1998

An Alternative Model for Scheduling on a Computational Grid

Dan C. Marinescu

Ladislau Bölöni

Ruibing Hao

Kyung-Koo Jun

Report Number:
98-023

AN ALTERNATIVE MODEL FOR SCHEDULING ON A COMPUTATIONAL GRID

Dan C. Marinescu
Ladislau Boloni
Ruibing Hao
Kyung-Koo Jun

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR #98-023
August 1998

An Alternative Model for Scheduling on a Computational Grid

Dan C. Marinescu, Ladislau Bölöni, Ruibing Hao, and Kyung-Koo Jun
Computer Sciences Department
Purdue University
West Lafayette, IN 47907

August 4, 1998

Abstract

In this paper we discuss scheduling on a computational grid of autonomous nodes. A two level process involves a meta-scheduler whose objective is to generate an optimal schedule and reduce the total execution time of an work-flow, and local schedulers whose objective is to optimize the utilization of local resources. We introduce several resource allocation and consumption models and discuss issues pertinent to decision making with incomplete and/or outdated information. Then we present a stock market model for scheduling on a computer grid with autonomous nodes.

1 Introduction

A computational grid is a large-scale, heterogeneous collection of autonomous systems, geographically distributed and interconnected by low latency and high bandwidth networks. The mission of a grid is to provide dependable services at a low cost for a large community of users and to support collaborative work. The term "computational grid" is based on the analogy with the "power grid", it reflects the desire to view the network as a pool of resources and to provide a commodity very much like the electric power and it does

not imply any particular topology of the network. The most difficult problems in a computational grid are the management and control of resources, dependability and security.

A power grid interconnects a large number of resources with an even larger number of consumers, for example the North American power grid has about 10,000 generators, 100,000 transmission lines, 70,000 sub-stations and 130,000,000,000 customers. Similarly, computational grids can be interconnected with one another to form regional, national and even global grids with possibly millions of systems and hundreds of millions of users. Networks of workstations, NOWs, represent particular forms of grids.

The heterogeneity of the grid nodes implies hardware heterogeneity and software diversity. Processors have different instruction sets, the architecture of the nodes is different, there are MIMD as well as SIMD, distributed and shared memory systems, the local resources of each node, main memory, secondary and tertiary storage, are different. Specialized nodes provide mass storage, image processing, and so on. The software available differs from node to node. The grid should support *permanent services* whose life-time is of the order of the life-time of the system, as well as *transient services*, activated on-demand to satisfy a request of one user of the system.

Autonomy implies that new systems can be added to the grid and existing systems can be removed from the grid without affecting applications running on the grid. The management of local resources of each node of the grid are under the control of a local agent, that may cooperate with external agents to solve problems with a scope broader than a single system but once a task is accepted, the system can work in isolation to complete that task.

Low latency and high bandwidth networks are required to guarantee an acceptable response time. Some of the applications of the grid may be data intensive, others may involve visualization and computation steering, some may require real-time delivery constraints. A computational task for the grid is specified as a work flow of activities and requires concurrent access to multiple resources and coordination among different activities.

Efforts to identify the long-term research problems for computational grids and the short-term strategies for building grids are under way [15]. The industry is building commercial grids for transaction-oriented reactive applications using distributed object frameworks. The driving force in building such grids are "natural grid applications" like the electronic commerce where the distribution is part of the problem not part of the solution. In the

case of grids for scientific and engineering applications the grid is viewed as the only viable solution for solving problems that cannot be solved using the largest systems available at a given time.

A number of ongoing meta-computing projects [4], [7], [8],[9],[11] investigate environments able to transform a computer grid into a user's meta-computer with abundant resources. Such an environment is expected to provide services we are accustomed to from single computer systems like directory, event, monitoring, scheduling and other services needed to support dependable computing. In this paper we are concerned only with resource allocation.

2 Resource availability and consumption models

In a meta-computing environment we expect support for scheduling dependent tasks [1], [2], [3], [13], [14]. The dependency graph is derived from the meta-program provided as input to a scheduling agent whose functions are: (a) map computational tasks to services, (b) map services to grid nodes, (c) ensure the data flow required by the meta-program. Scheduling on a grid of autonomous nodes raises difficult problems inherent to any decentralized system. The problems become even less tractable when there is contention for system resources.

Further complications arise when the knowledge about the time of need and the amount of each resource is imprecise. In the general case the knowledge of the individual needs of work-flows and of the resource availability is imprecise. Most systems require the user to provide estimates of the resource consumption of individual tasks of a work-flow. Sometimes the system maintains history databases and attempts to infer the resource consumption based upon past executions. Gathering status information about individual resources in a system with a large number of nodes is non-trivial because of the dynamics of the system, new nodes join the system, others quit. There is also a considerable overhead for gathering status information and the information becomes obsolete quickly.

When the execution time of individual components of a meta-program and the data transfer times over the network are known a meta-scheduler is

able to pre-compute an optimal schedule. An *optimal schedule* is one that minimizes the total execution time of the meta-program and it can only be enforced when there is no contention for system resources, a situation unlikely to occur on a computer grid with independent nodes. It follows that a meta-scheduler for a grid should pre-compute a family of near optimal schedules and attempt to adjust to the actual load of the grid. After the fact it can estimate the quality of service provided by the grid by comparing the actual schedule with the optimal one. When the execution time of the meta-components is not known, the meta-scheduler will only be able to compute an optimal schedule after the fact. The meta-scheduler will assign each component of the meta-program to the fastest system it can execute on and ensure that each component starts the execution at the earliest time after data dependency are satisfied, a strategy called *best effort scheduling.*

We now discuss models for scheduling on a computer grid involving local and meta-schedulers. *Local schedulers* control resource allocation on each node of the grid, their objective is to satisfy a class of preferred customers and maximize the throughput of the local system. *A meta-scheduler* is a transient agent whose objective is to optimize the execution of the application it controls.

In our models decisions are made by local agents that have access to accurate information about the local system and by remote agents that have incomplete or possibly outdated information about the state of grid nodes. Network latency and bandwidth limit the quality and the quantity of information available to a remote decision making agent. To quantify these concepts we introduce the following notations:

δ - network latency;

ψ - event realization delay;

τ - time needed by the remote agent to make a decision;

β - network bandwidth available for transmitting state information;

η - frequency of state information updates;

I - amount of state information required by the remote decision making agent;

A - amount of control information sent by the remote agent to individual nodes;

C - amount of information for coordination of remote decision making agents;

θ - the life-time of a model parameter;

	Guaranteed Access (G)	Non-guaranteed Access (N)
Deterministic execution (D)	GD	ND
Non-deterministic execution (N)	GN	NN

Table 1: Scheduling Models for a Computing Grid with Autonomous Nodes.

A first relation relates the quantity of information needed to make decisions to the available bandwidth: $\eta \times I + A + C \leq \beta$.

A second relation expresses that fact that one needs to hide behind a *temporal firewall* all parameters of a model whose life-time is shorter than the time needed to gather information, to calculate an action and send back the necessary directives to nodes of the grid, $\theta \geq \left(\psi + 2\delta + \frac{I+A}{\beta} + \tau\right)$.

The first resource availability model is based upon the assumption that access to resources is *guaranteed.* This model corresponds to systems which support resource reservations or to those with dedicated resources. For example, in a real-time system, critical tasks require exclusive access to resources; a large parallel system is shared using reservation schemes. A multi-computer system is space shared among different process groups and in this case one uses a batch queuing reservation system. The reservation schemes we are considering are slightly more sophisticated, they imply multi-class schedulers and guaranteed CPU time for each class.

The other availability model is based upon *non-guaranteed* resource access. In this case once a component is ready to run, the meta-scheduler determines where to execute the component subject to a set of constrains and based upon some knowledge of the state of the system. A network of resource brokers and QoS monitors provides the information necessary to map computations to available resources in an optimal way.

The *deterministic execution* resource consumption model, is based upon the assumption that the resources needed for each task and the time when they are needed are known in advance and the actual resource consumption will never exceed our expectations. In case of *non-deterministic execution* we may have no knowledge about the needs of a task, as is the case of a transient service provided by a newly written program, we may know an estimated value and be assured that there are small deviations from this expected value, or that there are large deviations from the expected value, as in the case of cases apply to some permanent and transient services.

Scheduling of meta programs with components whose execution time is a random variable requires characterization of the work requirements imposed upon the grid. If X[0,t] is the amount of work generated by a meta-program in the interval [0,t] and X[0,t] has stationary increments then *the effective bandwidth of a meta-program* [10] is: $\alpha(s,t) = 1/st \times logE[e^{s \times X[0,t]}]$ for $0 < s,t < \infty$ If $X[0,t] = \sum_i X_i[0,t]$ where $X_i[0,t]$ are independent then $\alpha(s,t) = \sum \alpha_i(s,t)$. Analysis of several types of sources including periodic ones, policed and shaped sources can be carried out.

When we combine the models described above we end up with the four cases presented in Table 1. The model studied extensively is GD, guaranteed access and deterministic execution. In this case an optimal or near optimal schedule can be pre-computed; the most difficult aspect of this computation is searching a possibly large solution space for optimal schedules. A variety of approximate methods can be used to reduce the time needed to compute a near-optimal solution, for example genetic algorithms and simulated annealing Schedulers based upon a data flow model are used for the ND case for example the one presented in [12]. Efforts to address the GN model are also reported [3]. But the most difficult problems are raised by the NN model. In the following section we introduce a model that has the potential to accommodate the uncertainties concerning resource availability and resource needs.

3 A stock market model for scheduling on a computer grid

We propose a macro model for resource allocation and consumption on a computer grid. The stock market provides intriguing answers to questions like estimation of the "value" of a resource based upon *market consensus.* Both the providers of a service and the consumers may act to raise or lower the price of a resource viewed as a commodity, based upon the demand for that resource. High demand for a resource will raise its price, low demand will lower the price. At the same time, the consumers of the resource may influence the price based upon the quality of service evaluated after the fact.

Trading involves a decentralized system. Individuals with little or no experience contact brokers who have global knowledge of all the stocks. Brokers

provide advise and act as proxies for the individual clients. Finally, brokers trade on the floor of a clearinghouse.

In our model meta-schedulers are transient agents whose only objective is to minimize the execution time of an work-flow without knowing the state of the individual resources available in the system, they are like clients with money but no knowledge of the stocks. Meta-schedulers need to work with *system brokers*, agents capable to locate resources and carry out the trading. In turn, *local brokers* manage resources available on every node of the network. There is also the need for a *clearinghouse*, the equivalent of the Stock Exchange. The model does not support scheduling of activities with hard deadlines. In most cases a best-effort policy is implemented.

The model we propose is suitable for dynamic systems consisting of large numbers of nodes and many consumers. The consumer population and the the number of nodes varies rapidly. We also assume that many nodes are capable to provide the same service and that each consumer generates *work-flows, or contracts* in our terminology, with many sub-contracts. Only under these assumptions the value of a resource established through market consensus is meaningful and brokers play a useful role. If a service is available only on few nodes the consumer of the service may contact directly the service providers.

Still missing from the model are indications on how to handle uncertainties about the amount of resources needed to accomplish a task. Relaying on information supplied by the user leads to under utilization of resources when the user takes a conservative approach or to a waste of resources in case of aggressive specification. Databases with historic information are perfectly suitable to provide hints when the variability of resource consumption is low. For example, the time needed to improve the dynamic range of an image through histogram equalization is a function of the image size, known a-priori. But the time taken by an algorithm to identify objects on an image depends not only upon the image size, but also upon the actual number of objects present in the image.

We assume that critical resources are the CPU time needed to complete a task and the communication time. The implicit assumption is that once a task is allocated CPU time on a node of the grid other resources needed by the task, for example main memory, disk space, and so on are allocated as well. The alternative is to consider vectors of resources and negotiate using these vectors.

We propose to use concepts from *futures and options markets* to accommodate the variability of resource consumption. A system broker acquires options to resources anticipating the needs of the potential consumers. Each acquisition is an option to acquire a specified amount of resources at a given time in the future for a given price per unit of resource. Clients, in our case meta-schedulers, contact system brokers with requests providing the start-up time and the estimated time of use of the resource. On the service provider side, local brokers offer to provide access to resources knowing the current state of the system.

Before describing our model in more detail we introduce the basic concepts pertinent to futures and options markets [14]. An *European call option* is a contract with the following conditions: at a prescribed time in the future known as the *expiry date* the holder of an option may purchase an asset for a prescribed amount, known as the *exercise price*. The other party of the contract is called the *writer* and has the obligation to sell the asset if the holder chooses to buy it. An *American option* is one that may be exercised at any time prior to expiry. Note that this contract is a *right* and not an *obligation*. Since the option confers to the buyer a right with no obligation it must have some value. The buyer must pay a price for this option at the time of signing the contract.

The price of options magnifies changes in the price of the asset, this effect is called *gearing*. An example will illustrate the amplification effect. Assume that the price of a call option is 10% of the current price of an asset and the value of the asset doubles by the expiry date. If the exercise price of the option is 20% lower than the market value of the asset at the expiration date of the contract, than the buyer of the option will make a 400% profit. If the value of the asset at the contract signing time was \$1,000 then the cost of the call option was \$100, the market value of the asset at the expiry date was \$2,000, and the value of the contract was \$1,600. The buyer made \$400 on an investment of only \$100.

Two critical questions are: (a) what is the value of an option, and (b) how can the writer minimize the risk associated with its obligation. While the price of the asset at the expiry date is not known at the time of purchesing an option, it seems reasonable that the higher is the price of the asset now, the higher the price is likely to be in the future. Therefore, the value of a call option today depends upon the today's price of the asset. The value of the option depends also upon the exercise price, the lower the exercise price,

the higher the option value. The call option price is also a function of the expiry date. Just before the expiry date there is little time for the price of the asset to change. Last but not least, the option value depends upon the volatility of the asset and the interest rates.

The right to sell an asset is called a *put option*, it allows a holder to sell an asset on a certain date for a prescribed amount. The writer is then obliged to buy the asset. The put option has payoff properties opposite to that of a call option. The holder of a call option wishes the price of the asset to raise, the holder of a put option wants the price of the asset to be as low as possible. The value of a put option increases with the exercise price.

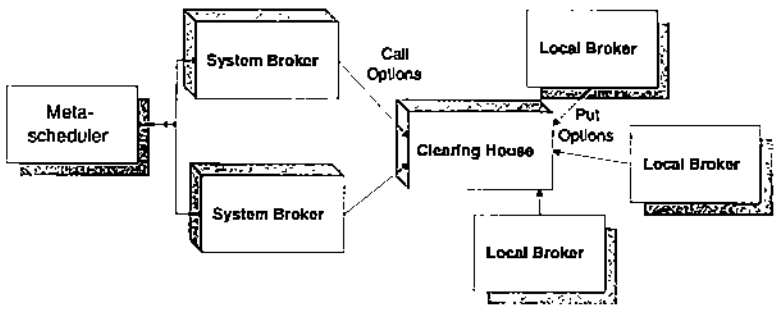

Figure 1: Stockmarkect model consists of meta-schedulers, system brokers, local brokers, and clearing house

We consider the following model for meta-scheduling on a grid, shown in Fig. 1.

(a) There are n resources, each under the control of a local broker. The local broker sells immediate access to the local resouce at the current price when the resource is available and issues put options reflecting its belief that there will be an excess of capacity at some future time. A put option is a tuple identifying the resource, the exercise time, the value, and the amount.

If LB_i^{begin} is the total values of the assets available for the local broker LB_i at the beginning of the scheduling period and LB_i^{end} the value at the end, the objective is to maximize the *local broker gain ratio* defined as: $LB_i^{gain} = LB_i^{end}/LB_i^{begin}$. Here $LB_i^{end} = \sum_k t_i^k$ with t_i^k the amount of each transaction carried out by the local broker. There is no reward for resources that have not been consumed. A local broker unable to make any trade will have a gain $LB_i^{gain} = 0$ and the gain for a profitable one will be $LB_i^{gain} > 1$. The current

price of a resource $V_i = Price(resource_i)$, $i = 1, n$ is established through market consensus.

(b) There are $m << n$ independent system brokers, SB_j, $j = 1, m$. At the beginning of each scheduling cycle a system broker is allocated an initial capital SA_j^{begin} and it invests some of it into call options and some into buying immediate access to resources. Its objective is to maximize the *system broker gain ratio* defined as: $SB_j^{gain} = SB_j^{end}/SB_j^{begin}$. Here $SB_j^{end} = \sum_k t_j^k$ with t_j^k the amount of each transaction performed by the system broker. A system broker unable to make any trade will have a gain $SB_i^{gain} = 0$. The gain of a broker able to break even will be $SB_i^{gain} = 1$ and a profitable one will have $SB_i^{gain} > 1$.

A call option is a tuple identifying the resource, the exercise time, the value, and the amount. The broker sells assets and options in response to requests from meta-schedulers or other system brokers.

(c) A meta-scheduler is assigned an initial capital to cover the cost of completing the meta-program it controls. The meta-scheduler maps computational tasks onto services and multicasts *contract requests* to the network of system brokers. It receives bids from system brokers, selects the best bid for each-subcontract, and issues firm orders.

The model requires the development of several algorithms.

Call-option and negotiation algorithms for a system broker. Call options are issued based upon prediction of client needs. The negotiation algorithms enable the system broker to sell the options it hold to clients or acquire from other system or local brokers resources that may allow the broker to present attractive global packages to clients. The objective of the algorithm is to maximize the gain ratio of the system broker.

Put-option algorithms for a local broker. The put-option algorithm is based upon predicition of local resource availability and the current state of the local system. The objective of the algorithm is to maximize the gain ratio of the local broker.

Selection algorithms for a meta-scheduler. The objective of the meta-scheduler is to ensure the earliest completion time of the the meta-program subject to the funds available for the project.

Quality of service algorithms for a meta-scheduler. After the completion of each sub-contract the meta-scheduler will evalute the quality of service. Quality of service reports will be provided to the system brokers.

Resource price agreement algorithms for brokers and the clearinghouse.

The updated price of a resource is based upon the current market value, the offers and the demands for each resource. The price is also influenced by the quality of service reports. Such reports are provided by individual meta-schedulers at the completion of each sub-contract.

Option price agreement algorithms for brokers and the clearinghouse. The price of an option will reflect the current market value, the exercise price, the expiry date, and the volatility of the resource.

Each of these algorithms have to be designed and implemented, then evaluated independently using analitical and simulation methods. The validation of these models will be rather challenging because a computational grid is still a fiction. Testing these ideas on small-scale testbeds will always raise the question if conclusions drawn on small scale models can be extended to large-scale systems.

Once the question of the feasibility of the model we propose is answered, one had to compare this model with other scheduling models. To carry out these comparisons one needs synthetic workloads for a computational grid. From this brief discussion it follows that the evaluation of the model is a challenging research topic in itself.

The local and system brokers will use prediction for resource availability and consumption. When the data model is known it is understood how to do optimal predictions. But in our case the data model is not known and we need a universal prediction algorithm [16].

Information theory exploits the duality between the growth rate of wealth and the entropy rate in the stock market for defining competitively optimal and growth rate optimal portfolio strategies [6] and may provide useful hints on the design of the algorithms introduced above.

4 Conclusions

In this paper we discuss resource allocation and consumption models and propose a stock market scheduling model for a computer grid with autonomous nodes. At the present time we are investigating analitical and simulation tools to evaluate the model and qualitative arguments supporting the model.

5 Acknowledgments

The work reported in this paper is partially supported by the National Science Foundation grant MCB-9527131, by the California Institute of Technology, under the Scalable I/O Initiative, by the Intel Corporation, and by the Computational Science Alliance and the NCSA at the University of Illinois.

References

[1] M. Atallah, C. Lock, D. C. Marinescu, H.J. Siegel, and T. L. Cassavant. *Models and Algorithms for Co-Scheduling Compute Intensive Tasks on a Network of Workstations.* Journal of Parallel and Distributed Computing. Vol. 16, No. 4, pp. 319-327, 1992.

[2] F. Berman. *High Performance Schedulers in Building a Computational Grid*, I. Foster and Carl Kesselman Eds., Morgan Kaufmann, 1998.

[3] L. Bölöni and D.C. Marinescu. *Robust Scheduling of Meta-programs in a Nondeterministic Environment*, Department of Computer Sciences, Purdue University CSD-TR #98-003.

[4] L. Bölöni, K.K. Jun, M. Sirbu, and D.C. Marinescu. *Seamless Metacomputing with Bond.* 1998 (submitted).

[5] K. Mani Chandy, A Rifkin, Paolo A.G. Sivilotti, J. Mandelson, M. Richardson, W. Tanaka, and L. Weissman. *A World-Wide Distributed System Using Java and the Internet*, IEEE International Symposium on High Performance Distributed Computing, 1996.

[6] Cover and Thomas. *Elements of Information Theory.*

[7] J.C. Fabre and T. Perennou. *A Metaobject Architecture for Fault Tolerant Distributed Systems: The FRIENDS Approach.* IEEE Trans. on Computers, Vol 47, No 1, pp. 78-95, 1998.

[8] I. Foster and C. Kesselman. *The Globus Project: A Status Report.* Proc. Heterogeneous Computing Workshop, 1998.

[9] A. Grimshaw and W. Wulf. *Legion - a View from 50,000 Feet.* Proc. 5-th IEEE Symp. on High Performance Distributed Computing, pp. 89-99, IEEE Press, 1996.

[10] F. Kelly. *Notes on Effective Bandwidth* Royal Statistical Society Lecture Notes Series, Vol 4, pp. 141-168, 1996.

[11] D.C. Marinescu and L. Bölöni. *Reflections on Metacomputing.* Department of Computer Sciences, Purdue University CSD-TR #98-006.

[12] M.G. Sirbu and D.C. Marinescu. *A Scheduling Expert Advisor for Heterogeneous Environments.* Proceedings of the HCW 97 Heterogeneous Computing Workshop, pp. 74-82, IEEE Press, 1997.

[13] Kuei Yu Wang, D.C. Marinescu, and O. F. Carbunar. *Dynamic Scheduling of Process Groups.* Concurrency: Practice and Experince, vol. 10, pp. 265-283, 1998.

[14] P. Wilmot, S. Howison, and J. Dewynne. *The Mathematics of Financial Derivatives.* Cambridge University Press, 1995.

[15] I. Foster and C. Kesselman Editors, *The Grid. Blueprint for a New Computing Infrastructure* Morgan Kaufmann, 1998.

[16] John Kieffer, *Prediction and Information Theory* in Six Lectures on Information Theory.