A Nash-Equilibrium based Algorithm for Scheduling Jobs on a Grid Cluster

Massimo Orazio Spata
massimo.spata@st.com
ST Microelectronics
Dipartimento di Matematica e Informatica
Università di Catania, Italy

Abstract

A distributed system such as a Grid environment needs an optimised scheduler that selects and allocates the most suitable resources for the execution of jobs that users submit. This paper proposes a novel scheduler, based on a microeconomic model. In the underlying model, several players submitting jobs compete to use resources and job allocation is determined by applying a Nash Equilibrium solution.

1 Introduction

A Grid environment can be thought of as a large collection of computing resources, a substantial part of which are devoted to executing jobs. Users can submit jobs to a Grid system abstracting from where they will be allocated and executed. Job submissions are processed by a scheduler, which determines how jobs will be allocated on available resources.

This paper proposes a scheduling algorithm that is based on the Nash equilibrium solution of game theory [5]. Related literature includes works [1] and [3]. The former shows that, with a scheduler based on the Nash equilibrium, the average waiting time in a queue decreases when system resource usage increases. On the contrary, with a scheduling algorithm performing optimal allocation, the average waiting time on a queue increases when resource usage grows. In [3], three different scheduling algorithms are compared, based on Nash equilibrium, random and MinMin strategies respectively. Results show that the MinMin approach turns out to be the best scheduling algorithm. In our (different) application scenario, Nash equilibrium proves more effective, as discussed below

2. Nash Equilibrium

In the following, an *m-player game* is modelled as a tuple of *m strategy profiles*, together with a tuple of *m payoffs u*. Suppose each player j in $1,\ldots,m$ has chosen a strategy s_j. This yields the strategy profile $\boldsymbol{s} = (s_1,\ldots,s_m)$, in which player i will get in return payoff $u_i(\boldsymbol{s})$.

A strategy profile $(s^*_1,\ldots,s^*_m)$ is a Nash equilibrium if no deviation from it by any single player i is profitable; i.e. if strategy s_i replaces s^*_i then, for all j, $1 \leq j \leq m$:

$$u_j(s^*) \geq u_j(s^*_1,\ldots,s^*_{i-1}, s_i, s^*_{i+1},\ldots, s^*_m)$$

In game theory, the *prisoner's dilemma* is a type of non-zero-sum game. In this game, each individual player's ("the prisoner") only concern is to maximize her own payoff, thoroughly disregarding the other player's payoff.

3. Modelling job submission as a game

The problem we intend to address as a game is that of job submission. We assume that $j_1, j_2,\ldots, j_m$ are m jobs submitted for execution, and $WN_1, WN_2, \ldots, WN_c$ are c distinct worker nodes endowed with comparable computing resources. Suppose that the number of jobs m is greater than the number c of worker nodes. It will then be necessary to co-allocate more than one job on some WN.

In the allocation game, the players-agents are the jobs themselves and must choose which worker node to be run on by exploiting the Nash equilibrium. Basically, we assume that jobs know in advance the system load dynamics and aim at maximizing the chance of being completed within the expected service time.

A game theoretic model can be formulated in terms of:

- players = agents = jobs: $j_1, j_2,\ldots, j_m$;
- available moves for each j_k: allocation on WN_1, or $WN_2,\ldots$, or WN_c;
- profit: the probability of meeting the job's desired completion time;
- a generalized payoff matrix M.

The payoff matrix generalises the classical one of the prisoner's dilemma in the usual fashion employed to deal with more than two prisoners. Matrix elements are constructed based on the following parameters:

- *estimated job service time* for each job,
- *job interarrival rate*,
- *estimated load* for each worker node.

The system scheduler exploits standard techniques based on the prisoner's dilemma Nash solution to determine a convenient job allocation. This reflects the Nash equilibrium reached when every agent, having made conjectures about the other agents' strategies, makes the best choice (the one with highest profit) for himself, ensuring that no other agent has other strategies with a higher profit.

3. Experimental Results and Conclusions

Experiments have been performed on a Grid site at STMicroelectronics. The goal of the experiment consisted in simulating EDA (Electronics Design Automation) batch jobs, and analyzing events produced by these simulations.

Our simulation of the Grid cluster models local data on disk storage shared via NFS and local execution of jobs on a LAN. Cluster hosts used for tests can be categorized in the following classes.

- *Class SUN SOLARIS*: V240 SparcIIIi bi-processor servers, with Solaris 8 OS and 8 GB RAM.
- *Class LINUX 32 bit*: Intel Xeon 32 bit bi-processor servers, with Linux Red Hat Enterprise 3.0 Update5 OS and 4 GB of RAM.
- *Class LINUX 64 bit*: AMD Opteron 64 bit bi-processor Servers, with Linux Red Hat Enterprise 3.0 Update5 OS and 16 GB of RAM.

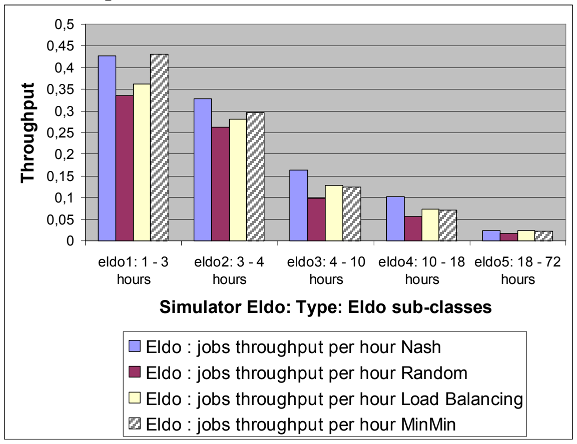

Fig.1 Jobs throughput per hour for "eldo" job classes with different schedulers

We have run about 5000 simulations for four classes of jobs executing the *Eldo* EDA application [4]. Each class is characterized by the job expected execution time, automatically estimated exploiting the technique proposed in [6]. Beside Nash equilibrium scheduling, the other allocation strategies used for comparison are: *MinMin* [3], random, and "load balancing" (as implemented by standard Grid schedulers).

Experiments have demonstrated that the best overall performance, expressed as the job throughput sustained by the system, is indeed obtained using the Nash scheduler. The only exception occurs for jobs whose expected completion time is one hour or less. MinMin exhibits the best response for job completion time below one hour and performs like Nash for completion times is above 18 hours. The range where Nash scheduling clearly outperforms the other is that where typical, most interesting jobs lie.

Acknowledgements

The author wishes to thank Marco Lucio Lovetere, Maria Grazia Messina, Carmelo Pulvirenti, Rosetta Rizzo.

References

[1] D. F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic Models for Allocating Resources in Computer Systems. In *Market-based control: a paradigm for distributed resource allocation*. World Scientific Publishing. 1996.

[2] D. Gross, C. M. Harris. *Fundamentals of Queueing Theory*. Wiley. 1998.

[3] Y.-K. Kwok, S. Song, and K. Hwang, Non-Cooperative Grids: Game-Theoretic Modeling and Strategy Optimization. Submitted to *IEEE Transactions on Parallel and Distributed Systems*, Dec. 2004.

[4] Mentor Graphics Corp. *Eldo simulator web page*. http://www.mentor.com/products/ic_nanometer_design/custom_design_simulation/eldo/index.cfm.

[5] J. Nash. Equilibrium points in n-person games.. *Proceedings of the National Academy of Sciences*, 36, 1950.

[6] M. O. Spata, G. Pappalardo, S. Rinaudo, T. Biondi "*Agent-based negotiation techniques for a Grid: the Prophet Agents*", 2nd IEEE International Conference on e-Science and Grid Computing 2006.