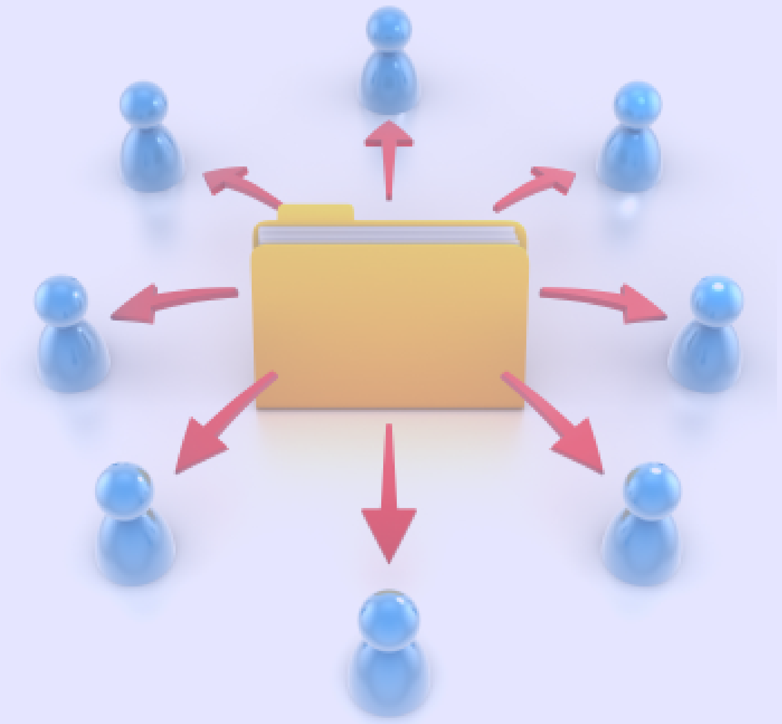
Master in High Performance Computing

Moreno Baricevic
Stefano Cozzini

CNR-IOM DEMOCRITOS
Trieste, ITALY



Resource Management

Scuola Internazionale Superiore
di Studi Avanzati

The Abdus Salam
International Centre
for Theoretical Physics

RESOURCE MANAGEMENT

We have a pool of users and a pool of resources, then what?

- some software that controls available resources
- some other software that decides which application to execute based on available resources
- some other software devoted to actually execute applications

RESOURCE MANAGEMENT

The resource manager allows:

- better resource control
- better resource utilization
- better access control

Some definitions (1/2)

Parallel computing

The simultaneous execution of a task split up on multiple processors in order to obtain results faster.

Distributed computing

Same thing but with many computers (concept of network).

Cluster

Group of linked computers working together (can be seen as a single computer).

Some definitions (2/2)

Batch Scheduler

Software responsible for scheduling the users' jobs on the cluster.

Resources Manager

Software that enable the jobs to connect the nodes and run.

Node (aka Computing Node)

Computer used for its computational power.

Frontend

It's through this node that the users will submit/launch/manage jobs.

Access Node

A cluster is usually isolated from outside for security purpose, this node is the access gateway.

Master Node

Management server, that might as well act as frontend and access node.

Management of Jobs and Resources

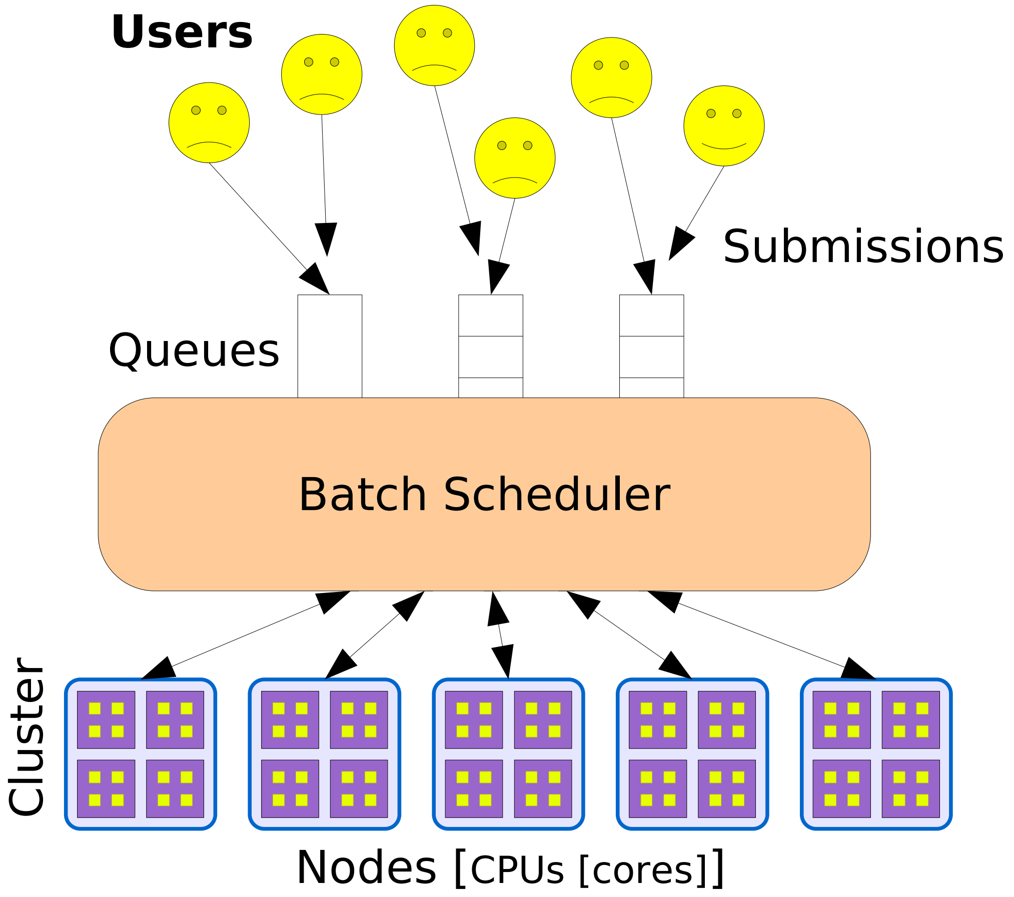
Management: Batch Scheduler and Resource Manager

- Submission
- Scheduling
- Resources Allocation
- Job Launch
- Monitoring, logging...

2 layers

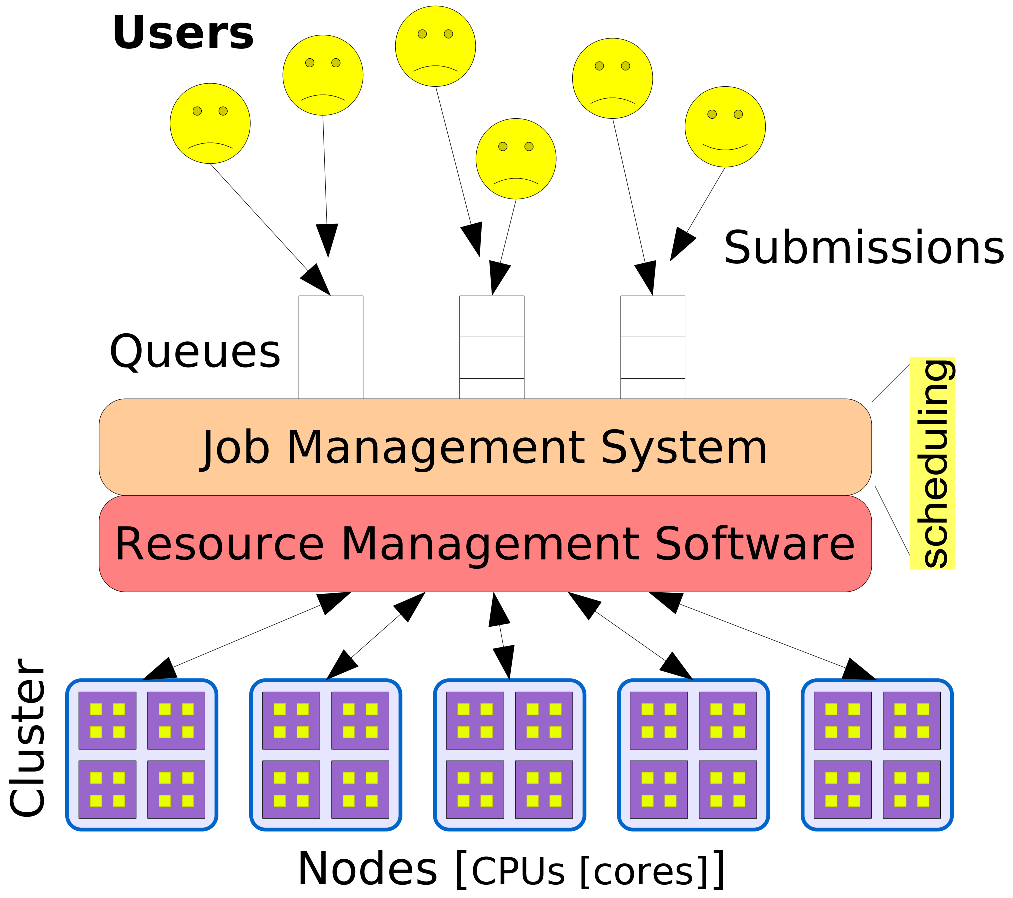
- Resource Management Layer: launching, cleaning, monitoring...
- Job Management Layer: batch/interactive job, Scheduling, Suspend/Resume, Preemption, Dependencies, Resubmission, Advance Reservation...

Batch Scheduler: a Global Picture 1



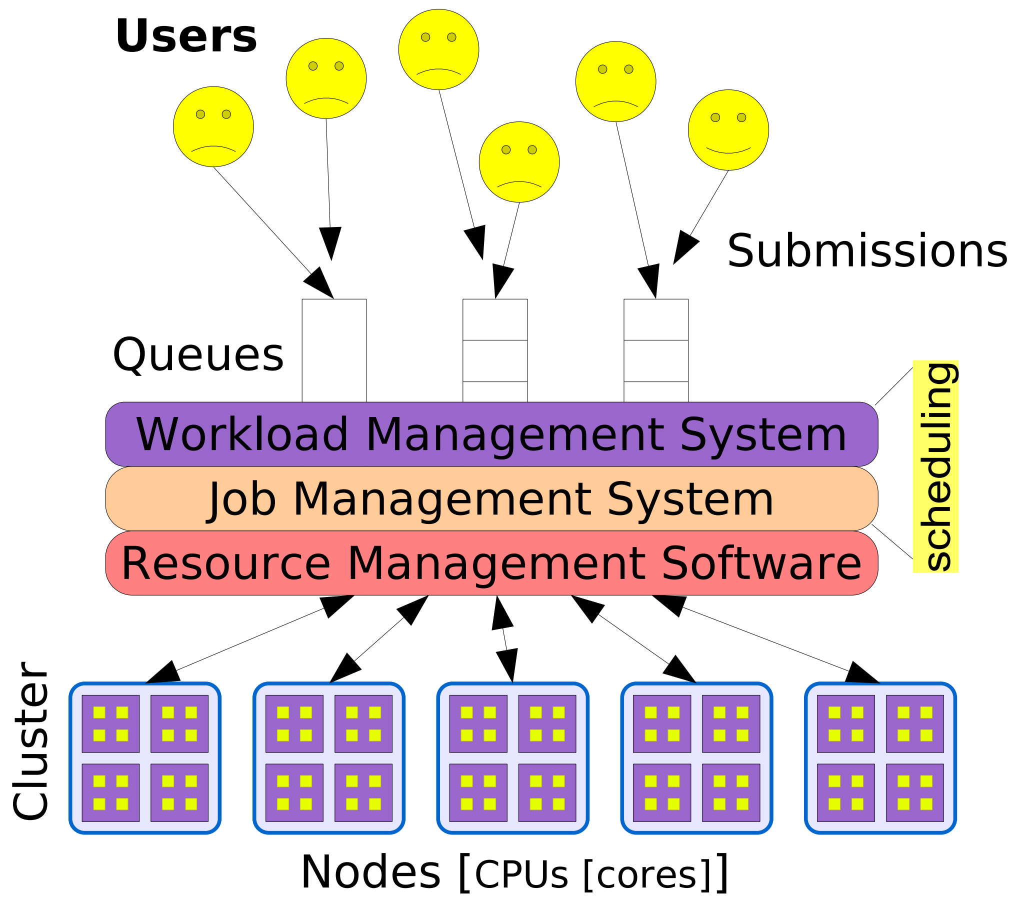
- Goals: Allocate resources for each applications with respect of their requirements and users' rights. Satisfy users (response time, reliability) and administrators (high resource utilization, efficiency, energy management...).
- Loadleveler(IBM), PBS, Torque, LSF, Slurm(LLNL), SGE/OGE, Condor, OAR

Batch Scheduler: a Global Picture 2



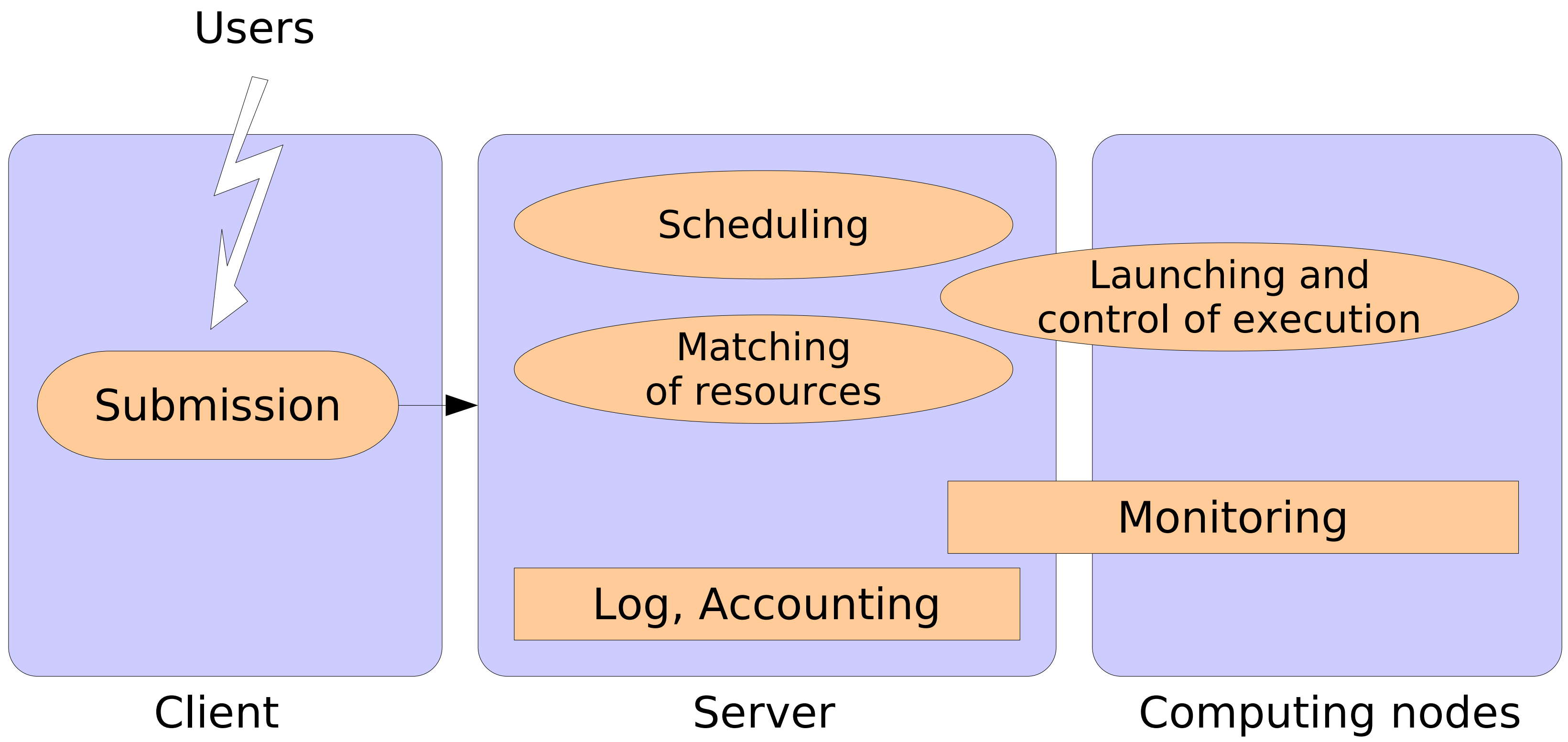
- Resource Management Layer: launching, cleaning, monitoring...
- Job Management Layer: batch/interactive job, Backfilling (EASY or Conservative) Scheduling, Suspend/Resume, Preemption, Dependencies, Resubmission, Advance Reservation...

Batch Scheduler: a Global Picture 3



- Workload/Job Management: more complete job scheduling policies
- Fairsharing, Quality of Service (QoS), SLA (Service Level Agreement), Energy Saving, Time Varying Policies (day/night, week-end, holidays ...)
- Dedicated software: MAUI and Catalina
- There is not true separation into some systems, for instance Slurm and OAR.

Architecture and main components

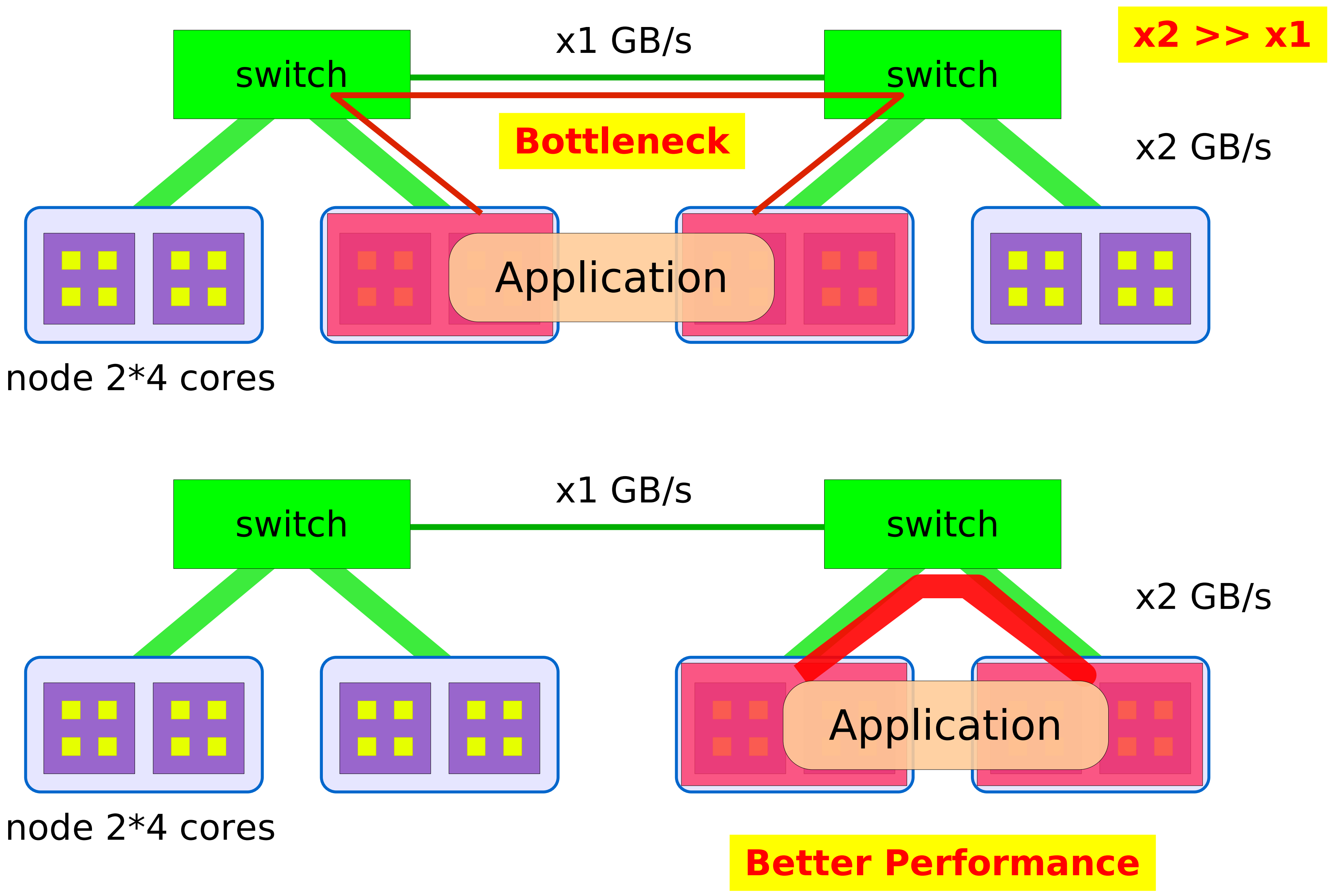


- Few components, but the number of jobs and resources states, plus the scheduling policies and a huge number of congurable parameters, lead to a great system complexity.
- It's not so easy to tune and to optimize a Batch Scheduler.

Challenges, Recent Features and Trends

- Scalability (remains the number one issue)
- Topology constraint (hierarchy, NUMA, I/O Bandwidth)
- Energy Saving (node power on/off, DVFS, not so simple)
- Dynamic jobs, massive submission
- Infrastructure diversity (virtual compute node, multi-cluster, GPGPU...)
- Master the increase of (global) complexity
- How to track the global efficiency of the global computing infrastructure (and how to optimize it) ?

Topology-aware Scheduling



Scalability

Which granularity for resource representation and manipulation

- core, thread (too fine)? (generally a flat data structure in batch scheduler)
- nodes (most used) (Slurm can manage upto 64K nodes, how many cores ?)
- add some policies for fine tuning (cpuset, cgroup, CPU affinity, Bulk I/O, (next steps bandwidth)...)
- partitions (set of nodes) (sometimes used in large cluster)

Other resources issues

- Memory, network cards, L3 Cache partitioning (Power 7), DVFS control...

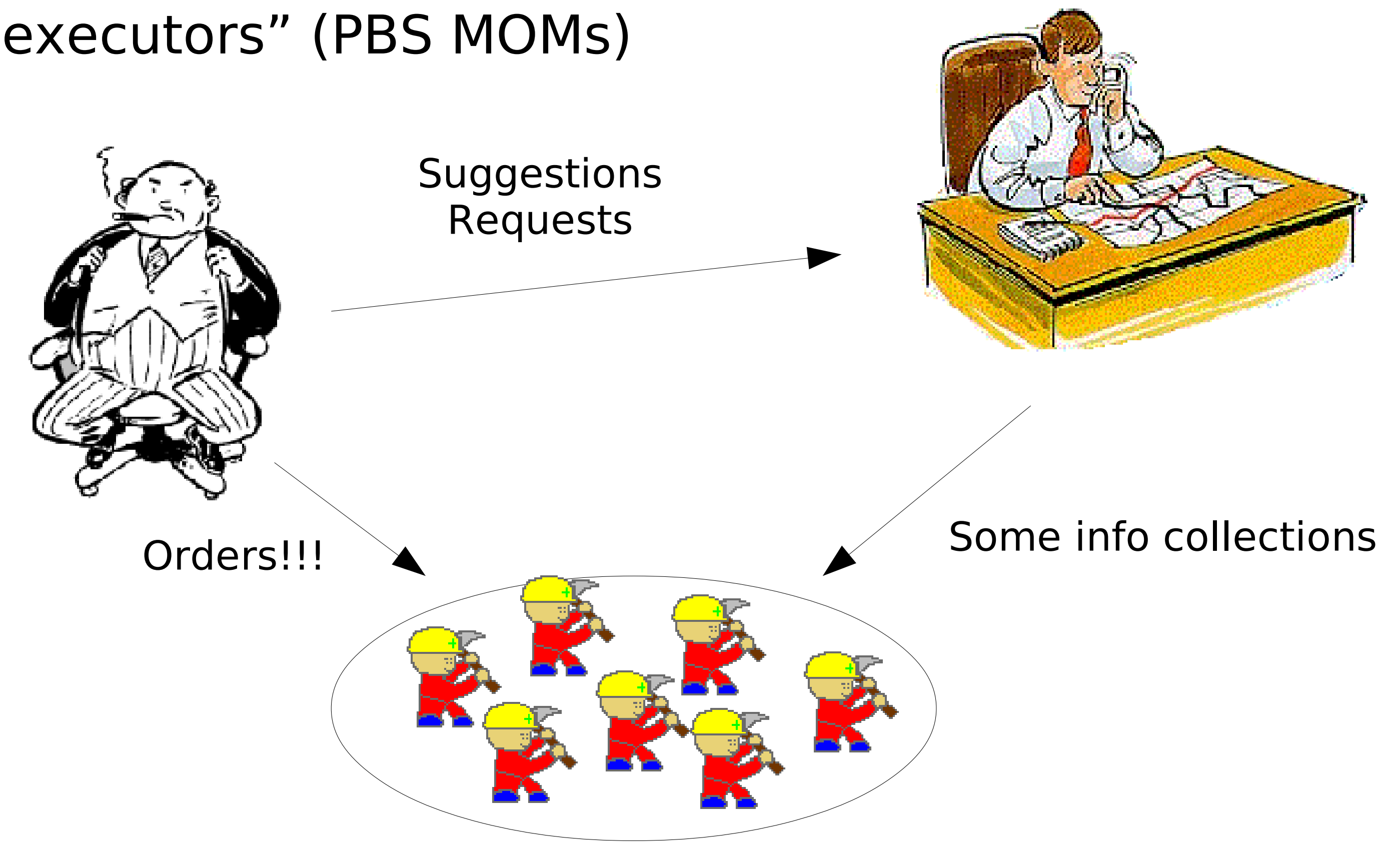
RESOURCE MANAGEMENT

The scheduler should have:

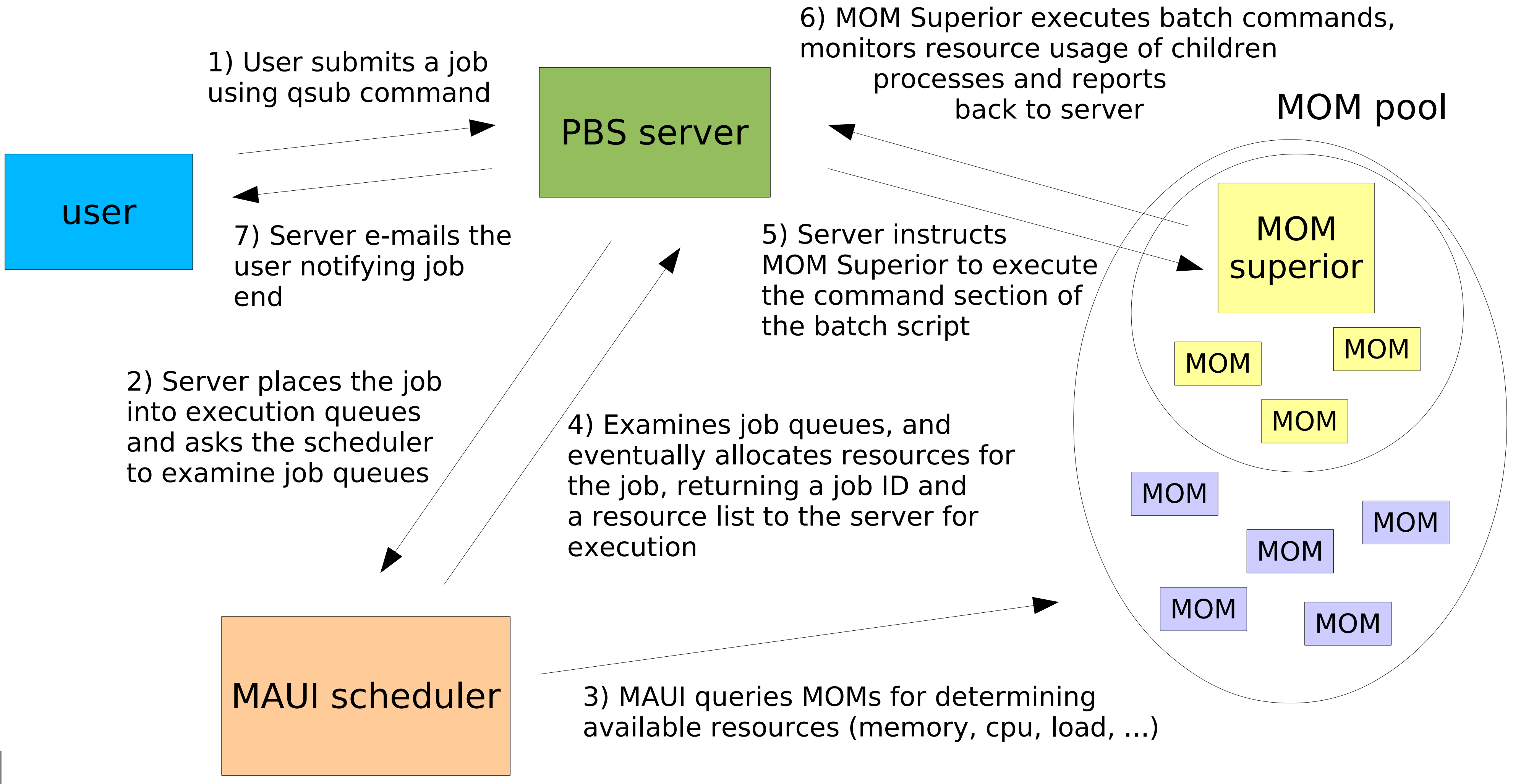
- **Fair Share mechanism**
- **Backfill scheduling algorithm**
- reservations for high priority jobs
- more control parameters on users
- commands for querying the scheduler

The Queue System - PBS/TORQUE + MAUI

- General Components
  - A resource manager (PBS server)
  - A scheduler (MAUI scheduler)
  - Many “executors” (PBS MOMs)



A typical job session



1) User submits a job using qsub command
PBS server
user
7) Server e-mails the user notifying job end
6) MOM Superior executes batch commands, monitors resource usage of children processes and reports back to server
MOM pool
MOM superior
5) Server instructs MOM Superior to execute the command section of the batch script
MOM
MOM
MOM
2) Server places the job into execution queues and asks the scheduler to examine job queues
4) Examines job queues, and eventually allocates resources for the job, returning a job ID and a resource list to the server for execution
MOM
MOM
MOM
MOM
MOM
MAUI scheduler
3) MAUI queries MOMs for determining available resources (memory, cpu, load, ...)

Fair sharing

Fairshare is a mechanism which allows historical resource utilization information to be incorporated into job feasibility and priority decisions.

Fairshare information only affects the job's priority relative to other jobs.

Using the standard fairshare target

- the priority of jobs of a particular group which has used too many resources over the specified fairshare window is lowered
- the priority of jobs which have not received enough resources will be increased

Fair sharing – How it works

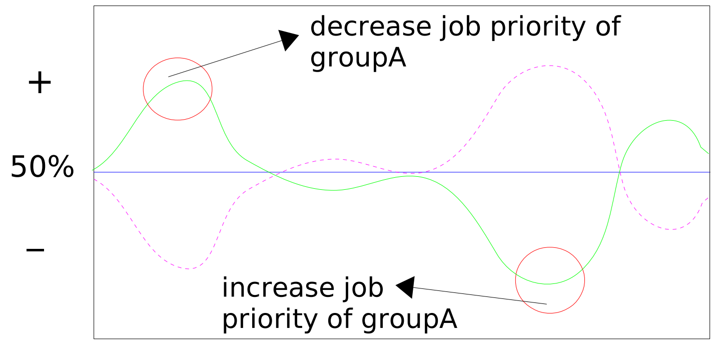
- At the beginning all the jobs are created equals (in term of priority)
- However some jobs are more/less equal than others
- Priority is increased/decreased when the fair sharing quota is below/above from its target
- Gain/lost in priority:
  - is configurable
  - 1% far from fair share means 4 hours on queues (DEMOCRITOS example)

```
GROUPCFG[groupA]    FSTARGET=50%    PRIORITY=5000
GROUPCFG[groupB]    FSTARGET=50%    PRIORITY=5000
```

Assume groupA has 50% of fairshare usage. When it uses more resources than those assigned, the priority of the jobs will be decreased; when it uses less resources, the priority of its jobs will be increased.

When a group is not computing, the other groups can benefit from the available resources

- better resource utilization
  - no idle CPUs



Backfill 1/2

Backfill is a scheduling optimization which allows a scheduler to make better use of available resources by running jobs out of order.

Consider this example with a 10 CPUs machine:

Job1 (priority=20 walltime=10 nodes=6)
Job2 (priority=50 walltime=30 nodes=4)
Job3 (priority=40 walltime=20 nodes=4)
Job4 (priority=10 walltime=10 nodes=1)

1) When Maui schedules, it prioritizes the jobs in the queue according to a number of factors and then re-orders the jobs into a 'highest priority first' sorted list.

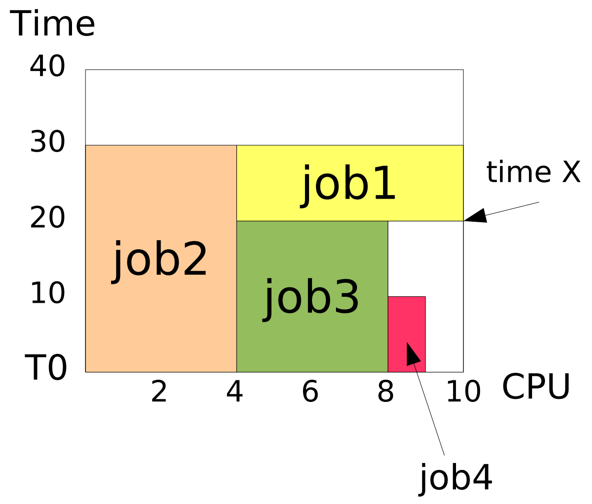
Sorted list:

Job2 (priority=50 walltime=30 nodes=4)
Job3 (priority=40 walltime=20 nodes=4)
Job1 (priority=20 walltime=10 nodes=6)
Job4 (priority=10 walltime=10 nodes=1)

Backfill 2/2

2) It starts the jobs one by one stepping through the priority list until it reaches a job which it cannot start.

3) All jobs and reservations have a start time and a walltime limit, so MAUI can determine:
   - the completion time of all jobs in the queue
   - the earliest the needed resources will become available for the highest priority job to start (time X)
   - which jobs can be started without delaying this job (job4)

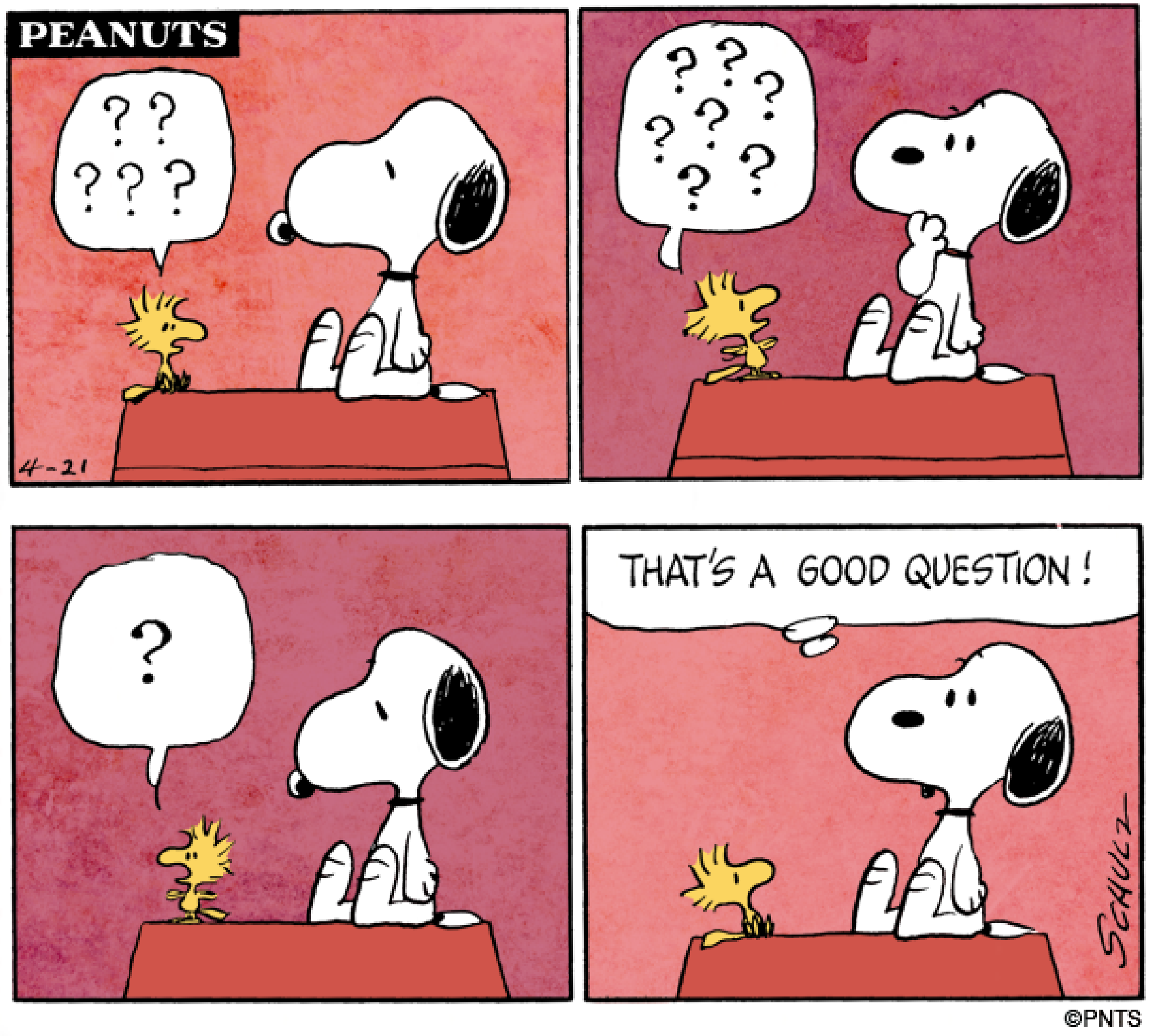


Job2 (priority=50 walltime=30 nodes=4)
Job3 (priority=40 walltime=20 nodes=4)
Job1 (priority=20 walltime=10 nodes=6)
Job4 (priority=10 walltime=10 nodes=1)

- Enabling backfill allows the scheduler to start other, lower-priority jobs so long as they do not delay the highest priority job, essentially filling in holes in node space.

- Backfill offers significant scheduler performance improvement:
  - increased system utilization by around 20% and improved turnaround time by an even greater amount in a typical large system
  - backfill tends to favor smaller and shorter running jobs more than larger and longer running ones: It is common to see over 90% of these small and short jobs backfilled.

Questions?



That's All Folks!

```
( questions ; comments ) | mail -s uheilaaa baro@democritos.it

( complaints ; insults ) &>/dev/null
```