Integration of Grid Cost Model into ISS/VIOLA Meta-scheduler Environment

Ralf Gruber[1,6], Vincent Keller[1,6], Michela Thiémard[2,6], Oliver Wäldrich[3,6], Philipp Wieder[4,6], Wolfgang Ziegler[3,6], and Pierre Manneback[5,6]

[1] École Polytechnique Fédérale de Lausanne, LIN-STI, Switzerland
Ralf.Gruber@epfl.ch, Vincent.Keller@epfl.ch
[2] École Polytechnique Fédérale de Lausanne, DIT-EX, Switzerland
Michela.Thiemard@epfl.ch
[3] Fraunhofer Gesellschaft, SCAI, St. Augustin, Germany
Oliver.Waeldrich@scai.fraunhofer.de, Wolfgang.Ziegler@scai.fraunhofer.de
[4] Froschungszentrum Jülich GmbH, D-52425, Germany
ph.wieder@fz-juelich.de
[5] Faculté Polytechnique de Mons and CETIC, Mons, Belgium
Pierre.Manneback@fpms.ac.be
[6] members of CoreGRID

Abstract. The Broker with the cost function model of the ISS/VIOLA Meta-Scheduling System implementation is described in details. The Broker includes all the algorithmic steps needed to determine a well suited machine for an application component. This judicious choice is based on a deterministic cost function model including a set of parameters that can be adapted to policies set up by computing centres or application owners. All the quantities needed for the cost function can be found in the DataWarehouse, or are available through the schedulers of the different machines forming the Grid. An ISS-Simulator has been designed to simulate the real-life scheduling of existent clusters and to virtually include new parallel machines. It will be used to validate the cost model and to tune the different free parameters.

1 Introduction

The Intelligent Grid Scheduling System (ISS) [3] has been proposed to submit n components C_k $(1 \leq k \leq n)$ of an application A to a computational Grid composed of r resources R_i $(1 \leq i \leq r)$ each being a parallel machine with p_i nodes and m_i main memory [9]. A component is executed on $p_k \leq p_i$ processors. It is planned to apply ISS first to the HPC applications in Switzerland that are executed on the parallel machines that form the SwissGrid. These machines are located at the Swiss National Supercomputing Centre (CSCS) in Manno, at the EPFL in Lausanne, at the ETHZ in Zürich, and at other Universities and research institutions in Switzerland. The aim of ISS is to submit the components of the different applications to well suited machines according to a deterministic cost function. This cost function is presented in this paper.

W. Lehner et al. (Eds.): Euro-Par 2006 Workshops, LNCS 4375, pp. 215–224, 2007.

The ISS cost function includes terms that represent the investment costs per sustained Tflops/s rate, energy consumption due to power supply and cooling, maintenance, licences, and management costs, the infrastructure (room and cooling system) and expenses for personnel taking care of the resources. The sustained Tflops/s rate strongly depends on the usage of the machines and on the type of application components that are executed. Machines that are not used all the time can be expensive. Ecological arguments are now more and more considered when deciding on the purchase of a machine. These can be taken care of by the energy price per kWh, and by the cooling installation costs. With all those characteristics, the overall costs of a component can vary by up to an order of magnitude on the different machines. ISS should optimize the usage of the different machines and help to decide on the policy when purchasing future machines.

In a previous paper [1], the integration of the ISS into the VIOLA Meta-Scheduler environment has been described. Specifically, the roles of the Meta-Scheduler and UNICORE clients have been detailed. The whole scheduling of a component C_k on machine i has been decomposed in three major steps: Prologue (starts at time t_k^0) and Decision (made just after t_k^0), Execution (starts at time $t_{k,i}^s$), and Epilogue (starts at time $t_{k,i}^e$ and ends after data collection at $t_{k,i}^d$). In the Prologue phase, data needed to construct the cost model are collected and used to minimize the cost function. The resulting choice is then forwarded to the UNICORE client for submission to the chosen machine. The information about the execution phase is treated in the Epilogue phase to create a file that can be reused in the next job submission.

Different modules help to decide. Besides the UNICORE [4] and the Meta-Scheduling Services (MSS) [5] there are three new modules: The DataWarehouse (DW), the System Information (SI), and the Broker. Before execution of a component, all the cost model relevant data collected during previous executions on different machines can be found in the DW. They are accessed through the SI and transferred to the Broker. SI also collects the data after execution that is prepared by the VAMOS system [6] and sent to the Broker. The VAMOS system maps Ganglia and accounting data into application relevent ones. This Epilogue data are interpreted according to the Γ model [2]. This model characterizes parallel machines and applications. These data on the behaviour of the component during execution are then stored by the SI in the DW, prepared to be reused in the Prologue phase for the next execution.

The Broker includes all the algorithmic steps for the evaluation of the cost model and for the preparation of the Epilogue data. The needed data are requested from the SI and the MSS. All features of the Broker are described in detail in this paper.

The application components C_k are parametrized by the Γ model [2] in which the needs in processor performance, main memory bandwidth, network communication bandwidth and latency are estimated. Together with similar parameters describing the parallel machines, a Γ value is computed that measures the computation over communication needs of C_k. Γ model relevant parameters such as

the number of operations O, the number of transferred data S, or the effective processor performance r_a can be measured on a machine that includes PAPI [10]. For those machines that do not have PAPI, these parameters can be estimated according to the input parameters of the next execution, or with the Γ model using measurements on one machine and by transforming them to the other.

The cost function model includes free parameters that have to be tuned for each Grid. For this purpose, a Simulator has been designed.

2 Application Component Characteristics

The major reason for the development of an Intelligent Grid Scheduling System (ISS) is the different needs of the application components in point of view of processing performance, main memory bandwidth, communication bandwidth and network latency. These characteristics have been parametrized in the Γ model [2]. Similarly, the computer architectures have also been parametrized in the same paper. Some important parameters can be directly measured using Ganglia data [11]. This enables predicting to which machine an application component should be submitted. For the Swiss HPC community the following type of applications consume the major part of the HPC resources:

2.1 Embarrassingly Parallel Application Components

These applications do not demand inter-node communications. As a consequence, the Γ parameter is huge. A big number of cases have to be executed, the results collected and handled by a server. Typical applications are the immense amount of independent data in high energy physics that has to be interpreted, the sequencing algorithms in proteomics, parameter studies in plasma physics to predict optimal magnetic fusion configurations, or a huge number of data base accesses for statistical reasons.

Embarrassingly parallel applications need master/slave computer architectures with a sufficiently powerful connectivity between the frontend server and the different slaves. There is no communication needed between the slaves. Thus, a weekly connected workstation farm can offer a sufficient computing performance. Such clusters can for instance be formed of individual machines connected through a bus-based network or even through the internet to a master. As a consequence, the costs of such application components can be small.

2.2 Application Components with Point-to-Point Communications

Point-to-point communications typically appear in finite element or finite volume methods when a huge 3D domain is decomposed in subdomains [7] and an explicit time stepping method or an iterative matrix solver is applied. If the number of processors grows with the problem size, and the size of a subdomain is fixed, γ_a (=number of operations O over amount of data S sent over the network) is independent of the problem size, and, consequently, Γ does not change. The

per processor performance is determined by the main memory bandwidth. The number O of operations per step is directly related to the number of variables in a subdomain times the number of operations per variable, whereas the amount of data S transferred to the neighboring subdomains is directly related to the number of variables on the subdomain surface. For huge point-to-point applications using many processing nodes, $\Gamma << 1$ for a bus (inadequate) $2 < \Gamma < 10$ for the Pentium 4 cluster with a Fast Ethernet switch, $10 < \Gamma < 50$ for the Xeon cluster with a GbE switch, and the Opteron cluster with a Myrinet switch, and $\Gamma >> 100$ for a Cray XT3. Application components with $\Gamma > 1$ can run well on a cluster with a relatively slow and cost-effective communication network.

2.3 Application Components with Multicast Communication Needs

The parallel 3D FFT algorithm is a typical example with important multicast communication needs. Here, γ_a decreases when the problem size is increased, and the communication network has to become faster. Since $r_a = R_\infty$, FFT reaches close to peak performance. Thus, γ_M is big, and, as a consequence, the communication parameter b must be big to satisfy $\Gamma > 1$. Such an application has been discussed in [2]. It has been showed that with a Fast Ethernet based switched network, the communication time is several times bigger than the computing time, even when the problem size is small. Such an application needs a faster switched network such as an efficient GbE, a Myrinet, a Quadrics, or an Infiniband network.

2.4 Components Demanding Shared Memory

There are a few application components that demand a shared memory computer architecture. A typical example are the direct simulation applications to study turbulence phenomena applying a spectral method to the Navier-Stokes equations. The main memory needs are small, the component can be run on single processor machine. Very small phenomena have to be studied, needing very small time steps. Typically, a million of time steps are needed for one simulation, one step takes a few seconds on one processor. This leads to a few months of CPU time per case. The user likes to distribute one case among a number of processors. It can be seen that a distributed memory architecture is not well suited for such a problem [8]. The reason comes from the fixed overall size of the application. If the number of processors is increased, the per processor size reduces, and the scalability is very poor. A real shared memory architecture is more adequate. To reduce the turn-around times of these application components, the Swiss computational Grid should include a few shared memory nodes.

3 Meta-scheduling Features

The Meta-Scheduling Service (MSS) delivers data on the availability of the different eligible machines in a Grid as a function of the number p_k of processors

reserved for a component. By means of these data it will be possible to estimate $t^s_{k,i}$, i.e. the starting time of the application component C_k on machine i. The time difference $t^s_{k,i} - t^0_k$ is the time during which the component will sit in the input queue. In fact, if p_k is high the execution time $t^e_{k,i} - t^s_{k,i}$ is small, but the waiting time $t^s_{k,i} - t^0_k$ can become big.

4 The Broker

4.1 Action List

The Broker is active during the Prologue, the Decision, and the Epilogue phases. It computes all data related to the cost model. The tasks of the Broker are:

1. Interprete job input data received from the UNICORE client
2. Collect application related data from DW through SI
3. Collect data on machine availabilities through MSS
4. Evaluate cost function and chooses a well suited machine
5. Send decision to MSS, after reservation to UNICORE client
6. Collect data on execution through SI
7. Prepare Γ model related data
8. Send epilogue data to DW through SI.

4.2 Decision: Grid Cost Model

The choice of a well suited machine depends on user requisites. Some users would like to obtain the result of their application execution as soon as possible, regardless of costs, some others would like to obtain results for a given maximum cost, but in a reasonable time, and some others for a minimum cost, regardless of time.

We will describe here in a few words the various elements that compose a cost function z being able to satisfy users' requests. This cost function depends on costs due to machine usage, denoted by K_e, license fees, denoted by K_l, energy consumption and cooling, denoted by K_{eco}, waiting results time, denoted by K_w, and amount of data transferred, denoted by K_d. Introducing two more quantities, $KMAX$ and $TMAX$, respectively maximum cost and maximum time, from users point of view, we can formulate the following optimization problem:

$$\begin{aligned}
&\min z = \beta K_w(C_k, R_i, p_k) + \sum_{k=1}^{n} \mathcal{F}_{C_k}(R_i, p_k) \\
&\text{such that} \sum_{k=1}^{n} \Big(K_e(C_k, R_i, p_k) + K_l(C_k, R_i, p_k) \\
&\qquad + K_{eco}(C_k, R_i, p_k) + K_d(C_k, R_i, p_k) \Big) \leq KMAX \\
&\qquad\qquad max(t^d_{k,i}) - min(t^0_k) \leq TMAX \\
&\qquad\qquad (R_i, p_k) \in \mathcal{R}(C_k),
\end{aligned}$$

$\forall\ 1 \leq k \leq n$, where

$$\begin{aligned}\mathcal{F}_{C_k}(R_i, p_k) = \ & \alpha_k\Big(K_e(C_k, R_i, p_k) + K_l(C_k, R_i, p_k)\Big) \\ & + \beta_k\Big(K_w(C_k, R_i, p_k)\Big) + \gamma_k\Big(K_{eco}(C_k, R_i, p_k)\Big) \\ & + \delta_k\Big(K_d(C_k, R_i, p_k)\Big) \quad [\text{ECU}]\,,\end{aligned}$$

$$\alpha_k,\ \beta,\ \gamma_k,\ \delta_k \geq 0,$$

$$\alpha_k + \beta + \gamma_k + \delta_k > 0,$$

and $\mathcal{R}(C_k), k = 1, ..., n$ is the eligible set of machines for component C_k. We express the money quantity as Electronic Cost Unit ([ECU]).

In our model, the parameters α_k, β, γ_k, and δ_k are used to weight the different terms. They can be fixed by the users and/or by a simulator. For instance, by fixing $\alpha_k = \gamma_k = \delta_k = 0$ and $\beta \neq 0$, one can get the result as rapidly as possible, independent of cost. By fixing $\beta = 0$ and $\alpha_k, \gamma_k, \delta_k \neq 0$, one can get the result for minimum cost, independent of time. These four parameters have to be tuned according to the policies of the computing centres and user's demands. In the case of the Swiss HPC Grid, the overall usage of the machines should be high. For instance, increasing β will increase usage of underused machines. One recognizes that a simulator is needed to estimate these parameters.

The quantities K_e and K_l have the same weight α_k. The reason is that license fees are either paid directly by the user, then $K_l = 0$, fully paid by the computing centre, then, the license fee is part of K_e, and $K_l = 0$ again, or it is invoiced per unit CPU time, then, $K_l > 0$.

The K_d quantity depends on the amount of data transferred. The other K values are:

Costs Due to Machine Usage: K_e

$$K_e(C_k, R_i, p_k) = \int_{t^s_{k,i}}^{t^e_{k,i}} k_e(C_k, R_i, p_k, \varphi, t)\, dt\ [\text{ECU}].$$

Each computing center has its specific accounting, but generally they just bill the execution time, depending on the number of CPU time used. Figure 1 shows an example of $k_e(t)$ when day time, night time and weekends have different CPU costs. The φ parameter introduces the **priority** notion.

Costs Due to License Fees: K_l

$$K_l(C_k, R_i, p_k) = \int_{t^s_{k,i}}^{t^e_{k,i}} k_l(C_k, R_i, p_k, t)\, dt\ [\text{ECU}].$$

As costs due to machine usage, costs due to licenses may simply depend on execution time and the number of CPUs used, independent of day time.

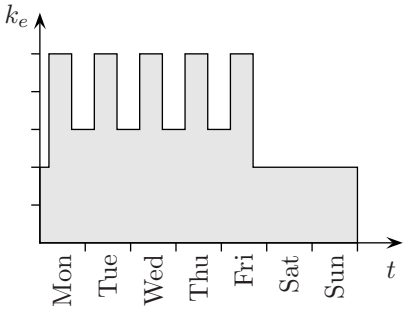

Fig. 1. Example of CPU costs as a function of daytime

Costs Due to Turn-Around Time: K_w

$$K_w(C_k, R_i, p_k) = \int_{min(t_k^0)}^{max(t_{k,i}^d)} k_w(t)\, dt\, [\text{ECU}].$$

This cost is machine and application component dependent since $t_{k,i}^e$ is machine and component dependent. It could be engineer's salary or a critical time-to-market product waiting cost.

Figure 2 shows an example of k_w concerning engineer's salary. Here, it is supposed that the engineer looses his time only during working hours. A more sophisticated function could be yearly graphs also including unproductive periods like vacations.

Figure 2 shows an example of k_w of a critical time-to-market product. This parameter could be also be used to tune the overall usage of the whole machine park of a user community. Increasing β will activate machines that are underused. Putting $\beta = 0$ in the simulator offers the opportunity to recognize overused machines that should in addition be installed.

Costs Due to Energy Consumption and Cooling: K_{eco}

$$K_{eco}(C_k, R_i, p_k) = \int_{t_{k,i}^s}^{t_{k,i}^e} k_{eco}(C_k, R_i, p_k, t)\, dt\, [\text{ECU}].$$

This cost can become relatively important if low-cost PCs are used in clusters. For components that are memory bandwidth bound, the frequency of the processor could be lowered. The energy consumption grows with the second power of the frequency, a reduction by a factor of 2.5 of the processor frequency reduces its energy consumption by a factor of 6. This has been tested with a laptop computer. When reducing frequency from 2 GHz to 800 MHz, the overall performance of the memory bandwidth bound application only was reduced by 10%. We have to mention here that for low-cost PCs energy costs are comparable to investment costs. Thus, in future it is crucial to be able to underclock the processor, adapting its frequency to the application component needs. This would reduce the

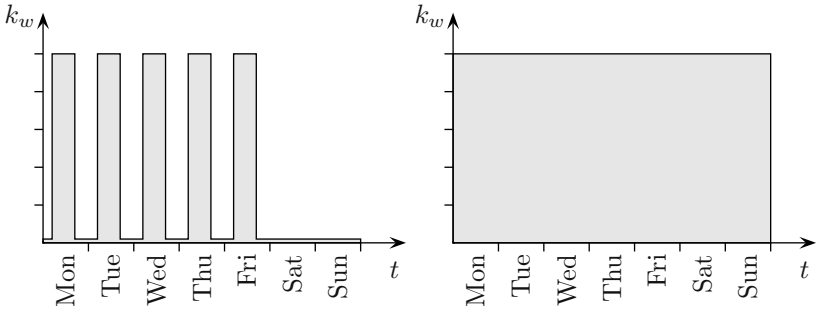

Fig. 2. Left: Engineer's salary cost function $k_w(t)$ due to waiting on the result. Right: Time-to-market arguments can push up priority of the job.

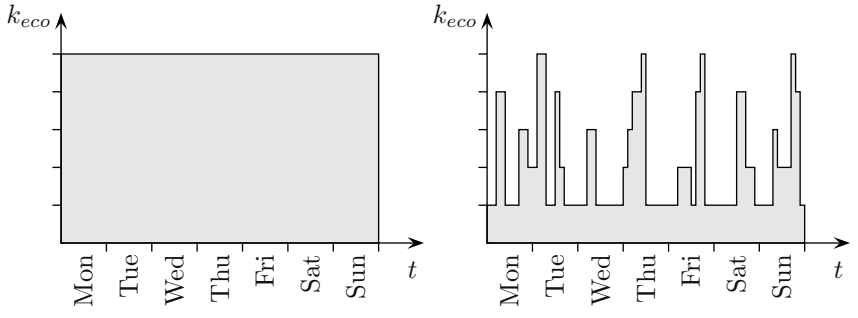

Fig. 3. Today: Excessive costs of energy consumption and cooling. Future: Energy consumption reduction due to frequency adaptation to application component needs. Computer manufacturers are invited to open for on-line frequency underclocking.

worldwide PC energy consumption by at least 75% and would free in the near future 30 nuclear power plants. Computer manufacturers must be convinced to be able to have energy consumption graphs as the one depicted in Figure 3.

4.3 Epilogue: Prepare Data for Next Execution

After execution, the SI collects the application component data from all the databases build up by Ganglia on each node. The Γ model relevant information is extracted and sent to the Broker which computes the Γ model [2] data used for the next execution of the same application component. This data is then sent to the DW through SI.

In addition to the execution data, information on the choice of the machine is also stored on the DW such that in an ulterior step a statistical study can be performed on the adequacy of the machines in the Grid for the set of application components submitted during a certain period in time. This study can then be used to get some insight on which machine should in future be purchased. After a few years it will then be possible to tend towards a well adapted set of machines that form the Grid for a given user community.

5 Simulator to Tune Parameters

The cost model equations include the free variables α, β, γ and δ. It is not clear how they have to be chosen. The ISS-Simulator has been designed to validate the choices of these parameters. In fact, the ISS-Simulator aims to understand the learning process of the Grid system with real machine parameters and real monitored data (see [3]) coming from real applications. It can also be used to predict an imaginary, well suited, set of machines adapted to the applications of a real or a virtual user community. This prediction could also be used in the future to help computer centers to buy new machines. This simulator will be described in more details in another paper.

6 Conclusions

The Broker with the cost function model of the ISS/VIOLA Meta-Scheduling System implementation has been described in detail. The Broker includes all the algorithmic steps needed to determine a well suited machine for an application component. This judicious choice is based on a deterministic cost function model including a set of parameters that can be adapted to policies set up by computing centres or application owners. All the quantities needed for the cost function can be found in the DW, or are available through the schedulers of the different machines in the Grid. An ISS-Simulator has been designed to simulate the real-life scheduling of existent clusters and to virtually include new parallel machines. It is used to validate the cost model and to tune the different free parameters.

Acknowledgement

The development of ISS is a joint project between LIN-EPFL, EIF (Ecole d'ingénieurs et d'architectes de Fribourg) and CSCS (Swiss National Computing Centre), and is part of the Swiss Grid Initiative. Some of the work reported in this paper is funded by the German Federal Ministry of Education and Research through the VIOLA project under grant #01AK605L. This paper also includes work carried out jointly within the CoreGRID Network of Excellence funded by the European Commission's IST program under grant #004265.

References

1. Keller, V., Cristiano, K., Gruber, R., Spada, M., Tran, T.-M., Kuonen, P., Wieder, P., Ziegler, W., Maffioletti, S., Nellari, N., Sawley, M.-C.: Integration of ISS into the VIOLA Meta-Scheduler environment. Pisa, 28-30 November 2005.
2. Gruber, R., Volgers, P., De Vita, A., Stengel, M., Tran, T.-M.: Parameterisation to tailor commodity clusters to applications. *Future Generation Computer Systems*, **19**:111–120, 2003.

3. Gruber, R., Keller, V., Kuonen, P., Sawley, M.-C., Schaeli, B., Tolou, A., Torruella, M., and Tran, T.-M., Towards an Intelligent Grid Scheduling System, Proc. of 6th Int. Conf. PPAM 2005, Poznan, Poland, Lecture Notes in Computer Science 3911 (Springer, 2006) 751-757
4. Erwin, D.,UNICORE plus final report – uniform interface to computing resource,Forschungszentrum Jülich,2003,ISBN 3-00-011592-7
5. Wäldrich, O., Wieder, P., and Ziegler, W., A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resource, In *Proc. of Conference on Parallel Processing and Applied Mathematics PPAM 2005*, Poznan, Poland, 2005, to appear.
6. Gruber, R. and Keller,V. ,Towards an Eco-Grid architecture, submitted to Future Generation Computer Systems
7. Gruber, R. and Tran, T.-M. Scalability aspects on commodity clusters, EPFL Supercomputing Review, **14**, 12-17 (2004)
8. Gruber, R., Keller, V., Leriche, E., and Habisreutinger, M.A., Can a Helmholtz solver run on a cluster?, accepted to appear in Procs. of Cluster 2006
9. Manneback, P., Bergére, G., Emad, N., Gruber, R., Keller, V., Kuonen, P., Noël, S., and Petiton, S., Proposal of a scheduling policy for hybrid methods on computationsl Grids, CoreGRID workshop (Pisa, 2005)
10. Dongarra, J., London, K., Moore, S., Mucci, P., and Terpstra, D., Using PAPI for hardware performance monitoring on Linux systems, www.netlib.org/utk/people/JackDongarra/PAPERS/papi-linux.pdf
11. http://ganglia.sourceforge.net/