IL NUOVO CIMENTO Vol. 32 C, N. 2 Marzo-Aprile 2009
DOI 10.1393/ncc/i2009-10402-0

Colloquia: CSFI 2008

Development of a tool to optimize the performance of a Maui Cluster Scheduler

F. Cantini(1), M. Mariotti(2), L. Servoli(1) and C. Tanci(2)

(1) *INFN, Sezione di Perugia - Perugia, Italy*
(2) *Dipartimento di Fisica, Università di Perugia - Perugia, Italy*

(ricevuto il 22 Giugno 2009; pubblicato online il 16 Settembre 2009)

Summary. — The use of Linux cluster computing in a scientific and heterogeneous environment has been growing very fast in the past years. The often conflicting user's requests of shared resources are quite difficult to satisfy for the administrators, and, usually, lower the overall system efficiency. In this scenario a new tool to study and optimize the Maui Cluster Scheduler has been developed together with a new set of metrics to evaluate any given configuration. The main idea is to use the Maui internal simulator, fed by workloads produced either by a real cluster than by an *ad hoc* one, to test several scheduler configurations and then, using a genetic algorithm, to choose the best solution. In this work the architecture of the proposed tool is described together with the first results.

PACS 07.05.Bx – Computer systems: hardware, operating systems, computer languages, and utilities.
PACS 07.05.Kf – Data analysis: algorithms and implementation; data management.

1. – Introduction

In our Physics Department we have a Batch System based on a GNU/Linux cluster (about 200 CPUs) and Torque/Maui as Resource Manager/Scheduler [1]. Our farm has some non-standard characteristics:

– It is a GRID site connected with the EGEE / WLCG European infrastructure [2].

– It serves both GRID jobs and local jobs.

– It has been built with contributions of different local groups (working on different physics experiments).

– It uses extensively virtualization technologies, namely Xen [3].

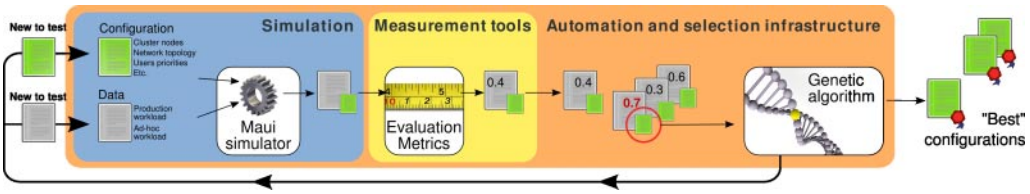

Fig. 1. – Description of the data flow.

In this scenario the scheduling policies are a key point of the whole infrastructure: there are many needs to be satisfied and often requirements are not compatible one another. The number of variables that influence the scheduling behaviour is so high that the optimization is a hard task. There are already some algorithms that tackle this problem [4], among which the *fair-sharing* one. However they are not completely satisfying and the need to compare different scheduling policies, especially when managing a heterogeneous cluster and/or when dealing with different groups that change their resource usage patterns, is felt by the administrators of a computing cluster.

This work is aimed to build an infrastructure capable of assessing, via a genetic algorithm, the goodness of different scheduling policies. In fig. 1 the logical flow of the software structure is shown:

– Maui trace data coming either from the real cluster or from a test cluster where they are produced under controlled conditions are stored into a repository;

– an automatic procedure based on a genetic algorithm is used to create the scheduler configuration to be fed into the simulator to process the data;

– Maui internal simulator (fed on workload from the real cluster or an ad hoc one) is used to test a scheduler configurations either on real or simulated data;

– the goodness of each configuration is evaluated using measurement tools based on specifically developed metrics;

– the output is then evaluated using the metrics and the results are sent to the genetic algorithm to start a new cycle.

– When the genetic algorithm converges on a solution, the process is stopped and the best configuration is extracted.

2. – Definition and implementation of the infrastructure

An extended infrastructure has been built (fig. 2) starting from the real cluster to allow a centralized and efficient management of all the steps and to automate the cycle.

The core of this structure is the workload repository, where are stored all the data coming from the real cluster, all the data coming from the test machines, all the MAUI configuration files used in the analysis.

There is also a Virtual Farm, a set of 4+1 virtual machines hosting the TORQUE + MAUI system. This facility has been used to generate workloads according to controlled patterns and conditions.

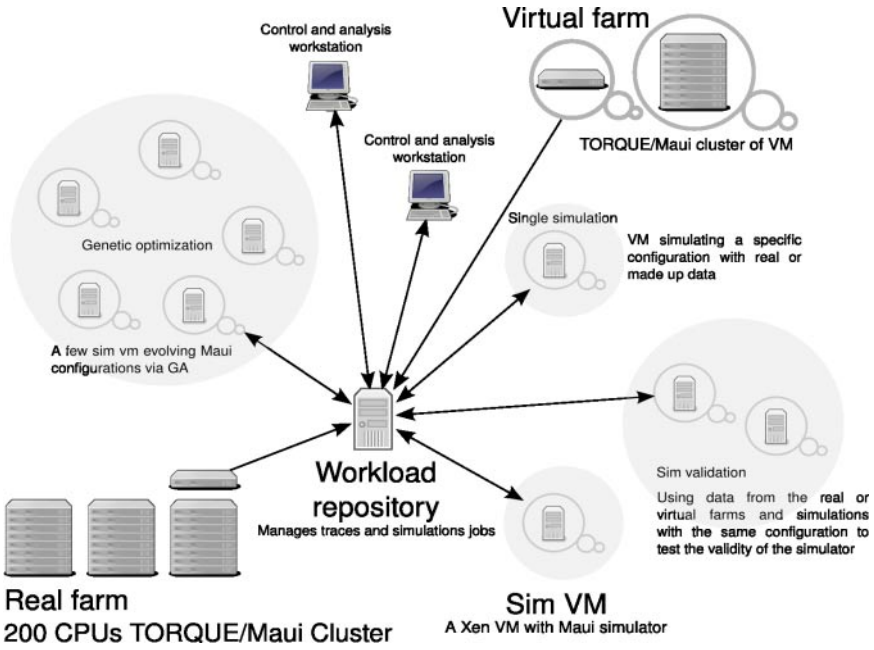

Fig. 2. – Architecture of the infrastructure to automate the data treatment, with all the relevant data flows.

A few others virtual machines have been used as simulation nodes to run the genetic optimization code, simulate all the combinations of a few configurations variables or for individual simulations.

On the software side a set of tools was developed to manage pre and post-simulation steps, to automatically generate cluster and use-cases configurations, to arrange for batches of simulations and real scheduling cases, to retrieve and submit simulations jobs and finally to analyze the results.

3. – MAUI Simulator's validation

The Maui simulator [5] is a special run mode of Maui developed to reproduce the behaviour of a real MAUI scheduler starting from configuration and workload data. It has been installed on a Virtual Machine and a first phase of tests has been carried on, using a virtualized test batch system, in order to identify its peculiarities and limits. Figure 3 (red lines refer to real data and green ones to simulated data) illustrate the comparison among the real behaviour in the test system and the simulated one. It can be appreciated that the simulation does not reproduce the time overhead due to the hardware delays in the real system. This difference can be easily neglected because we are not interested in the exact time ordering of the jobs and the time shift due to time quantization is of the order of 1 s, *i.e.* negligible with respect to the duration of normal batch jobs.

In fig. 4 a node swap between real and simulated data is also illustrated; again this effect is not relevant, because we are not really interested in which of identical nodes has been assigned to a certain job.

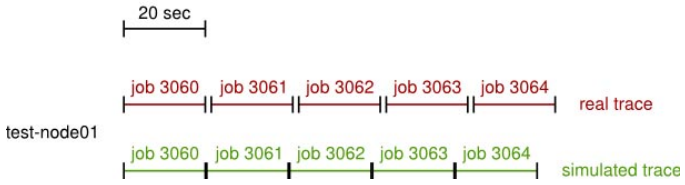

Fig. 3. – (Colour on-line) Comparison among real (red) and simulated (green) data. A time overhead in the real system, due to hardware delays, is observed, together with a difference caused by the time quantization in the two systems.

4. – Evaluation metrics

In order to evaluate each configuration and to compare its effects a series of specific metrics have been defined, tailored to the specific needs of our use-case:

– *Fairness metric:* Every user and group is entitled to use its own share of hardware resources; fairness measures how much the system succeeds in ensuring this goal. First the *single job fairness* is defined as eq. (1) where J_{queued} is the number of queued jobs for that user, J_{running} is the number of running jobs and R_{granted} is the number of granted resources. Then the global fairness for a certain user over a defined time interval is computed, where ΔT_i is the length of each interval in which the time domain is divided in the workload database.

$$(1) \qquad F_\epsilon = 1 - \frac{\min(J_{\text{queued}}, R_{\text{granted}} - J_{\text{running}})}{R_{\text{granted}}}; \qquad F_{\text{global}} = \frac{\sum_{i=1}^{n} F_{\epsilon i} \Delta T_i}{\sum_{i=1}^{n} \Delta T_i}.$$

– *Efficiency metric:* When a job is queued, its waiting is justified only if there are no free resources. Efficiency quantifies if the waiting time is justified; first the *job efficiency* is defined according to eq. (2):

$$(2) \quad E_{\text{job}} = 1 \quad \text{if the waiting time is 0;} \quad E_{\text{job}} = \frac{\sum_{i=1}^{m} E_{\epsilon i} \Delta T_i}{\sum_{i=1}^{m} \Delta T_i} \quad \text{in all other cases.}$$

where $E_{\epsilon i} = N_{\text{job}}/N_{\text{resources}}$ is the ratio between the number of jobs running in the system in a given time interval ΔT_i and the total of the available resources.

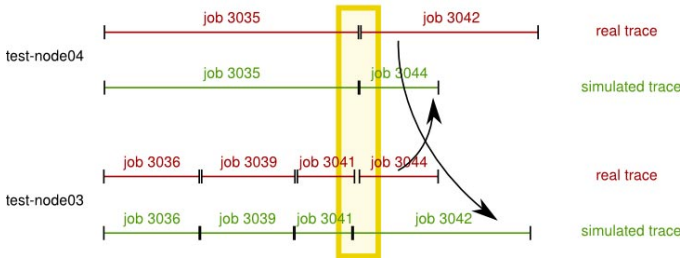

Fig. 4. – Description of the node swap mechanism; one job executed in the real cluster on one node, in the simulated one is assigned to another node.

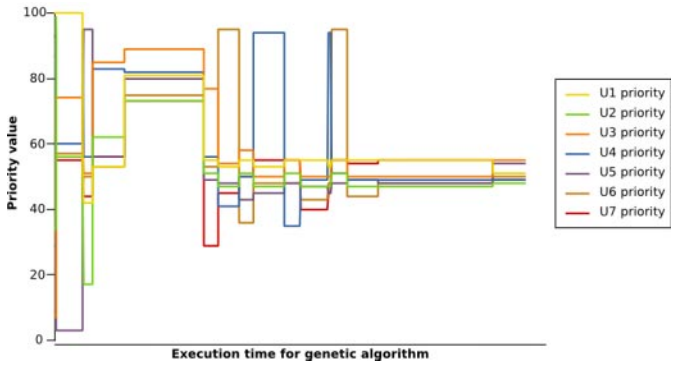

Fig. 5. – User priorities evolution *vs.* execution time for genetic algorithm relative to maximum fitness.

In eq. (3) the total efficiency for a single user over a given time interval is then defined:

$$E_{\text{total}} = \frac{\sum_{i=1}^{n} E_{jobi} \Delta Q_i}{\sum_{i=1}^{n} \Delta Q_i}. \tag{3}$$

– *Fitness metric:* This metric is used as a guide for the convergence of the genetic algorithm, and measures the degree to which the algorithm achieves the goal to minimize the difference in the average queue time between jobs of different users. It takes values in the range 0–1, with 1 for a null difference and a perfect result.

5. – Early analysis: Simple evolution of user priorities

A simple test case was selected to verify that both simulation framework and genetic algorithm work properly: a batch system of 4 nodes, 1 processor per node with 8 users, each submitting the same job sequence has been simulated. The priority for the first user was fixed at a value of 50, while the others were free to evolve in the range of 1–100; a genetic algorithm to minimize the difference in the average queue time between all the users has been implemented. The expected best configuration was the one with the evolving users priorities all matching the value of 50. 20 generations each with a population of 100 were evolved and a stable configuration was found (fig. 5) matching the expected results.

6. – Conclusions

A flexible and robust infrastructure has been put in place to allow to study the MAUI scheduler performance in a computing cluster. The MAUI simulator has been validated and a genetic algorithm optimization procedure has been implemented and tested on simple use-cases using a Virtual Computing Farm. Several specific metrics have also been defined in order to characterize quantitatively the influence of each different MAUI configuration on a given schedule and cluster configuration. A refinement of the metrics is in progress, to quantify different aspects of quality of service concerning the batch system, together with a further validation of MAUI simulator using Monte Carlo methods. The

optimization of the real cluster will then be carried out together with the monitoring in quasi real-time of the system performances.

* * *

The authors would like to thank the Physics Department of Università degli Studi di Perugia for the use of the Computer Science Laboratory.

REFERENCES

[1] Cluster Resources, Inc., *Maui Administrator's Guide* (2008).
[2] Bird I., Bos K., Brook N., Duellmann D., Eck C., Fisk I., Foster D., Gibbard B., Girone M., Grandi C. *et al.*, *LHC Computing Grid Technical Design Report* (2005).
[3] Barham P., Dragovic B., Fraser K., Hand S., Harris T., Ho A., Neugebauer R., Pratt I. and Warfield A., *Xen and the art of virtualization*, in *ACM SIGOPS Operating Systems Review*, Vol. **37** 5 (ACM, New York, NY, USA) 2003, pp. 164-177. 10.1145/1165389.945462.
[4] Jackson D. B., Snell Q. and Clement M. J., *Core Algorithms of the Maui Scheduler*, in *Lecture Notes In Computer Science*, Vol. **2221** (Springer-Verlag London, UK) 2001, pp. 87-102. 10.1007/3-540-45540-X.
[5] Cantini F., *Studio delle prestazioni di un batch system basato su TORQUE/Maui*, Tesi di Laurea, Università degli Studi di Perugia (2007) `http://www.infn.it/thesis/thesis_dettaglio.php?tid=2225`.