QoS-constrained List Scheduling Heuristics for Parallel Applications on Grids*

Ranieri Baraglia, Renato Ferrini, Nicola Tonellotto
Information Science and Technologies Institute
Italian National Research Council
Via G. Moruzzi 1, Pisa, Italy

Laura Ricci
Deptartment of Computer Science
University of Pisa
Largo B. Pontecorvo 3, Pisa, Italy

Ramin Yahyapour
Institute for Robotics Research
University of Dortmund
Otto-Hahn-Straße 8, Dortmund, Germany

Abstract

This paper presents QLSE (QoS-constrained List Scheduling hEuristics), a Quality of Service-based launch time scheduling algorithm for wide area Grids. QLSE considers applications described by a Task Interaction Graph (TIG) whose nodes and edges are labeled according to the Quality of Service requirements of the application. The high values obtained in the performance evaluation for both the tasks communication and computation throughput demonstrates the applicability of the proposed approach.

1. Introduction

One of the most challenging problem of Grid computing is the selection of a suitable set of resources for the application of interest and the mapping of its tasks onto the selected resources. Given a graph describing the Grid, the mapping should assign Grid resources to the tasks in order to minimize a objective function. The mapping problem is known to be NP-complete, hence several heuristics have been proposed. The goal of these heuristics is to map a set of highly interacting tasks to a set of Grid resources connected by an acceptable bandwidth communication. For this reason, a suitable model of the Grid is needed as well. A Grid interconnection structure may include links with alternative performances, hence no constraint on the network topology can be assumed. This paper presents QLSE (QoS-constrained List Scheduling hEuristics), a Quality of Service-based launch time mapping algorithm for wide area networks. QLSE considers applications described by a Task Interaction Graph (TIG) whose nodes and edges are labeled according to the Quality of Service requirements of the application. To avoid the resource aging phenomena, QLSE maps at launch time an application by applying a set of fast heuristics exploiting dynamic information about the Grid status returned by typical Grid monitoring tools. QLSE detects subsets of LANs connected by threshold values of communication bandwidth by clustering the Grid graph. These thresholds are defined by a statistical analysis of the QoS of the communications required by the user. QLSE orders the tasks of the applications according to a priority which takes into account both their computational requirement and their bandwidth requirements. The clusters and the LANs within each cluster are ordered on the basis of their computational power and of their communication bandwidth as well. Then, a list based mapping algorithm is defined and evaluated.

*This work has been supported by the European CoreGRID NoE (European Research Network on Foundations, Software Infrastructures and Applications for Large Scale, Distributed, GRID and Peer-to-Peer Technologies, contract no. IST-2002-004265).

2. Related Work

The topic of scheduling in Grid systems has been subject of research for quite some time. The scheduling for Grids can be distinguished into resource-centric and application-oriented. The resource-centric approach focuses on the optimization of the performance of a resource or the whole Grid system, e.g. throughput, average response, or utilization [5]. The application-oriented approach focuses on the improvement for individual applications/jobs. The work in this paper is related to the application-oriented scheduling with bandwidth consideration. As a well-known example, the AppLeS approach addresses specifically the scheduling aspects on an application level [2]. There are also approaches for jobs that are workflows with task dependencies. These workflows are usually modeled by directed acyclic graphs (DAGs). A typical example is HEFT [7] or FCP [6]. Both approaches are list-based heuristics in which

0-7695-3089-3/08 $25.00 © 2008 IEEE
DOI 10.1109/PDP.2008.68

nodes in the DAG are prioritized and assigned to resources. Communication cost has also been considered in several approaches, e.g. by applying clustering heuristics [4]. There is a trend to higher-level programming of Grid systems by using components. That is, jobs are composed from several components whose computational and communication cost are modeled e.g. by a TIG[1]. The authors proposed in [1] a launch time heuristic to schedule component-based parallel applications on Grids in wide-area networks. The applications were also modeled as TIGs graphs and the Grids topology was derived from dendrograms. While the approach tackles the same scenario as in this paper, we are now considering a new approach for the dynamic clustering of LAN subsets with high bandwidth connectivity. The network topology of Grids in wide-area networks is typically not directly known. However, the are several tools to get information about available bandwidth between sites, like the Network Weather Service [8] or TopoMon [3].

3. Problem Description

QLSE assumes that a parallel application is composed by multiple tasks in continuous execution, and that the communications between tasks are bidirectional and may occur at any time during the application execution. It is also assumed that the tasks computation and communication costs are known before the application execution. This kind of application requires the co-allocation of parallel tasks before starting its execution.

A weighted Task Interaction Graph (TIG), denoted as $G_{App} = (N, E)$, is adopted to model a parallel application. In G_{App} a node $n_i \in N$, with $1 \leq i \leq |N|$, models a task, and a undirected edge $a_{ij} \in E$, with $i \neq j$, models a communication occurring between node n_i and node n_j. Each node has an associated weight representing the amount of computation required to execute the corresponding task, and edges have an associated weight representing the amount of data exchanged between two nodes. Application costs can be obtained either by static program analysis, by benchmarking the code on a reference system or statistically by analyzing historical data describing the application execution.

In Fig. 1 an example of a TIG is shown. In general, the practical evaluation of computation and communication requirements is not an easy task. To simplify the evaluation, we assume that jobs are uniform, so each one can be executed on every computational resource of the target Grid. The weight of a node is its *Minimum Computational Requirement* (MCR) and it represents the minimum computational power, expressed in Mflops, required to execute a node task. The weight of an edge is its *Minimum Bandwidth Requirement* (MBR) and it represents the minimum bandwidth required between the machines connecting the jobs at its two endpoints.

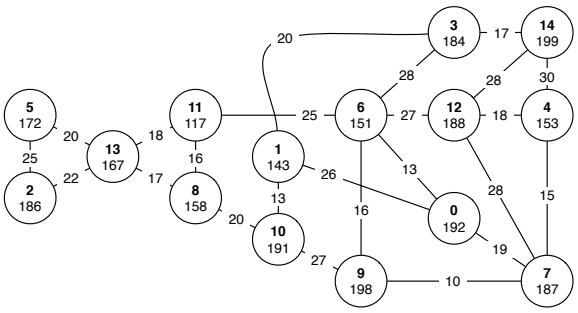

Figure 1. Example of a weighted Task Interaction Graph with 14 nodes

A Grid is modeled as a set of computational resources geographically dispersed and interconnected through an interconnection network, e.g. Internet. To model this network, we assume that each resource is connected to a single, well known Local Area Network (LAN). A LAN is shared between the interconnected resources (i.e. shared bus), and it is characterized through its nominal communication bandwidth, expressed in Mb/s. Several LANs are interconnected through an unreliable network whose topology is unknown. Exploiting network monitoring tools it is possible to estimate the bandwidth between two separate LANs. However, it is not viable to monitor every single communication path, and some paths can not exist or have a very limited bandwidth (e.g. smaller than 1Mb/s). Nevertheless, without loss of generality we assume the topology connected.

For each LAN, we assume to know the following information: the number of hosts connected to the LAN and their instantaneous computational bandwidth[1] and the inner communication bandwidth of the LAN. An example of a Grid model is shown in Fig. 2. Note that if there is not a direct connection between two LANs, we assume they can not communicate, even if there is a path between them composed by more than one edge. This assumption has a strong impact on the mapping of two communicating tasks. In fact, they are forced to be mapped on the same LAN or onto two directly connected LANs.

4. Algorithm Architecture

The proposed algorithm takes as input the weighted graph of a parallel application G_{app} and the weighted graph of the target Grid G_{grid}. The algorithm goal is to find a mapping of the application tasks to the Grid hosts in a way to fulfill simultaneously the minimum computational

[1]We suppose that each computational resource supports a multiprogramming environment.

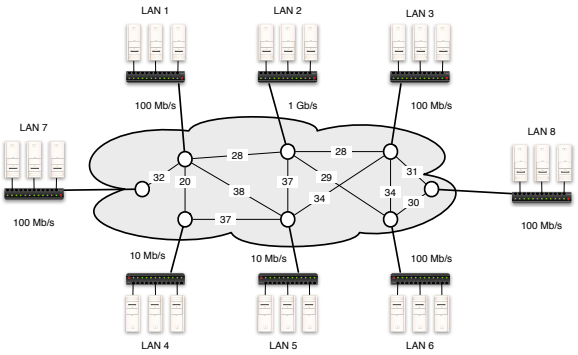

Figure 2. Example of a modeled Grid with bandwidth information

requirements of tasks and the minimum bandwidth requirements of communications.

The algorithm exploits a list scheduling heuristics which, by "appropriately" ordering the tasks, LANs and hosts, tries to meet the application QoS requirements by mapping communicating tasks on the same LAN or on directly connected LANs.

4.1. Tasks Ordering

Task are ordered according to a *score* assigned to each of them. This score takes into account the MCR of each task and it incorporates information on the topology of the application in order to exploit the high communication bandwidth of a LAN:

$$\forall n_i \in N, \quad score_i = MCR_i + \sum_{a_{ij} \in E} \left(\alpha_T MBR_{ij} + \beta_T MCR_j \right) \tag{1}$$

where α_T is a conversion factor and $\beta_T \in [0, 1]$ is a relative weight of the communicating tasks computational requirements. This means that the score of a node is the sum of its MCR, its complete communication requirement (the sum of MBRs) and a percentage of the MCRs of the nodes which it is communicating with. We performed several experiments to tune such parameters, and we obtained the best results with $\alpha_T = 1$ and $\beta_T = 0.25$.

The tasks are then structured in layers. The highest score task is placed in the first layer. The tasks directly connected to tasks in the previous layer are in the subsequent layer. The tasks in each layer are then grouped in subsets of communicating tasks; i.e. tasks directly connected in the TIG are put in the same subset. Then the subsets are ordered in decreasing order of number of tasks, and, inside a subset, tasks are ordered in descending order of score. The final task ordering is built considering the task in the first position, followed by the ordered tasks of the second layer and so on.

According to this ordering, we will try to allocate communicating tasks in the same LAN or in directly connected LANs. A problem may arise when dealing with cycles in the application graph. With the proposed ordering, we have no guarantee that the first and the last allocated tasks of a cycle will be allocated in the same LAN or in directly connected LANs, thus generating a mapping that could be unable to fulfill the communication requirements. To solve this issue, we force all the cycle's tasks but the first to be considered part of the same subset of tasks.

4.2. Hosts Ordering

Each host in a LAN share the same communication medium, so in a LAN the hosts can be ordered in decreasing order of computational power. The priority of a LAN is calculated similarly to (1):

$$\forall L_i \in N_{LAN}, \quad prio_i = \sum_{h_{ij} \in R_i} r_{ij} + \sum_{(ij) \in E_{LAN}} \left(\alpha_L bw_{ij} + \beta_L \sum_{h_{jk} \in R_j} r_{jk} \right) \tag{2}$$

where N_{LAN} is the set of the Grid LANs, R_i is the set of hosts connected to the LAN L_i, h_{ij} is the j^{th} host of the LAN L_i and r_{ij} is its computational power and bw_{ij} is the average bandwidth of the link between L_i and L_j.

Also in this case, α_L is a conversion factor and $\beta_L \in [0, 1]$ is a relative weight of the total power of the directly connected LANs. Again, we obtained the best results choosing $\alpha_L = 1$ and $\beta_L = 0.25$.

4.3. LAN Clustering

LAN to LAN interconnections are unreliable. To try to guarantee the required bandwidth performance during the execution of the application, we would like to exploit LAN to LAN interconnections with the highest communication bandwidths. To do so, we can, in first instance, ignore the low bandwidth links of the Grid. However we have no guarantees that this link removal will allow us to find feasible solutions. Anyway, we can proceed with some "tentative mappings", starting from a Grid graph with only the highest communicating edges. Then, if no solution is admissible, we add some other edges, and we continue until, at the end, we consider the original Grid graph.

After several tries, we have decided to use the *quartiles* of the links bandwidth distribution to set the values of the edges to be removed in each step.

In the first tentative mapping, we consider the $x_{1.00}$ quartile, hence removing every LAN to LAN interconnection from the Grid graph. In doing so, we are looking for a single LAN able to hosts the whole application. In the second tentative mapping, we remove from the Grid graph the

edges with bandwidth values smaller than the upper quartile $x_{.75}$. In doing so, we may obtain several disconnected LAN clusters. In such case, we order the clusters and we try to find a solution on a single cluster. If no cluster allowing an admissible value is found, we repeat the procedure with the middle quartile $x_{.50}$, then with the lower quartile $x_{.25}$ and eventually with the original Grid graph (in general we can assume $x_{.00} = 0$).

The clusters are ordered in decreasing order of their priority. The priority of a cluster is computed as the sum of the computational and communication bandwidths of the cluster LANs and the communication bandwidths between the LANs of the cluster.

4.4. Task Allocation

When a task and a LAN are selected, the allocation of the selected task on the selected LAN can be performed if the following *allocation conditions* are met:

- The LAN contains at least one host with computational power greater than or equal to the task MCR value.
- The sum of the MBR values of the edges between the selected tasks and the communicating ones already allocated on the selected LAN must be smaller than or equal to the LAN bandwidth.
- The sum of the MBR values of the edges between the selected tasks and the communicating ones already allocated on another LAN must be smaller than or equal to the bandwidth between the two LANs.

If such conditions are met, the task is allocated to the LAN and the computational power of the selected host is decreased by task's MCR value, the LAN bandwidth is decreased by the sum of the MBR values of the edges between the tasks and the already allocated ones and the LAN-LAN bandwidths are decreased similarly.

5. Performance Evaluation

The evaluation of the scheduling solutions carried out by QLSE was conducted by simulations applying the algorithm to a suite of task graphs and Grid platforms randomly generated.

Tables 1 and 2 show the bounds values defined for each parameter used by a uniform distribution to generate TIGs and Grids, respectively. TIGs and Grids with three different sizes were used in each test. To obtain valid statistical values, for all the 2187 combinations obtained from the TIGs and Grids parameters, 10 TIGs and 10 Grids were randomly generated to run a test. In total, 21870 test cases were generated.

To validate and evaluate QLSE we implemented a greedy scheduling algorithm, and we applied it to every test case.

The algorithm performs a *best fit* heuristics to map the queued tasks on the LANs' hosts.

The validation of QLSE was based on the criterion that it would be able to carry out at least a feasible solution, i.e. a solution that satisfies the MCRs and MBRs required by an application, when at least a valid solution exists for a TIG mapping. This criterion can be deterministically evaluated generating TIGs and Grids in such a way that a valid solution exists. Figure 3 shows the average percentage of scheduling failures obtained by both algorithms w.r.t. the number of tasks per TIGs. A scheduling failure occurs when the algorithm is not able to map a TIG satisfying both MCRs and MBRs requirements. An increment of the number of tasks causes an almost linear increment of the scheduling failures obtained by the Greedy algorithm, while QLSE is able to carry out almost all the valid solutions.

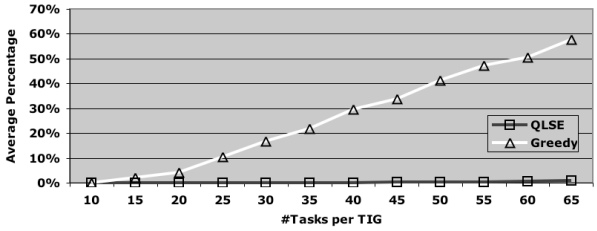

Figure 3. Average percentage of scheduling failures.

To evaluate the mapping carried out by QLSE we exploited two criteria: the LAN hit ratio and the task-machine computational ratio. The LAN hit ratio returns the percentage of communications allocated on the same LAN. It is computed as the ratio between the sum of TIG's MBRs associated to the communicating tasks mapped in the same LAN and the TIG's MBRs sum. An higher LAN hit ratio corresponds to an higher throughput between communicating tasks increases. The task-machine computational ratio measures how the more powerful machines are exploited

Size	$\#\,Task$	$MCR(Mflop/s)$	$MBR(Mb/s)$
Small	$8-32$	$100-200$	$10-30$
Medium	$32-48$	$200-400$	$20-40$
Large	$48-64$	$400-600$	$30-50$

Table 1. Parameters used to generate TIGs.

Size	$\#\,LAN$	$\#\,M.$ $per\ LAN$	$M.Power$ $(Mflop/s)$	LAN $bw(Mb/s)$
S	$8-16$	$4-12$	$600-800$	$20-40$
M	$12-24$	$8-24$	$800-1000$	$30-50$
L	$16-32$	$16-48$	$1000-1300$	$40-60$

Table 2. Parameters used to generate Grids.

to run TIG's tasks. An higher task-machine computational ratio corresponds to an higher application throughput. It is defined as the ratio between the computational power of the machine where a task is mapped and the task's MCR. To compute these performance parameters values only the simulations that carried out a feasible solution by both algorithms were considered.

Figure 4 shows the results of the average LAN hit ratio, expressed in percentage, w.r.t. the number of tasks per TIG, using QLSE and the greedy algorithm.

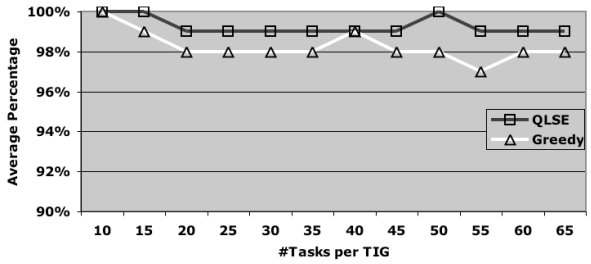

Figure 4. Average LAN Hit Ratio.

In these tests both algorithms obtain a LAN hit ratio very close to 100%. We can observe that QLSE is always able to obtain better values than the greedy algorithm, since it is able to allocate in the same LANs the tasks with an higher degree of communication.

Figure 5 shows the average task-machine computational ratio w.r.t. the number of tasks per TIG.

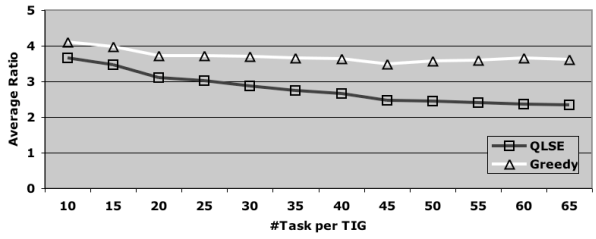

Figure 5. Average task-machine computational ratio.

It can be observed that QLSE is able to map the tasks on machines with a computational power 2 or 3 times greater than their MCR. The Greedy algorithm obtained a average task-machine computational ratio always greater than QLSE, because it selects first the LANs with a greater aggregate computational power,thereafter it exploits for the valid solutions a smaller number of LANs (no more than 2 in our test cases) than QLSE to map a TIG. However, by exploiting appropriate inter-LANs links, QLSE is able to garantee the communication QoS (i.e. the required TIG's MBRs) also when TIG's tasks are mapped on more LANs.

These results have been obtained running the algorithms on a Pentium4 processor with a clock frequency of 3.0GHz and a memory of 1GB.

6. Conclusions and Future Work

In this paper a new heuristics to map at launching time parallel applications on Grid platforms is described. The heuristics is oriented to large computational Grids made up of dynamic not-dedicated resources, generally known as wide-area Grids. The mapping of an application is conducted in order to meet both application computational and communication requirements (QoS). A preliminary performance analysis was conducted by simulation applying the algorithm to a suite of task graphs and Grid platforms randomly generated. Moreover, such analysis was done by comparing the proposed algorithm with a greedy one, which exploits a best fit heuristics. The conducted tests demonstrated that our algorithm was able to carry out a valid solution (i.e. satisfying the QoS) in almost the 100% of the simulated test cases. Moreover, the high values obtained for both the tasks communication and computation throughput demonstrated the applicability of our approach.

References

[1] R. Baraglia, R. Ferrini, N. Tonellotto, L. Ricci, and R. Yahyapour. A launch-time scheduling heuristics for parallel applications on wide area grids. *Journal of Grid Computing*, 2007 to appear.

[2] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov. Adaptive Computing on the Grid Using AppLeS. *IEEE Trans. Parallel Distrib. Syst.*, 14(4):369–382, 2003.

[3] M. den Burger, T. Kielmann, and H. E. Bal. Topomon: A monitoring tool for grid network topology. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part II*, pages 558–567, London, UK, 2002. Springer-Verlag.

[4] A. Gerasoulis and T. Yang. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. *J. Parallel Distrib. Comput.*, 16(4):276–291, 1992.

[5] J. Nabrzyski, J. M. Schopf, and J. Weglarz, editors. *Grid resource management: state of the art and future trends*. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[6] A. Radulescu and A. J. C. van Gemund. On the complexity of list scheduling algorithms for distributed-memory systems. In *ICS '99: Proceedings of the 13th international conference on Supercomputing*, pages 68–75, New York, NY, USA, 1999. ACM Press.

[7] H. Topcuouglu, S. Hariri, and M. Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.*, 13(3):260–274, 2002.

[8] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Gener. Comput. Syst.*, 15(5-6):757–768, 1999.