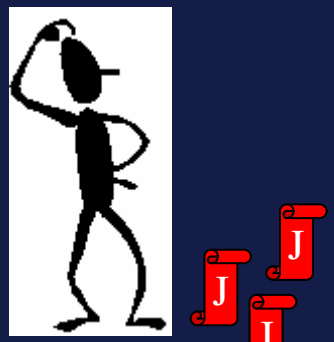
Managed by

A two-level scheduler to dynamically schedule a stream of batch jobs in large-scale grids

M. Pasquali, R. Baraglia, G. Capannini, L. Ricci, and D. Laforenza

7th Meeting of the Institute on Resource Management and Scheduling (WP6)

Problem description



Meta Scheduler

Scheduler

Scheduler

Scheduler

Scheduler

Multi-Level Scheduler Objectives

- Scheduling the arriving jobs respecting their QoS requirements,

  Deadline, Submitting User, Licenses, etc.

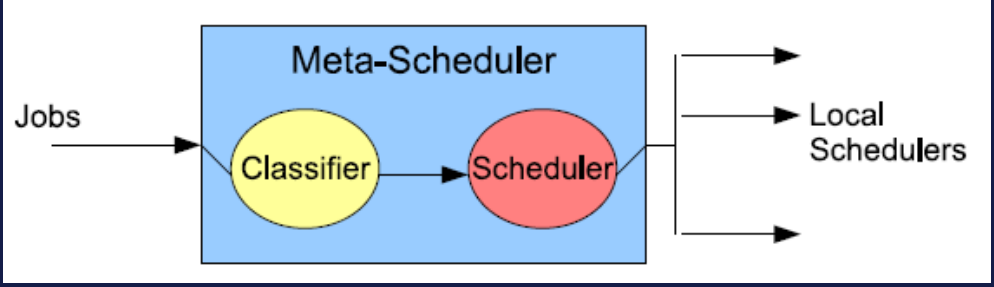
- Optimizing the utilization of hardware resources as well as software resources.

  Machines, Licenses.

Meta-Scheduler Objectives

- Definition of new jobs classification mechanism, in order to improve the quality of the MS scheduling decisions.

- Study new scheduling policies able to balancing the workload among clusters exploiting the carried out classification.



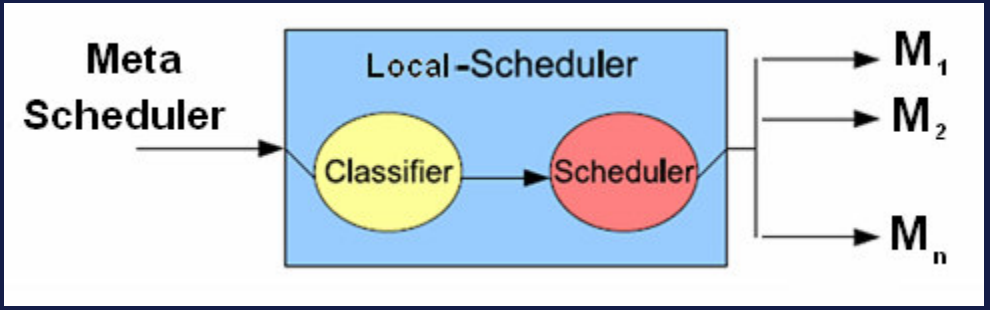
Local-Scheduler Objectives

- **Assigning jobs to the appropriate machine, respecting hardware and software constraints.**

# of CPUs requested by the jobs must not be greater than # of machine CPUs

The number of active licenses must not be greater than their availability.

- **Maximizing the resource usage, and the number of jobs executed respecting their QoS requirements**



Classification heuristics

	Meta-Scheduler (MS)	Local-Scheduler (LS)
Deadline	●	●
License	●	●
User	●	
Anti Aging		●
Wait Minimization		●

MS Classification Heuristics

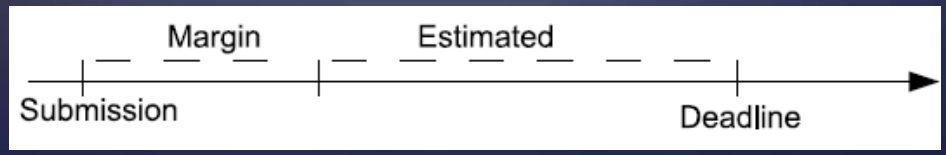
Users → We defined three classes of users in order to execute jobs respecting the user peculiarities: *Gold*, *Silver*, and *Regular*, to which are assigned decreasing priority values.

License → It aims to favor the execution of jobs that improve the contention on the sw licenses usage. It is computed as a function of the number of sw licenses required by a job.

Deadline → Jobs closer to their deadline get a boost in preference that gives them an advantage in scheduling.

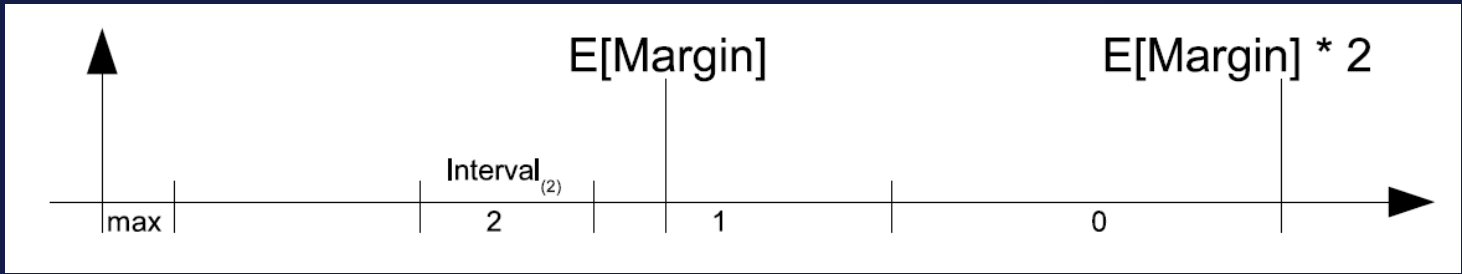
MS Deadline Heuristics [1]

Job deadline →



Heuristics main concept →

$$E[Margin] = \frac{\sum_{i=1}^{|N|} Margin_i}{N}.$$



$$\begin{cases} S_0 = 0 \\ S_k = S_{k-1} + MinUnity \cdot 2^k \end{cases}$$

$$MinUnity = \frac{2 * E[Margin]}{\sum_{k=1}^{max} 2^k}$$

MS Scheduling Heuristics

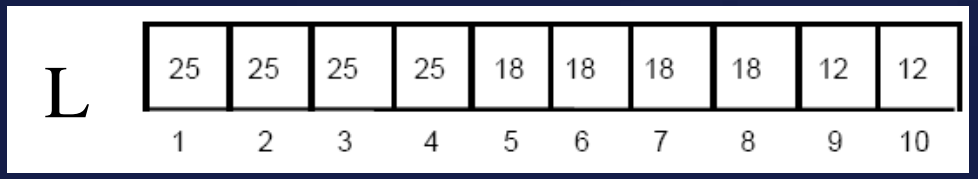
Load → Load aims to balance the workload among clusters by assigning a job to the less loaded cluster. The workload on a cluster is estimated summing the load due to the jobs queued to it.

Ordering → Ordering aims to balance the number of jobs with equal priority in each cluster queue.

MS Scheduling Heuristics [2]

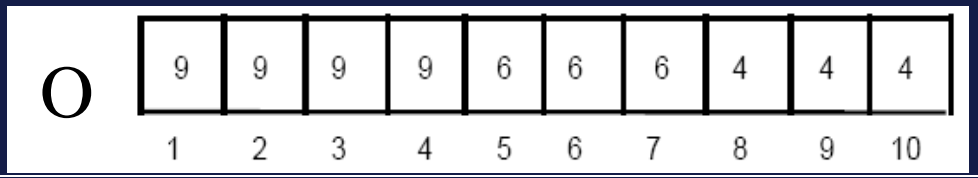
Load

The best cluster to assign a job is the idle one or the one with the minimum load, due to the jobs with priority equal to or greater than the priority of the job currently analyzed.

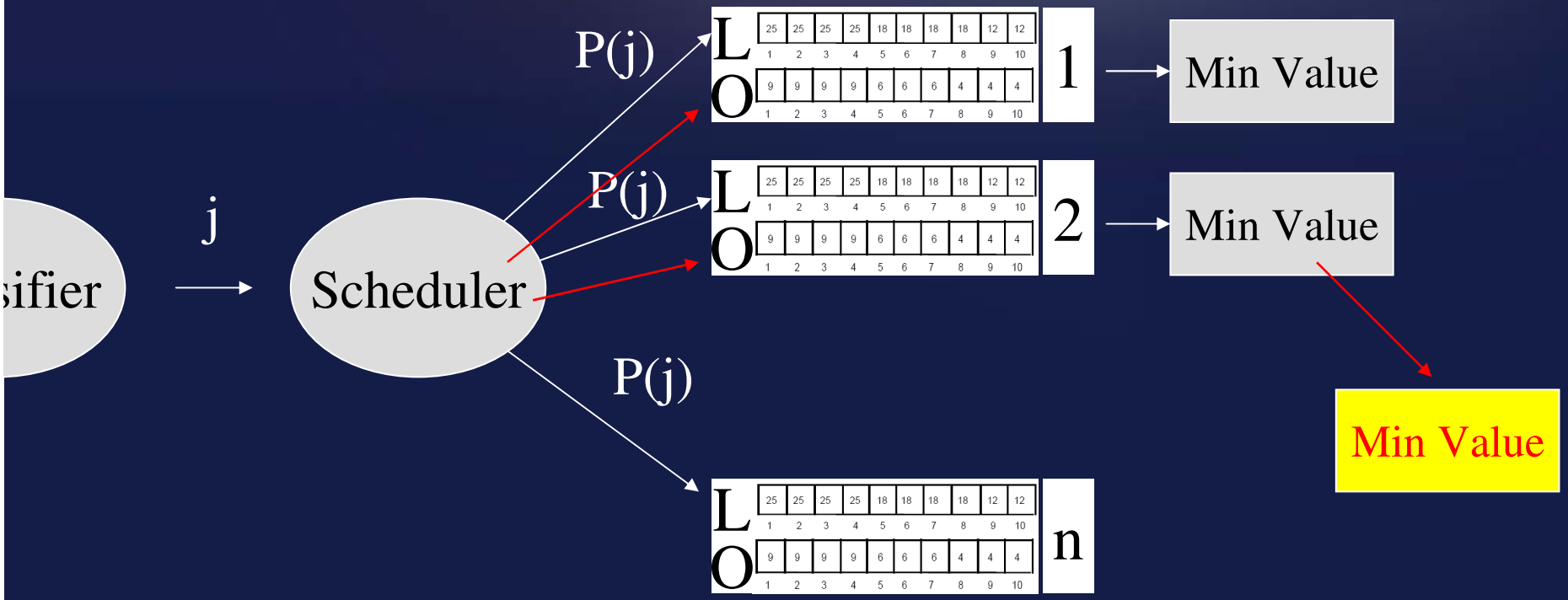


Ordering

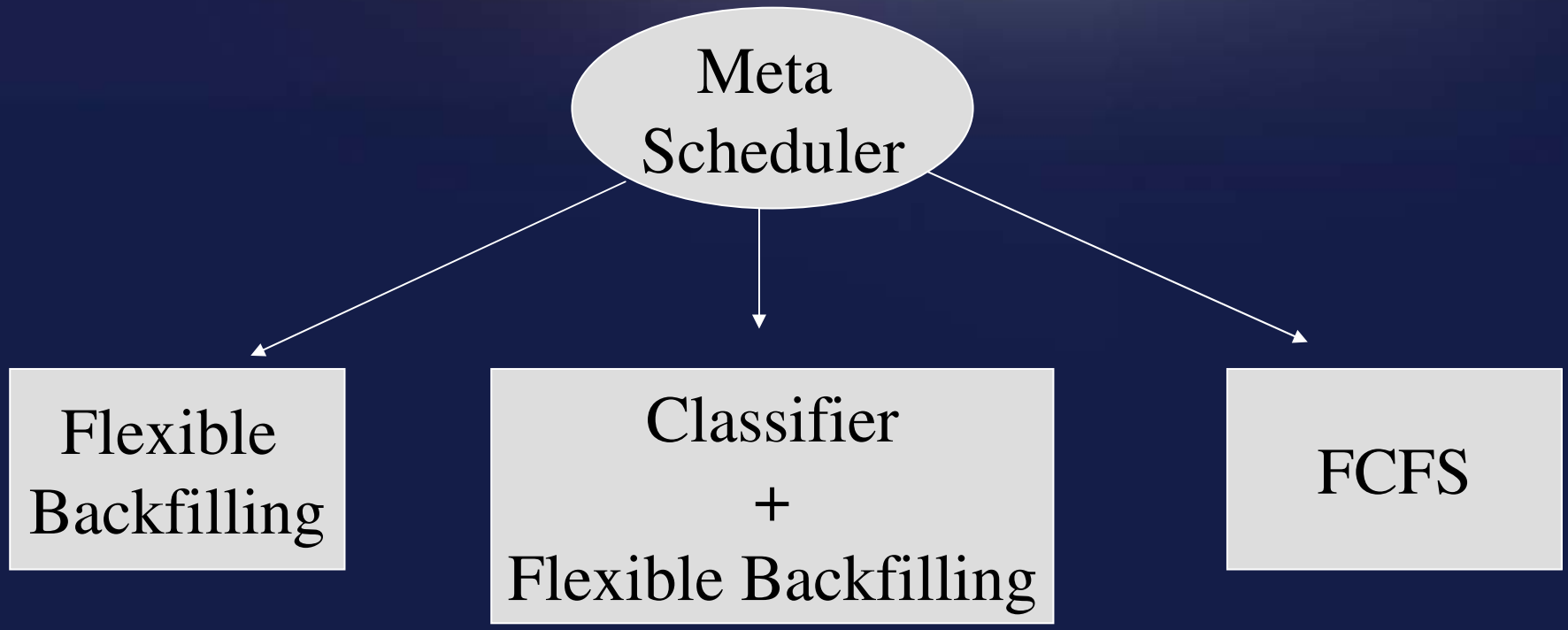
The best cluster to assign a job is the one with the minimum number of queued jobs, with priority equal to the priority of the job currently analyzed.



MS Scheduling Heuristics Composition



Experimental Environment



Experimental Environment [2]

Adopted Metrics:

- **Percentage of workload elaborated by each cluster**. It shows in which way the MS scheduling policies work.
- **Percentage of jobs that do not respect their deadline**. It shows the ability of the scheduler to execute jobs respecting their QoS requirements.
- **Average *Slowdown* of jobs without deadline**. It shows how the system load delays the execution of such jobs.
- **Percentage of system and sw license usage**. It shows as the job classification and scheduling solutions adopted both at MS and LS level allow a fruitful exploitation of the available resoruces.

Experimental Environment [3]

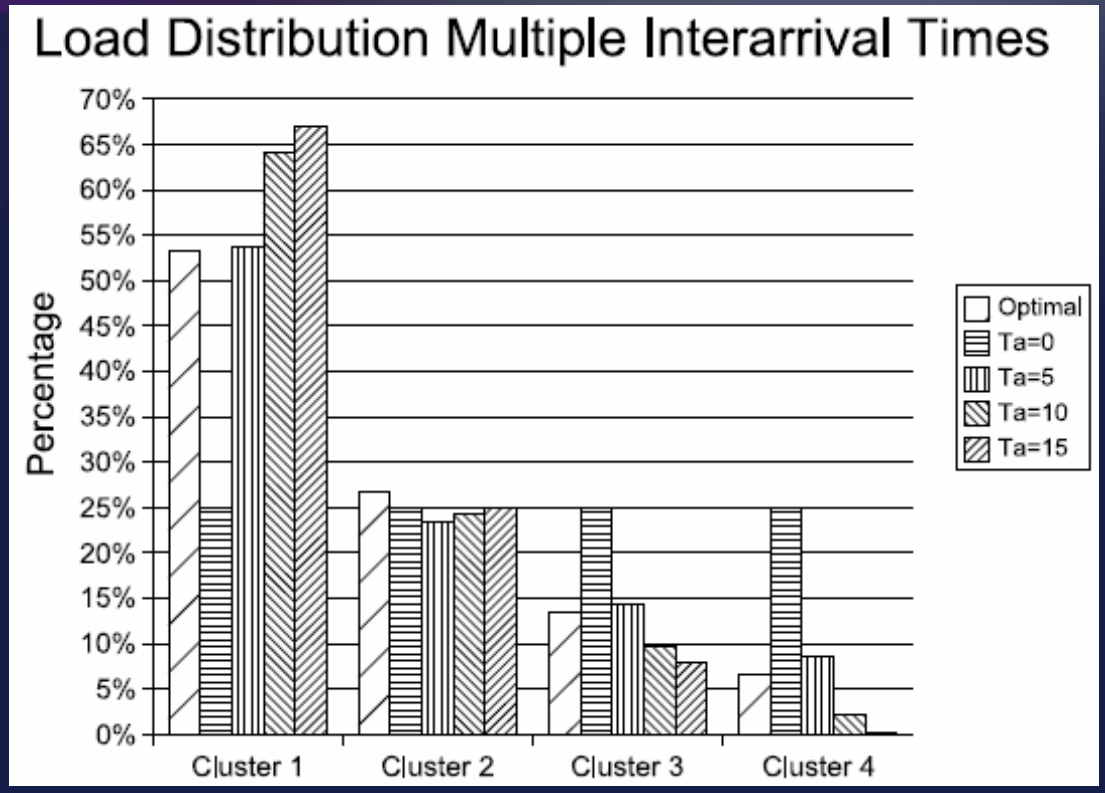
Simulation:

The evaluation was conducted by using a stream of 5000 jobs, 20 sw licenses, and a grid composed by 225 machines, distributed on four different clusters, each one including 120, 60, 30, and 15, respectively

We employed an event-based simulator, using four streams of jobs with jobs interarrival times fixed equal to 0, 5, 10, and 15 simulator time unit.

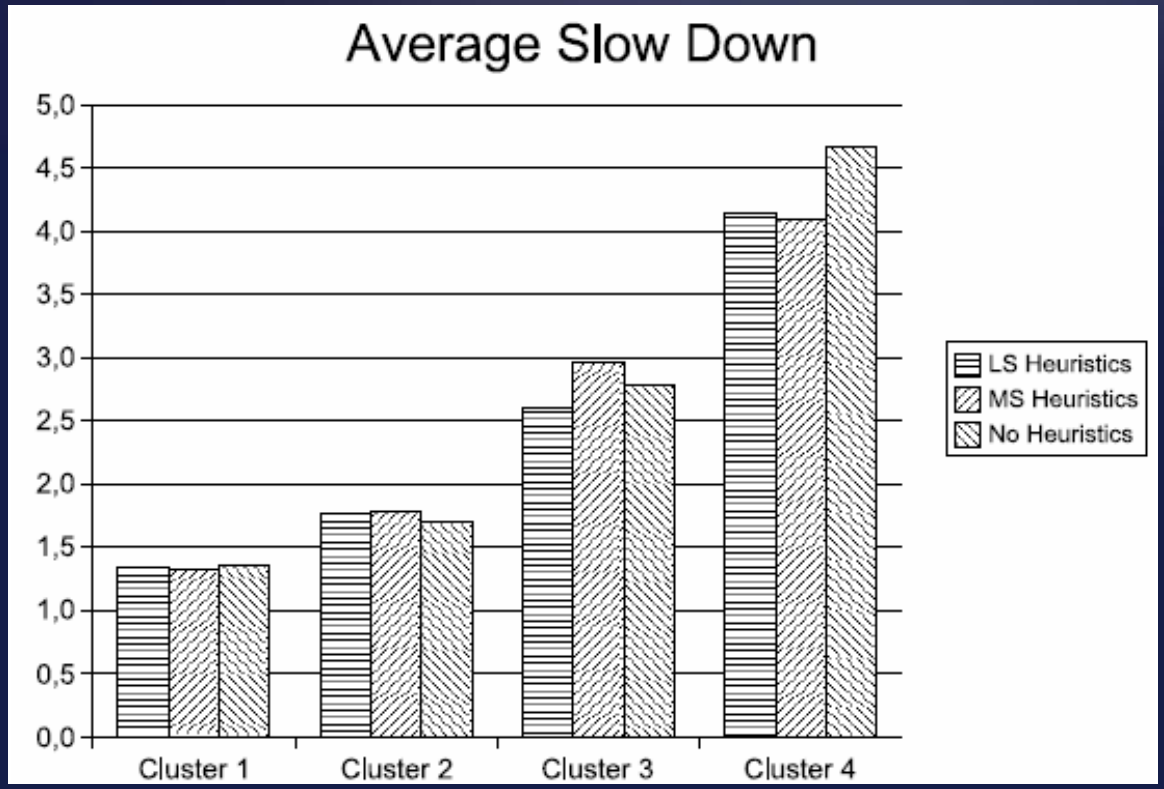
Parameters	Jobs	Machines	Sw Licences
Estimated	8000-10000		
Benchmark	100-500	100-500	
Margin	1500-5500		
CPU	1-8	4-32	
Estimated	8000-10000		
License needs	30%		
License ratio			50%-70%

Experiments

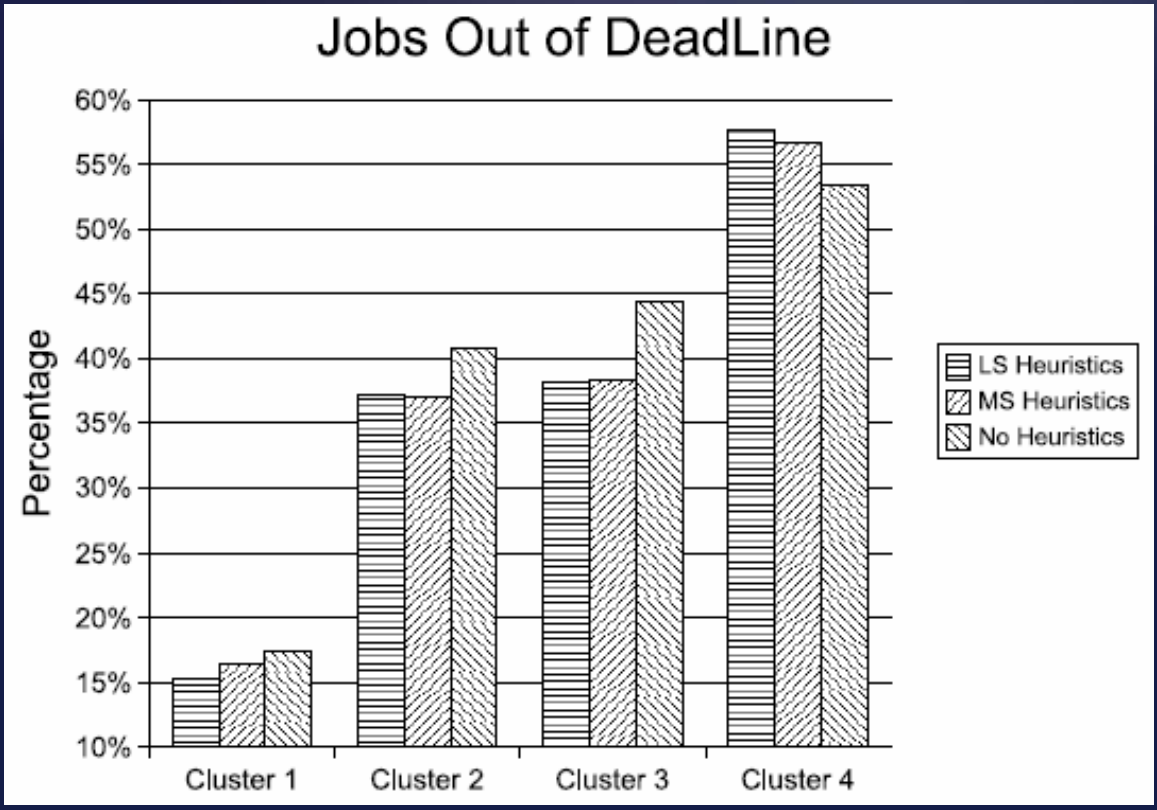


$$optimal\ cluster_i = \frac{\#\ of\ machines \in cluster_i}{\sum machine_k}$$

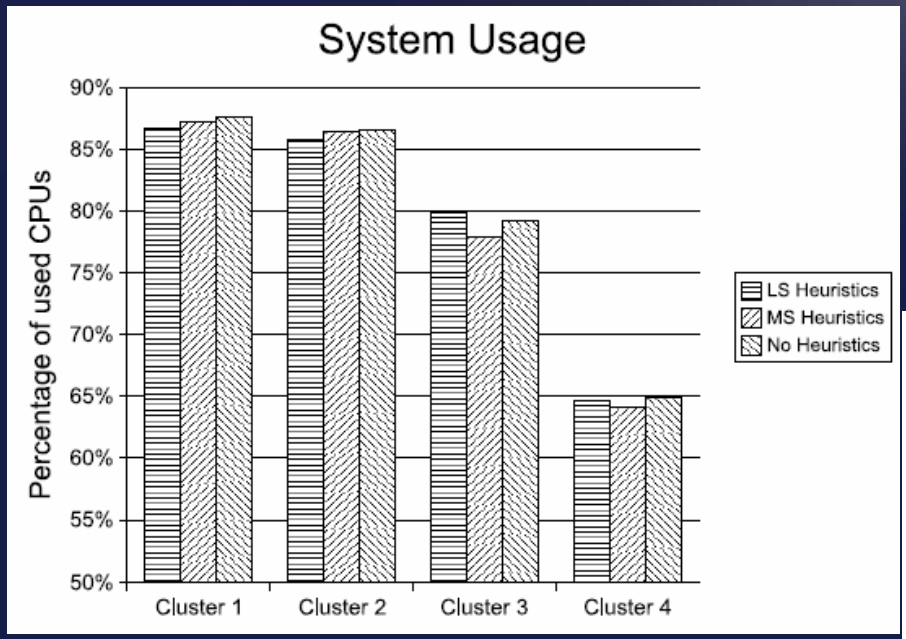
Experiments [2]



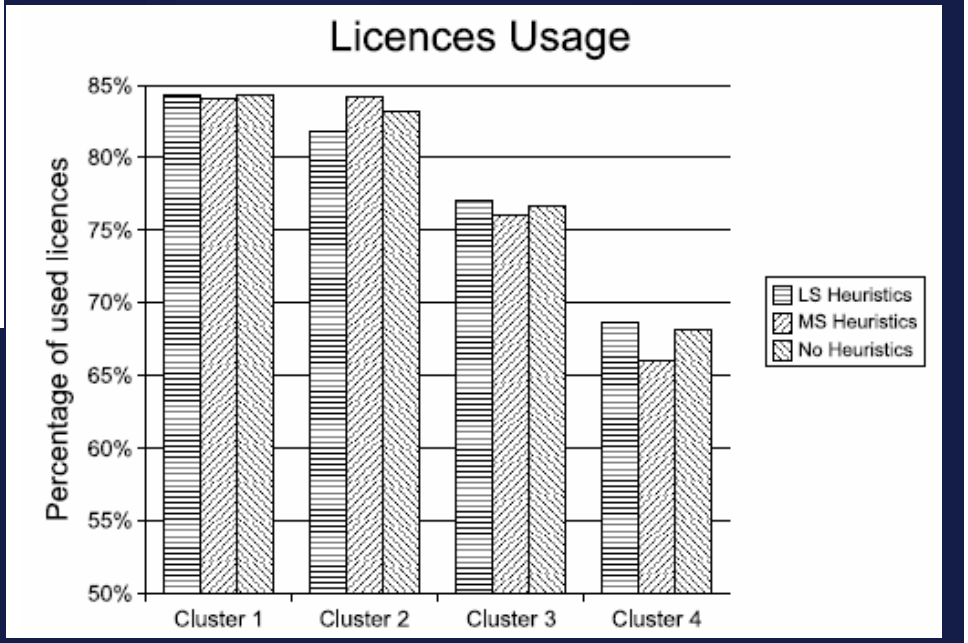
Experiments [3]



Experiments [4]



$$\frac{\#\ of\ active\ res}{min(\#\ of\ available\ res, \#\ of\ res\ requested\ by\ jobs)}$$



Conclusions and Future Works

Conclusions:

- We design and evaluate a two level scheduler able to classify incoming jobs with respect to their QoS requirements.
- Our Meta-Scheduler is able to dispatch jobs among the clusters of underling level balancing the workload among them with respect to their estimated computational power

Future Works:

- We plan to integrate our Mete-Scheduler with our Convergent Scheduler.
- We plan to evaluate or scheduling framework using the GridSim simulator.