Convergent Scheduler: a Job Scheduling Framework for Heterogeneous Computing Farms

G. Capannini*, M. Pasquali+,*, R. Baraglia*

* ISTI - Institute of the Italian National Research Council, Pisa, Italy
+ IMT - Lucca Institute for Advanced Studies, Lucca, Italy
email{gabriele.capannini, marco.pasquali, ranieri.baraglia}@isti.cnr.it.

Abstract

Computing farms provide the ability to harness the power of a set of computational resources as an ensemble. A computing farm can integrate heterogeneous hw/sw resources such as workstations, parallel systems, servers, storage arrays, and software licenses. Such infrastructures require appropriate schedule mechanisms in order to satisfy the application performance requirements (QoS) and to optimize the resource usage. In such an environment, users should submit their computational requests without necessarily knowing on which computational resources these will be executed.

We propose a new framework, called *Convergent Scheduler*, for scheduling a continuous stream of batch jobs on the machines of large-scale computing farms. This method exploits a set of heuristics that guide the scheduler in making decisions. Each heuristics manages a specific problem constraint, and contributes to carry out a priority value that measures the degree of a matching between a job and a machine. Jobs priorities are re-computed in the case of selected system events (i.e. arrival of a new job or end of a job execution).

In order to exploit the convergent scheduling method, we define a matrix, called *job-machine matrix*. Each entry stores a priority value specifying the degree of preference of a job for a machine. Our scheduling framework is structured according to three main phases: *Heuristics*, *Clustering & Pre-Matching*, and *Matching*.

The first phase, Heuristics, changes priority values in the job-machine matrix in order to increase/reduce jobs-machines matching degrees. Seven heuristics have been implemented to build our Convergent Scheduling framework: *Minimal requirements*, this heuristics fixes the associations job-machines selecting the set of machines that has the computational requirements suitable to perform a job. *Deadline*, which aim is to compute the job-machine update value in order to execute jobs respecting their deadline. *Licenses*, this heuristics updates the job priorities to favor the execution of jobs that increase the critical degree of licenses (a license becomes critical when there is a number of requests greater than the available number of its copies). *Input*, the goal of this heuristics is to update the matrix entries according to the cost due to the transfer of the job input data on the machines candidate to run it. *Wait minimization*, which aim is to minimize the average time that jobs spend waiting to complete their execution. *Overhead minimization*, which aims is to contain the overhead due to the interruption of a job execution on a machine and its resuming on another one, or on the same one at a different time. *Anti-aging*, which goal is to avoid the jobs starvation.

The second phase, Clustering & Pre-Matching, is executed after the job-machine matrix has been updated by all the heuristics. All jobs requesting critical sw licenses are clustered by putting in the same cluster the jobs asking for the same license/s, and with each job belonging to only one cluster. A new event (job submission or ending) can change the license usage. The MKP (Multidimensional 0-1 Knapsack Problem) optimization method is applied to each cluster to find the subset of jobs that could be simultaneously executed, without violating the constraint on the licenses usage. The remaining subset will be discarded, i.e. their entries are deleted from the job-machine matrix. The resulting job-machine matrix will be passed to the Matching phase to carry out the job-machine associations.

The third phase, Matching, carry out the best job-machine associations, i.e. the new job scheduling plan. Starting from the job-machine matrix, resulting from the previous phases, it looks for a matching which corresponds to the largest preference degree, according to the constraint that each machine must perform one job at time, and that a job can be scheduled only on one machine.

The Convergent Scheduling was evaluated by simulations using different streams of jobs with different inter-arrival times between jobs. Three different versions of the CS scheduler, which exploit different features, were evaluated comparing them with EDF and Backfilling schedulers. Even if the conducted evaluation constitutes the first step in analyzing the behavior and quality of the presented approach, it seems that the proposed solution is a viable one.

Acknowledgment This work has been supported by SUN Microsystems's grant, and it is in the activity of the European CoreGRID NoE (European Research Network on Foundations, Soft- ware Infrastructures and Applications for Large Scale, Distributed, GRID and Peer-to-Peer Technologies, contract no. IST-2002-004265).