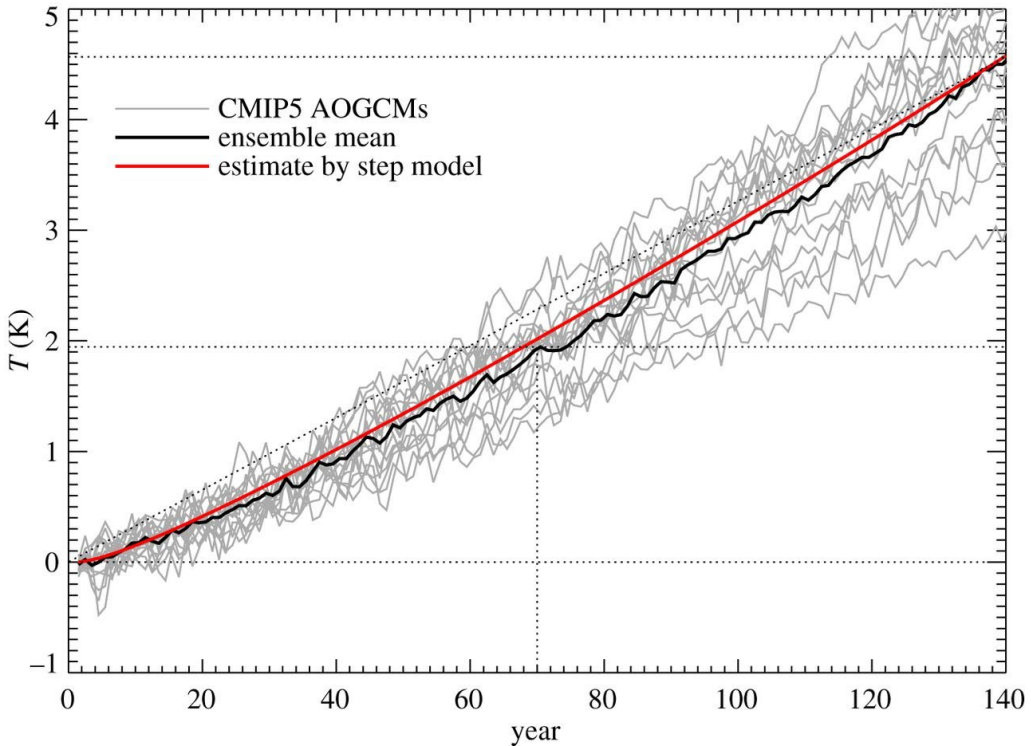
For the *massively parallel sequencing it is used* **BWA** tool (Burrows-Wheeler Aligner) for indexing and alignment

- Memory request~ 3,5 GB
- Total time ~ 50h using the group local resources



Parameter sweep and ensemble simulations





References

- https://en.wikipedia.org/wiki/IOPS
- https://en.wikipedia.org/wiki/Template_talk:Bit_and_byte_prefixes
- https://en.wikipedia.org/wiki/Standard_RAID_levels
- https://www.cavium.com/Documents/TechnologyBriefs/Adapters/Tech_Brief_Introduction_to_Ethernet_Latency.pdf
- https://en.wikipedia.org/wiki/Storage_area_network
- https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/fairshare_about_lsf.html
- https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/backfill.html
- https://community.fs.com/blog/do-you-know-the-differences-between-hubs-switches-and-routers.html
  https://en.wikipedia.org/wiki/Computer_cooling
- https://en.wikipedia.org/wiki/Power_usage_effectivenes
- https://puppet.com/
- https://www.theforeman.org/
- https://grafana.com/