European Grid Infrastructure
Towards a sustainable grid infrastructure

Increasing cluster computing performance through a dynamic load rearrangement

Federico Calzolari federico.calzolari@sns.it
Scuola Normale Superiore

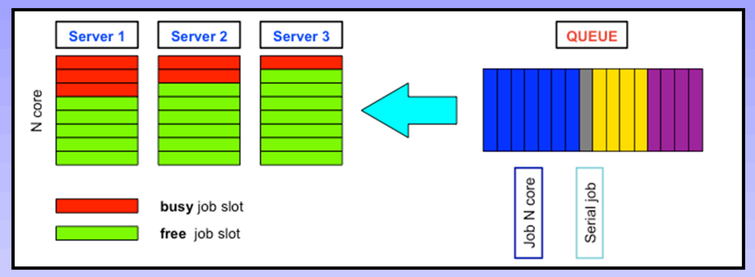
Silvia Volpe
University of Pisa, Italy

Abstract

One of the main problems for a computing center is given by the under exploitation of the available resources. It may happens that in a batch queue system, available for both serial and parallel jobs, one or more computational nodes are serving a number of jobs lower than their actual capability. A job displacement, executed at runtime is able to free extra resources able to host new processes.

A stuck condition

A typical stuck condition is represented by more single core jobs running each one over a multi core server, while more parallel jobs - requiring all the available cores of a host - are queued.



Batch queue system features and issues

- The batch queue system cannot modify the queued jobs order, except for brief interval in case of preemption.
- The scheduler has to respect fair share and job priorities.
- The batch queue system cannot move jobs at runtime.

Possible solutions

Cluster partition

The classical and probably the most used solution is the cluster partition: the computing farm is divided into independent blocks, in order to run in separate environments and over separate sets of nodes the serial jobs (requiring a single core) and the parallel jobs (multi-core).
This solution entails the loss of all the benefits coming from the sharing of the computing resources, and the under-exploitation of the computing farm in case of one of the two partition is not fully used.

Job rearrangement

In order to solve this problem, we suggest a new strategy: a Job mover able to displace and rearrange the jobs over the farm at runtime.

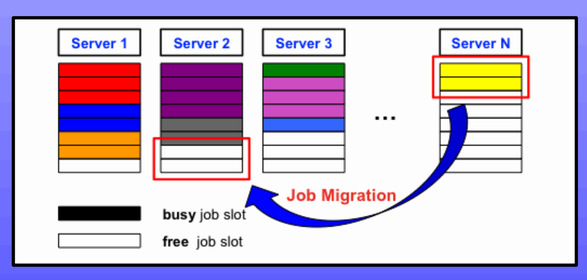
Problem complexity

The problem of the possible permutations achieved by moving a set of jobs, each one requiring a variable number of core, over a set of servers, is described by an NP-complete complexity class.

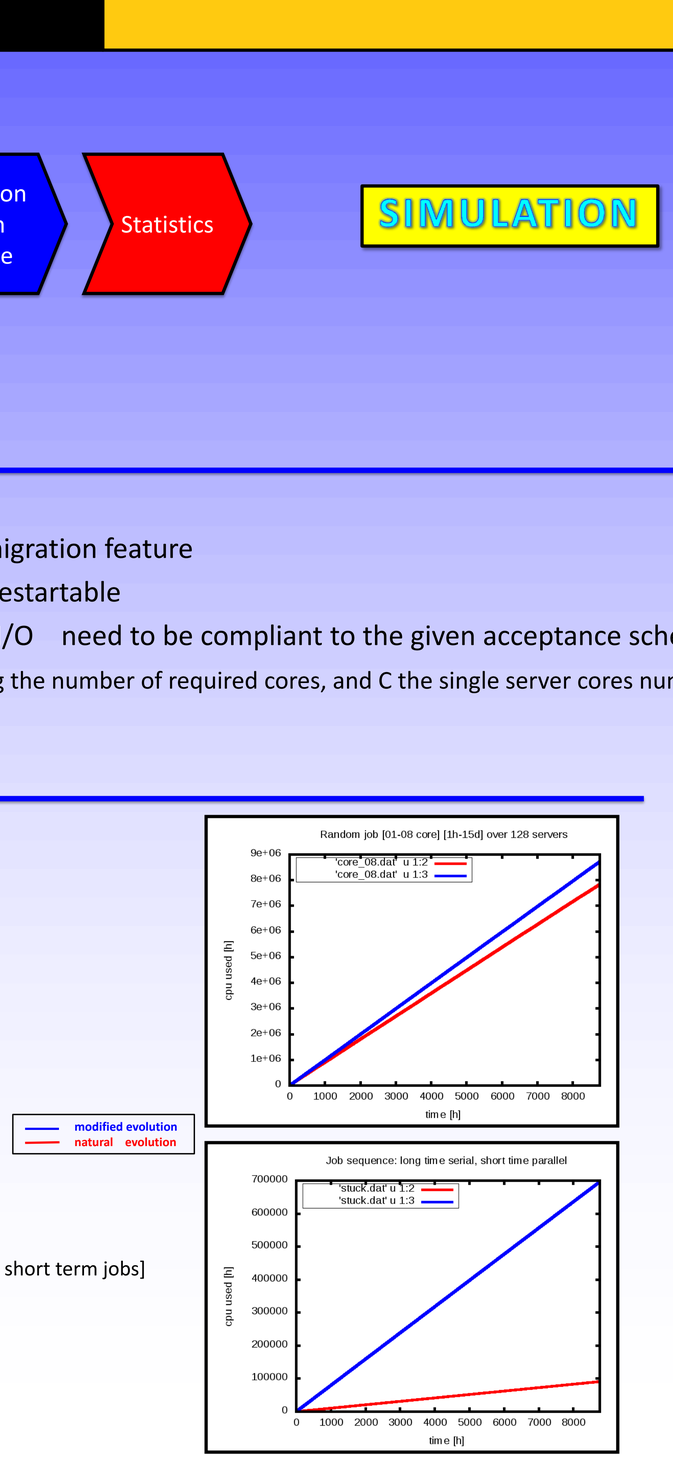
Due to the difficulty in finding the best solution, we focused on searching for a solution able to improve the current load status of the cluster, certainly not the optimum one.

The **Job mover**

The idea is to pile up the maximum number of jobs over the minimum number of hosts, compatibly with the available CPU and memory on the single hosts, in order to fill as many contiguous job slots as possible.
A job displacement, executed at runtime in order to stack up more single core processes over a single multi core server, is able to free extra resources able to host new processes - both single or multi core.



In order to ensure a full hosts exploitation, and prevent at the same time the overload of one or more nodes in the cluster, the job migration takes place only under certain conditions. We also paid a special attention to avoid a too frequent job displacement, damaging the global performance.



Requirements

- Batch queue system needs to provide the job migration feature
- Jobs have to be checkpointable, independent, restartable
- Jobs requirements in terms of CPU, RAM, disk, I/O need to be compliant to the given acceptance schema

  N core, N/C % {RAM, disk, I/O} N being the number of required cores, and C the single server cores number

Tests and results

Random job sequence

Jobs distribution:

- Cores number 1 to 8
- Running time 1 hour to 15 days
- Queue filling random
- Farm: 128 servers 8 core - 1 year of data acquisition

- Efficiency improvement = **12 %**
- Job moved / total = 4717 / 10726 [0.439]

Stuck condition

Jobs distribution:

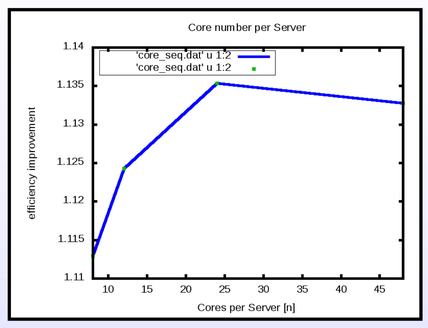
- Repeated sequence of: [serial long term jobs, followed by parallel short term jobs]
- in a 10 servers, 8 core farm - 1 year of data acquisition

- Efficiency improvement = **800 %**
- Job moved / total = 2239 / 17156 [0.130]

Algorithm efficiency

with respect to Cores number per server

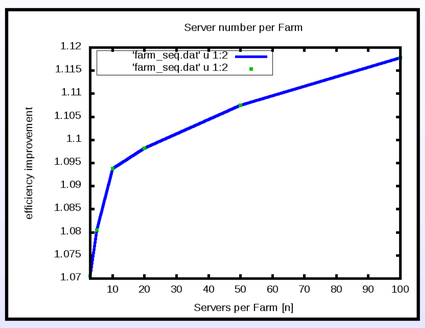
- **Server with 8, 12, 24, 48 cores**



- Efficiency improvement = **[11 - 13] %**

with respect to Servers number per farm

- **Farm with 3, 5, 10, 20, 50, 100 servers**



- Efficiency improvement = **[7 - 12] %**
- Job moved / total = [10 - 50] %

  depending on the farm size

The farm efficiency increases with the increasing of the servers number

The Algorithm

Rearrange jobs algorithm:

- reverse sort the servers by busy job slots;
- fill the most full server with jobs coming from the most free server.

Green computing

A secondary effect, probably not less appealing by the point of view of the "green computing", is represented by the power efficiency improvement through a dynamic job rearrangement, with an energy saving up to 90% in some particular cases - more frequent than expected.
By the use of a remote controlled power supply, it is possible to switch off the unused hosts, waiting to be switched-on at request.

Conclusions

A job displacement, executed at runtime in order to stack up the maximum processes number over single multi core servers, is able to free extra resources - and consequently host new processes in the computing farm.

The system, developed at Scuola Normale Superiore, in collaboration with the Information Engineering Department of the University of Pisa [Italy], is able to provide an increase in the number of running jobs.

The efficiency increase is from 7 to 13% in typical use cases, arriving up to 8 times more in some particular situations.

Any other idea with respect to new algorithms is welcome.