STUDIA INFORMATICA
Nr 2(9) Systemy i technologie informacyjne 2007

Game Theoretical Model Applied to Scheduling in Grid Computing

Piotr Świtalski[1], Franciszek Seredyński[2,3]
[1] The University of Podlasie, Computer Science Department,
ul. Sienkiewicza 51, 08-110 Siedlce, Poland
[2] The University of Warmia and Mazury,
ul. Oczapowskiego 2, 10-719 Olsztyn, Poland
[3] Institute of Computer Science, Polish Academy of Sciences,
ul. Ordona 21, 01-237 Warsaw, Poland

Abstract. We consider a grid computational model which consist of a number of computation nodes and a number of users. Each user generates a computation load (jobs) requesting computational and communication resources. A deadline for each job is also defined. We propose a scheduling algorithm which is based on Iterated Prisoner's Dilemma (IPD) under the Random Pairing game, where nodes (players) of the grid system decide about their behavior: cooperate or defect. In this game players play a game with randomly chosen players and receive payoffs. Each player has strategies which define its decision. Genetic algorithm (GA) is used to evolve strategies to optimize a criterion related to scheduling problem. In this paper we show that GA is able to discover a strategy in the IPD model providing a cooperation between node-players, which permits to solve scheduling problem in grid.

Keywords. scheduling, game theory, prisoner's dilemma, genetic algorithm, grid, task, job

1 Introduction

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1]. A grid is a decentralized heterogeneous system in which computational resources are located in a number of computation nodes. Computation nodes are often personal computers which have different CPU power, amount of memory and bandwidth of communication channel. Their available resources change in time. Each node is attributed to a user which generates a computation load (jobs) requesting computational and communication resources. A job composes of a number of tasks. *Task scheduling* is a process in which tasks are distributed to the nodes with user's requirements.

From user's perspective, a grid is a problem-solving environment in which one or more user jobs can be submitted without knowing where the resources are or

even who owns the resources. The real and specific problem that underlies the grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [2]. This coordinating is difficult because grid nodes are heterogeneous and autonomous. Traditional approaches use centralized policies that need complete state information and a common fabric management policy, or a decentralized consensus-based policy. Due to the complexity in constructing successful Grid environments, it is impossible to define an acceptable system-wide performance matrix and common fabric management policy [3]. For example SETI@home [4], launched in 1999, is a widely-known very simple grid computing project. It uses the processing power of thousands of Idle CPU's that are connected to the Internet. In this system, the CPU power is donated by the users who are considered truthful, and there is no competition between them. But this problem becomes more challenging when resource owners are being paid, or other issues exist such as the social reputation gained by participating in the Grid society [5].

Economical models in a computational grid are simple example of behavior in which nodes (producers and consumers) can be cooperate [4][6][7][8]. In this model a market concentrating producers (resource) and consumers (computation load) are considered. Producers can sell their own resources to consumers for an established price. A pricing policy is a determinant behavior for producers and consumers. Current market-oriented models are based on a general equilibrium theory. The general equilibrium theory can be transformed to an optimization problems, which in this case is a scheduling problem. But this approach has some disadvantages. Firstly, a grid cannot be treated as free market. In the market each node has information about all others nodes. In grid this information is restricted to nodes in their neighbourhood. Secondly, pricing policy do not assume deadline for executing tasks.

In [9] author presents promising idea of resource management system based on the immune system metaphor, making use of the concepts of Immune Network Theory and Danger Theory. By emulating various elements in the immune system, manager could efficiently execute tasks on very large systems of either homogeneous or heterogeneous resources in grids. The distributed nature of the immune system allows efficient scheduling of tasks, even in extremely large environments, without the use of a centralized or hierarchical scheduler.

In this paper we will show a new concept of scheduling in grid computing. This concept is based on game theory. The major point of using this theory is to keep cooperation between nodes. We propose a scheduling algorithm which is based on IPD under the Random Pairing game [14], where nodes (players) of the grid system decide about their behavior: *cooperate* or *defect*. In this game players play a game with randomly chosen players and receive payoffs. Each player has strategies which define its decision. GA is used to evolve strategies to optimize a criterion related to scheduling problem. In this paper we show that GA is able to discover a strategy in the IPD model providing a cooperation between node-players, which permits to solve scheduling problem in grid.

The paper is organized as follows. In the next section, we describe task scheduling problem in computation grid. Section 3 introduces one of game theory model: Prisoner's Dilemma (PD). In this section we also explain evolution of

behavior in IPD under Random Pairing game. Next, in Section 4 we give details of our computational grid model. Last section concludes the paper.

2 Task Scheduling Problem in Grid

In such systems like computational grid, task scheduling is not easy, for the reason that nodes are not centrally controlled and information about their state is available only for their closest neighbourhood. Task scheduling is a core process of resource management systems. The most important point of task scheduling is allocating computation load (divided into tasks) to appropriate resources, attempting to achieve some performance goals as the shortest executing time of user's job or load balancing on the nodes. Here, jobs can be executed both on local and remote nodes. Scheduler (see Fig. 2.1) is a system which take decision about allocating tasks on the nodes in Grid.

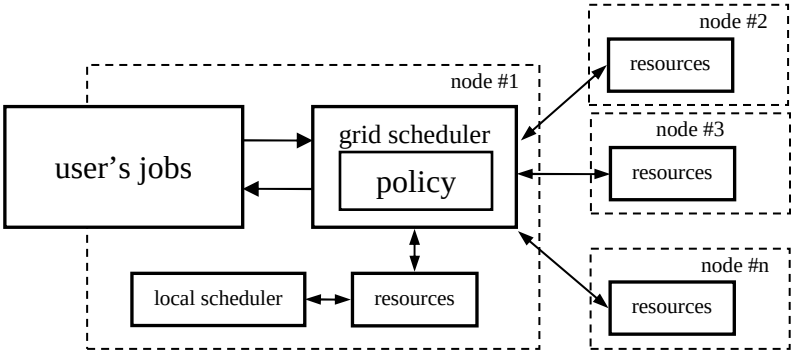

Figure 2.1. Grid and local scheduler

In grid systems we consider two types of schedulers:
- *grid scheduler* which allocates tasks among nodes
- *local scheduler* which allocates tasks in a single node after allocation by the grid scheduler.

Grid scheduler might be de facto each node (Fig. 2.1), because this process starts when the node needs to send several tasks for neighbour nodes. Nodes take their decision autonomously. The scheduling policy determines how an application should be scheduled and how the resources should be utilized. Most importantly, the scheduling policy is responsible for defining the performance goals for the Grid system.

In economical methods scheduling policy is a price. A price specify behaviour of nodes. Nodes have to pay for the executing tasks if they want to send and are paid when they execute tasks coming from the other nodes. In the commodity market model, resource owners specify their service price and charge users according to the amount of resource they consume. The pricing policy can be derived from various parameters and can be flat or variable depending on the resource supply and

demand. In general, services are priced in such a way that supply and demand equilibrium is maintained [6].

2.1 Model of the task

Computation job generated by a user is divided into smaller indivisible parts named tasks. We can consider two models of tasks:

- dependent tasks organized in directed acyclic graph (DAG),
- independent tasks.

In the first model the job can be represented by weighed, directed and acyclic graph $G_p = (V_p; E_p)$ whose vertices v_i are tasks z_i and edges e_u reflect the precedence relations. Each task (vertices) has an execution cost. The weight of an edge is called the communication cost of the edge. The precedence constraints of a DAG dictate that a node cannot start execution before it gathers all the data from its preceding nodes. Graph G_p is called a program graph or precedence task graph [10]. Figure 2.2 presents a precedence graph for four tasks in precedence relation.

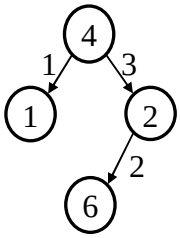

Figure 2.2. An example of a program DAG graph.

The second model is represented by a large number of various size tasks which are independent. Each task similarly to dependent tasks has execution cost, but there is no precedence constraints.

The cost of task executing on the node is the total time of execution of task, time to send of task to node and time to receive results of executed task. The execution cost depends on a number of instructions in the task. Each processor is characterized by a speed, i.e. by a number of instructions computed per unit time.

Hence, time to execute of task is:

$$t_z = \frac{i_z}{S},$$

where i_z is the number of task instructions, S is the speed of a processor. A grid is heterogeneous, so processors in a grid have various speed by nature. Time of task execution will be different in different nodes.

2.2 Model of resources

A resource is any physical or virtual component of limited availability within a computer system [11]. A typical resource types in computers are CPU time, size of

memory, hard disk space, network throughput. In grid systems resources are heterogeneous. Each node may have a specified resources, which are not available on the others nodes. A node often have also more than one resource. A scheduler needs to have information about these resources. Tasks can require specified resources and scheduler have to allocate tasks on nodes where those resources are accessible.

3 Iterated Prisoner's Dilemma under the Random Pairing game

3.1 Background

Participating in a grid system with no rules causes situations where nodes willingly send their own tasks to the other nodes, but they refuse to receive tasks from the other nodes. This behaviour is not good for a society in grid. In societies, trust is a fact of everyday life. A decision to trust is a decision laced with risk [12].

Trust is relevant with reputation. Reputation can be described as the opinion of the public toward a person, a group of people, or an organization [11]. In a grid systems reputation can be interpreted as desire for cooperating. If node has a high reputation, other nodes could cooperate with this node with minimal or without risk.

3.2 Game Theory: Prisoner’s Dilemma

Game is often described as a situation where multiple players have to make a decision in conflicted situations. Such a situation exists when two or more decision makers who have different objectives act on the same system or share the same resources [14].

The Prisoner's Dilemma is non-zero-sum game in which the sum of gains and losses by the players are always more or less than what they began with. There are two players. The players in the game can choose between two moves, either "cooperate" or "defect". For every move players receive payoff. Each player gains when they both cooperate. If only one of them cooperates the other one that defects will receive the highest possible payoff from payoff’s table. If both defect, both lose. The "dilemma" in this game is related to the fact that, whatever the other does, each of them is better off defecting than cooperating. But the outcome obtained when both defect is worse for each than the outcome they would have obtained for both cooperating.

3.3 Iterated Prisoner’s Dilemma under Random Pairing

In the Iterated Prisoner's Dilemma the game is played repeatedly and players memorize their previous encounters. Each player plays against the same opponent for a defined number of rounds. This gives each player an opportunity to punish the other player for previous non-cooperative behavior.

In the Iterated Prisoner's Dilemma under Random Pairing each player plays against a different randomly chosen opponent at every round [14]. In such a case the evolution of cooperative behavior is much more difficult than in the IPD due to short interaction sequence. In the evolutionary version of the game a population of players plays the IPD among themselves [14]. Each player uses his own strategy from a population of strategies. During the evolutionary process the lower scoring

strategies are eliminated and the higher scoring strategies are discovered and increased in number. The process is repeated until the best strategies are found.

4 Grid model

4.1 Model of node

In computational grid participates a large number of nodes. Each node takes decisions autonomously in own environment. A node is a user's system which consist:

- resources (CPU, memory, disk space)
- capability to communicate with the neighbour nodes
- a level of trust.

A user can generate computational load for a Grid. This load is divided into independent tasks. For our model we assumed that tasks are incoming in time with a Poisson distribution. This distribution is used to model the number of events occurring within a given time interval.

Tasks can be inserted into local queue of node or distributes into neighbour nodes. Nodes are heterogeneous, so time to execute of each task will be different on different nodes. Furthermore, for each task will be defined deadline for it execution. This deadline is defined as below:

$$T_{ti} < TF_{ti},$$

where T_{ti} is a real time of task execution, TF_{ti} is deadline time for task execution.

Tasks may miss their deadlines. We introduced penalty model to minimize the loss. Each node has a level of trust (reputation). If given node continuously exceeded deadline, level of its trust is decreasing.

Level of trust is defined as below:

$$tr_i = \frac{nt_i}{nc_i},$$

where:
tr_i is a level of trust,
nt_i is a number of completed tasks without exceed of deadline,
nc_i is a total number of tasks accepted by node.

Tasks in the local queue are scheduled by the heuristic local scheduling algorithm which is described in the algorithm #1.

Algorithm #1

```
insert_place = place_n;
//calculate the penalty if insert the job at insert_place
penalty = calculate_penalty(insert_place);
for (place_i = from place_n-1 to place_0) {
  penalty_i = calculate_penalty(place_i);
  if(penalty_i < penalty) {
    penalty = penalty_i;
    insert_place = place_i;
  }
```

```
}
insert the job at insert_place
```

The approach is based on the fact that when a job is inserted, the relative order of the jobs in the origin queue is often unchanged [15]. In this algorithm we insert task into the queue and calculate penalty. The lowest penalty for the inserted task is the best schedule.

4.2 Cooperating of nodes

Grid scheduler allocates tasks among nodes. Every node can be a grid scheduler. This scheduler gain information from neighbour nodes which contains:

- level of trust,
- length of local queue,
- types of resources,
- other parameters of node.

Algorithm #2

```
if (trust_level_node_i is null) {
  trust_level_node_i = 0.5;
}
divide_load_into_tasks(computational_load_i);
while (get_task_from_queue is null) {
  task = get_task_from_queue;
  task_parameters = get_parameters_of_task(task);
  nodes[] = get_neighbour_nodes;
  /* get the best node for task */
  for (nodes[n] from n=0 to n=max_node) {
    node_parameters = get_parameters_of_task(nodes[n]);
    calc_rate = set_rate(node_parameters,task_parameters);
    if (rate < calc_rate)
      chosen_node = nodes[n];
    else
      rate = calc_rate;
  }
  /* take a decision */
  decision = play_game(chosen_node);
  if (decision is cooperating) {
    send_task(chosen_node,task);
    receive_results;
  } else {
    send_back_task_into_queue;  }}
```

This information is used to choose node which will be participate in executing of tasks. Algorithm #2 shows scheme of node actions when user generates computational load.

When computational load is generated by a user within a node it is divided into tasks and placed into temporary queue. Since this moment tasks are distributed to nodes in neighbourhood. Scheduler is seeking for a node which is the best for chosen task. When a node is chosen, game is played. Node have to take one of two decisions: cooperation (accept of task) or defection (refuse of acceptance of task). If it chooses cooperation, the task may be sent. In the other case we need to repeat this

procedure skipping this node. The trust at a node which accepted and executed a task is increased.

4.3 Taking of decision

Decision of acception or refusing an offer is taking when a node is chosen by node which has task to send. In this situation game is played. Result of this game partly depends on parameters of the node, such as trust level and a number of exceeded of deadlines for tasks. Nevertheless decision is taken by strategy of node. The strategy is represented by a binary string of the length 12 (Fig. 4.1).

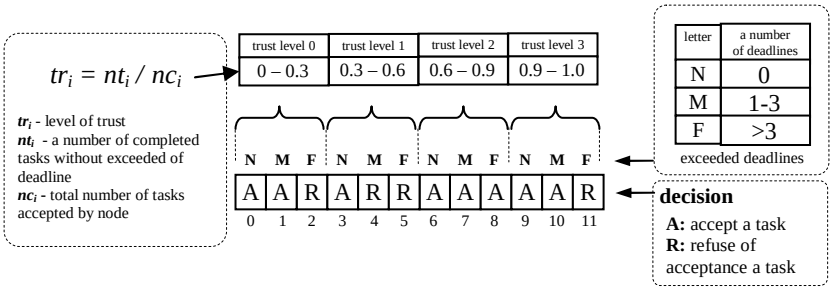

Figure 4.1. Coding of the strategy

The strategy is coding decision which depends on the trust level and a number of exceeded deadlines. Trust level was described in section 4.1. The second parameter describes deadlines which was exceeded for a period of time. This period defines a piece of historical information about busyness of node of local queue. If a number of deadlines is zero, queue is empty or node can execute of tasks from this queue without deadlines. In other case node cannot execute tasks without deadlines.

Let's suppose that node *i* has trust level equal 0.55 and a number of deadlines exceeded in past time is 2 then we turn down a task (see Fig. 4.1).

This strategy is chosen from population of strategies created by evolutionary algorithm. According to IPD under Random Pairing [13] nodes are chosen randomly. Games can be played with known nodes (in neighbourhood). After the game payoff is calculated from the payoff table.

5 Conclusions

In this paper, we have proposed a new approach based on game theory to the task scheduling problem. In order to enforce cooperation of nodes in grids we have used IPD under Random Pairing. It seems that this model can be a good solution for task scheduling in heterogeneous and autonomous systems like computational grids. Currently the model is implemented and will be the subject of intensive study to verify its main assumptions.

References

1. Foster I. and Kesselman C. (ed.), (1999). *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA.
2. Foster I., Kesselman C., Tuecke S., (2001). *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal Supercomputer Applications, 15(3).
3. Ferguson D., Nikolaou C., Sairamesh J., Yemini Y., (1996). *Economic models for allocating resources in computer systems. Market-Based Control: A Paradigm for Distributed Resource Allocation*, World Scientific Press: Singapore.
4. SETI@home: http://setiathome.ssl.berkeley.edu
5. Forghanizadeh S., (2005). *Grid Processor Scheduling based on Game Theoretic Approach*, CPSC532A Final Project.
6. Buyya R., Abramson D., Giddy J. and Stockinger H., (2002). *Economic Models for Resource Management and Scheduling in Grid Computing*, Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, USA.
7. Ygge F., (1998). *Market-Oriented Programming and its Application to Power Load Management*, Ph.D. Thesis, Lund University.
8. Wellman M.P., Walsh W.E., Wurman P.R., MacKie-Mason J.K., (2001). *Auction Protocols for Decentralized Scheduling*, Games and Economic Behavior, 35: 271-303.
9. Wilson L.A., (2008). *Distributed, heterogeneous resource management using artificial immune systems*, International Parallel and Distributed Processing Symposium (IPDPS '08), NIDISC.
10. Switalski P., Seredynski F., Hertel P., (2006). *GAVis System Supporting Visualization, Analysis and Solving Combinatorial Optimization Problems Using Evolutionary Algorithms*, Intelligent Information Processing And Web Mining: Proceedings of the International IIS, 75-84.
11. Wikipedia: http://en.wikipedia.org
12. Marsh S.P., (1994). *Formalising Trust as a Computational Concept. PhD Thesis*, University of Stirling, UK.
13. Camerer C.F., (2003). *Behavioral Game Theory: Experiments in Strategic Interaction*, Princeton University Press.
14. Namikawa N., Ishibuchi H., (2005). *Evolution of Cooperative Behavior in the Iterated Prisoner's Dilemma under Random Pairing in Game Playing*, Proceedings of the Congress on Evolutionary Computation, IEEE Press, 2637-2644.
15. Lijuan X., Yanmin Z., Lionel N. and Zhiwei X., (2005). *GridIS: An Incentive-based Grid Scheduling*, Proceedings of the International Parallel & Distributed Processing Symposium, Denver, Colorado.