# Game Theoretic Analysis of Multi-Processor Schedulers: Matrix Multiplication Example

Oleksii Ignatenko

[1] Institute of Software Systems NAS Ukraine
o.ignatenko@gmail.com

**Abstract.** This paper deals with a game model of users performing parallel computing in a heterogeneous multiprocessor system. The proposed approach is applied to the problem of matrix multiplication on the system with the scheduler of min-min type. The user's action is to choose the size of the blocks into which the matrix is cut. Each user tries to optimize own finish time, which leads to conflict. Using the game theoretic approach, we build game model and found the conditions of Nash equilibrium existence in the scheduling game of two users. We developed simulation model to verify the results.

**Keywords.** parallel computing, schedulers, game theory, Nash equilibrium, Pareto efficiency.

**Key Terms.** MathematicalModel, MultiAgentSystem, Infrastructure

## 1 Introduction

Modern scientific problems require significant computing resources, so the problem of resource optimization in multiprocessor environments is very important. Nowadays, computing algorithms operate in complex heterogeneous environments. It is common to have a distributed computational unit with many simultaneous programs from different users competing for computing and network resources. In most cases, the user is unable to control the distribution of resources. The allocation algorithms may contain defects and inefficiency and this can lead to a significant increase in processing time. That is why distributed computing requires efficient algorithms providing a flexible and stable allocation of resources. The problem is to deal with unfair and uneven access to resources, caused by heterogeneity of users and their tasks where each user is a rational agent that tries to increase its share of resources. This could bring the system to the inefficient equilibrium. A key element is an efficient algorithm for load distribution – scheduler, providing services to users.
The idea of this work is to apply the game-theoretic approach to the problem of scheduling and allocation of computing resources in a dynamic heterogeneous environment with many competitive users.

In this work we consider heterogeneous multi-processor computing system and associated constraint set a finite pool of processor resources (for each node), and limited link capacity. Our goal is to specify work schedule for each node in the way that the computing time (the time when the latest task is finished) is the smallest. This schedule also should satisfy all defined constraints. This is the classical multiprocessor scheduling problem, investigated in many works starting from [1]. It is well known that this problem is NP-hard. One possible solution is to reduce scheduling complexity by using fluid model setup [2]. Another approach to this problem is to consider a game-theoretic approach, which is the current trend in networks and computing systems [3]. In this work we construct static non-cooperative game for load balancing problem in single-class job distributed systems. This problem is investigated by many authors in cooperative (optimal) and game-oriented settings (see [4 - 7] for references). Here, we propose a new analysis of scheduler for matrix multiplication problem in game setup. The proposed approach is applied to the problem of matrix multiplication on the scheduler of min-min type. The user's action is to choose the size of the blocks into which the matrix is cut. Each user tries to optimize own finish time, which eventually brings the system to an equilibrium state. This equilibrium is defined by scheduler's type, so characteristic of equilibrium is important for scheduling policy analysis.

As a result, we presented conditions for the existence of Nash equlibrium end proved Pareto inefficiency. Finally, we implement simulations using CloudSim package [8] and provide experiments on a heterogeneous multi-processor system to verify simulation and theoretic models.

## 2 Matrix Multiplication Model

We consider an analytic computational model of parallel matrices multiplication developed in papers [9, 10]. It is assumed that computation nodes cannot interact and share data. Every node is connected with scheduler. Scheduler receives computational tasks from users and sends them to an available processor (according to its resource allocation algorithm). Every single processor works on its task and returns result. Let us fix notation. Consider the heterogeneous multi-processor system of $m$ computational elements, and every element has capacity $p_i$, $i = 1, \dots, m$. Tasks are translated by network lines, which are assumed to be identical and have capacity $q$.

In this work it is assumed that:

1. All processors start to work at the same time;
2. Scheduler makes assignments instantly;
3. Scheduling is deterministic.

We will proceed by building simple fluid model for matrices multiplication ignoring transfer delays. Secondly, we provide generalization by including data transfer into model. Finally, we consider discrete model, which is the closest to practice.

### 2.1 Simplified Fluid Model

First, let us consider parallel multiplication of two square matrices $N \times N$. Using Block matrix multiplication algorithm user specifies block size $n$. Let the size of block $n$ be a natural number such that $k = \frac{N^2}{n^2}$ is a natural number too. The algorithm will produce $k$ tasks, each with the complexity of $O(n^2)$. Let $T(N, n)$ be the time when all user's tasks are finished. The problem of time optimal matrix multiplication is the minimization function problem:

$$T(N, n) \to min$$

Function $T(N, n)$ has in general many local extremes, so searching for global minimum could be resource demanding problem.

Promising direction of research is the fluid model analysis [2]. Let the user choose vector $x(k) \in R^m$ with components $x_i(k)$, where $x_i(k) \geq 0$, $x_i(k) \leq k$, $i = 1, \dots, m$, $\sum_{i=1}^{m} x_i(k) = k$. Denote $X(n)$ as the set of all vectors $x(n)$. Every component $x_i(k)$ is the amount of computation designed for execution on $i$-th processor. Firstly, we take into account only multiplication operation. This is a simplification but it allows us to understand basic properties of the computation process.

Finish time was found in [9] and is defined by the following formulae

$$T(x, X(n)) = \max_{i=1,..,m} \left\{\frac{x_i N n^2}{p_i}\right\}.$$

The idea of the proposed approach is a fluid approximation to the problem. For the approximation, we assume that work is composed of homogeneous fluid rather than discrete jobs.

**Proposition 1.** [9] Minimal task finish time (excluding transmit time) for a fluid model with one user is equal $T = N^3(\sum_{i=1}^{m} p_i)^{-1}$ and $argmin(T(x)) = N(p_1, \dots, p_m)$.

In this work we employ general approach using convex analysis functions, which we expect to be a promising direction for future investigation of more complex systems.

Define Minkowski functional for set $X$ and vector $p \in R^m$ as:

$$\mu_X(p) = inf\{\lambda > 0: \ p \in \lambda X\}.$$

It is known that Minkowski functional is convex for convex $X$. Let us define the set of capabilities of the computation system $R = \{r \in R^m: \ r_i \in [0, p_i]\}$ and rescale it as following: $R(n) = \frac{R}{Nn^2}$

**Proposition 2.** The following equality is true $T(x, X(n)) = \mu_{R(n)}(X)$.

**Proof**. Consider right-hand side. It is clear that$\mu_{R(n)}(x) = inf\{\lambda > 0: \ x \in \lambda R(x)\}$. A Vector $x$ belongs to the set $R$ if and only if $\max_{i=1,..,m} \left\{\frac{x_i}{p_i}\right\} = 1$, so $\mu_{R(n)}(x) = inf\left\{\lambda > 0: \ \max_{i=1,..,m} \left\{\frac{x_i}{p_i}\right\} = \frac{\lambda}{Nn^2}\right\}$. This is equivalent to equality $\lambda = \max_{i=1,..,m} \left\{\frac{x_i N n^2}{p_i}\right\}$.

**Note.** There is exists minimum $T_{min} = \min_{x \in X(n)} \mu_{R(n)}(x)$ and this minimum is unique.

### 2.2 General Fluid Model

To take into account transfer time we note, that block algorithm sends $2x_iNn$ elements on each node and receives $x_in^2$ elements. So, summary time is equal to $T_s(x, X(n)) = \max\limits_{i=1,\dots,m}\left\{\frac{x_iNn^2}{p_i} + \frac{x_i(n^2+2Nm)}{q}\right\}$.

**Proposition 3.** There is minimum of $T_s(x, X(n))$ with respect to $x \in X(n)$

*Proof.* $\mathrm{X(n)}$ is convex compact. Function $\mathrm{T_s(x, X(n))}$ is continuous and convex, so there is the minimum point.

### 2.3 Discrete Model and Schedulers

Time minimization should be performed on finite net:

$$Y(n) = \{y \in R^m : y_i \in \{0,1,\dots,k\}, \textstyle\sum_i y_i = k, i = 1,..,m\}.$$

It is clear that $Y(n) \subset X(n)$.

Now we consider the notion of a scheduler.

The user chooses the size of the block $n$ and $Y(n)$. The scheduler is an algorithm responsible for specifying concrete point $y^* \in Y(n)$. In this work, we will consider only simple scheduler of extreme – extreme type and perform simulations and investigation for min-min scheduler only. The min-min algorithm is quite popular and simple. The idea of min-min is following.

1. The scheduler receives $k$ tasks, every task has complexity $Nn^2$;
2. The scheduler chooses task with minimal computation complexity (in the case of equal tasks the choice is random);
3. The scheduler sends chosen task on the free processor with maximal capacity or waits in the case of no free resources.
4. If the queue is not empty then return to 2.

The result of algorithm is the pair of vector $y$ and finish time (finish time of the last task)

Fix the size of block $n$ and consider finish time.

**Proposition 4.** For arbitrary $n$ following inequalities are true:

$$T_d(n) = \min_{y \in Y(n)} T(y, X(n)) \geq \min_{x \in X(n)} T(x, X(n))$$

$$T_{minmin}(n) \geq T_d(n)$$

*Proof.* Here we give a sketch of proof. The first inequality is true because $Y(n) \subset X(n)$. The second inequality holds since $T_d(n)$ is minimum by definition.

## 3 Non-Cooperative Scheduling Game Formulation

We will deal with non-cooperative games in strategic or normal form. A non-cooperativeness here does not imply that the players do not cooperate, but it means that any cooperation must be self-enforcing without any coordination among the players. The strict definition is as follows.

A non-cooperative game in strategic (or normal) form is a triplet $G = \{N, \{S_i\}_{i \in N}, \{u_i\}_{i \in N}\}$, where:

• $N$ is a finite set of players;

• $S_i$ is the set of admissible strategies for player i.

• $u_i: S \to R$ is the utility (payoff) function for player i.

A game is said to be static if the players take their actions only once, independently of each other. In some sense, a static game is a game without any notion of time, where no player has any knowledge of the decisions taken by the other players.

Based on the assumption that all players are rational, the players try to maximize their payoffs when responding to other players' strategies. Generally speaking, the final result is determined by non-cooperative maximization of integrated utility. In this regard, the most accepted solution concept for a non-cooperative game is Nash equilibrium [3], introduced by John F. Nash. Loosely speaking, Nash equilibrium is a state of a non-cooperative game where no player can improve its utility by changing its strategy if the other players maintain their current strategies. Formally, pure-strategy Nash equilibrium (NE) of a non-cooperative game $G$ is a strategy profile $s^* \in S$ such that for all $i \in N$ we have the following:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*) \text{ for all } s_i \in S_i \ .$$

Here $s_{-i}$ denotes the vector of strategies of all players except *i*. In other words, a strategy profile is a pure-strategy Nash equilibrium if no player has an incentive to unilaterally deviate to another strategy, given that other players' strategies remain fixed.

Let us formulate scheduling game for matrix multiplication problem. Players are users of distributed system. Define the set of players as $\{u_i\}_{i \in L}$, where $L$ is set of indexes. Users have available multi-processors system with $m$ computational elements. Each element has capacity $p_i$, $i = 1, .., m$. We assume every user has two square matrices with dimension $n_l$, $l \in L$. The set of admissible strategies for player l is a conjunction of all possible cuts $k_l \in K_l$, , $l \in L$. After the player chooses his strategy blocks are translated to a scheduler, which sends them to processors.

Define finish time for l-th player as the time of finishing of last task $T_l$, $l \in L$. Every player wants to minimize his finish time.

In current work we consider problem when:

1. The min-min scheduler is used.
2. The set of admissible sizes $\{n_j\}_{j=1,..,s}$ is sorted in ascending order.
3. If players choose the same sizes, their finish times are the same. Finish time in this case is equal to double individual time for this size.
4. There is unique minimum $T_d(n_j)$, $j = 1, \dots, s$. Denote argminimum index as $j^*$.
5. There are two players in the system.

In order to use game theory methods, we build the game matrix, using following rules. Let players choose strategies $n_1, n_2$. Denote their payoffs as $T_1(n_1, n_2)$ and $T_2(n_1, n_2)$ respectively.

1. If $n_1 < n_2$, then $T_1(n_1, n_2) = T_d(n_1)$, $T_2(n_1, n_2) = T_d(n_1) + T_d(n_2)$.
2. If $n_1 = n_2$, then $T_1(n_1, n_1) = T_2(n_1, n_1) = 2T_d(n_1)$.

**Proposition 5.** If there is exists $j^*$ such that inequality $2T_d(n_{j^*}) \leq T_d(n_{j^*-1})$ holds, then $(n_{j^*}, n_{j^*})$ – Nash equilibrium.

Proof. By definition $(n_{j^*}, n_{j^*})$ is Nash equilibrium if:

$$T_1(n_{j^*}, n_{j^*}) \leq T_1(n_{j^*-1}, n_{j^*}), T_2(n_{j^*}, n_{j^*}) \leq T_2(n_{j^*}, n_{j^*-1}).$$

Using rules, defined above we obtain:

$$T_1(n_{j^*}, n_{j^*}) = 2T_d(n_{j^*}),\ T_1(n_{j^*-1}, n_{j^*}) = T_d(n_{j^*-1}).$$

The same is valid for the second player. If inequality $2T_d(n_{j^*}) \leq T_d(n_{j^*-1})$ holds, $(n_{j^*}, n_{j^*})$ – Nash equilibrium.

**Proposition 6.** If there is exists $j^*$ such that inequality $2T_d(n_{j^*}) \geq T_d(n_{j^*-1})$ holds, then $(n_{j^*-1}, n_{j^*-1})$ – Nash equilibrium.

Proof. By definition $(n_{j^*-1}, n_{j^*-1})$ is Nash equilibrium if:

$$T_1(n_{j^*}, n_{j^*}) \geq T_1(n_{j^*-1}, n_{j^*}),\ T_2(n_{j^*}, n_{j^*}) \geq T_2(n_{j^*}, n_{j^*-1}).$$

Using rules, defined above we obtain:

$$T_1(n_{j^*}, n_{j^*-1}) = T_d(n_{j^*-1}) + T_d(n_{j^*}),\ T_1(n_{j^*-1}, n_{j^*-1}) = 2T_d(n_{j^*-1}).$$

The same is valid for the second player. If inequality $2T_d(n_{j^*}) \geq T_d(n_{j^*-1})$ holds, then $T_1(n_{j^*}, n_{j^*-1}) \geq T_1(n_{j^*-1}, n_{j^*})$ and $(n_{j^*-1}, n_{j^*-1})$ – Nash equilibrium.

**Note.** Nash equilibrium is always Pareto-inefficient, in other words $T_1(n_{j^*}, n_{j^*}) \leq T_1(n_{j^*-1}, n_{j^*})$, $T_2(n_{j^*}, n_{j^*}) \leq T_2(n_{j^*-1}, n_{j^*})$. This is common situation in games of considered type.

## 4 Simulations

We have implemented simulations using CloudSim package and real multi-processing system of Institute of Software Systems NAS Ukraine. The experiments were designed to optimize finish time for one user. Matrices of 1200 on 1200 dimension were used. The experiment was performed on the node with two processors Quad Core Intel Xeon E5405 2GHz and 16 GB DDR2 @667 Mhz RAM (making 7 slave servers and 1 scheduler node).

The graph of Time-Size dependency is shown in the Fig. 1.

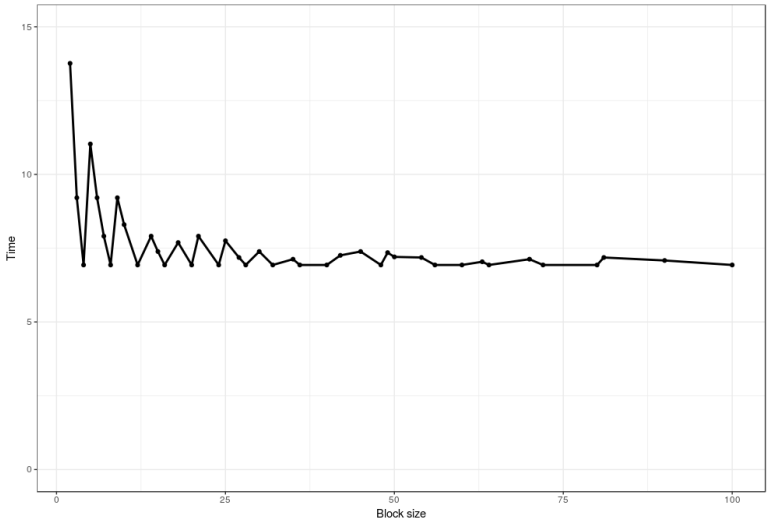


**Fig. 1.** Finish time – size of task graph

Using estimation of the main parameters from experiments we have built the simulation model for scheduling game with two players (1200x1200 matrix, strategies – size of tasks) and min-min scheduler. Results are shown in the Fig 2.

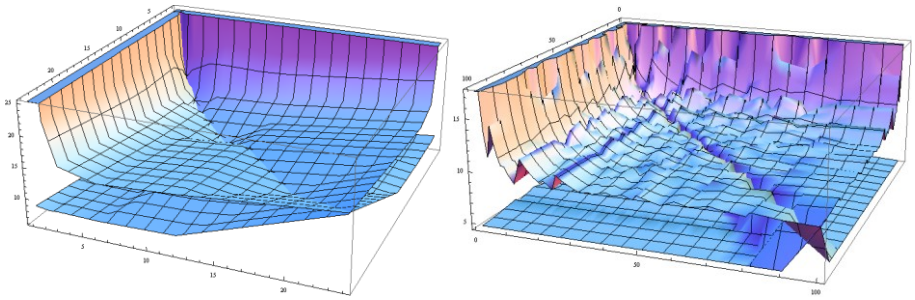

**Fig. 2.** Finish time surface for scheduling game of two players. Real experiments approximation (left) and simulation using theoretic construction (right)

## 5 Conclusions

In this paper, we have presented the approach to deal with problems of scheduling and load balancing using a game-theoretic framework. The general objective was to identify and address the efficiency problems, where game theory can be applied to the model and evaluate user conflicts problems and consequently to design efficient solution. We consider the fluid model of computations to calculate the "ideal" finish time, which gives the lower bound of possible real time. We propose the game model of user's interaction on the example of matrix multiplication problem. We use simulation environment GridSim to obtain experimental data and validate theoretical results.

We investigate the interaction model's users performing parallel computing in a heterogeneous multiprocessor system. The proposed hike is applied to the problem of matrix multiplication using the scheduler min-min. Resolving this case is the size of the blocks into which the matrix is cut. The experimental system characteristics have been used to adjust the simulation model, allowing to measure the time estimate for completion of all possible combinations of partitioning tasks to processors and end time to build finish time surface for each user. The findings were substantiated and summarized based on the game approach, in particular, found the conditions of Nash equilibrium point existence in the game interaction between two users.

## References


1. Srinivasa Prasanna G.N., Musicus B. Generalized Multiprocessor Scheduling Using Optimal Control // Proc. SPAA. – 1991, P. 216 – 228.
2. Nazarathy Y., Weiss G. A Fluid Approach to Large Volume Job Shop Scheduling // Journal of Scheduling, 13(5), 509-529, (2010).
3. Han, Zhu, et al. Game theory in wireless and communication networks. Cambridge University Press, 2012.

4. S. Penmatsa, A. T. Chronopoulos. Game-theoretic static load balancing for distributed systems. Journal of Parallel and Distributed Computing 71, no. 4 (2011): 537-555
5. Wei, Guiyi, et al. A game-theoretic method of fair resource allocation for cloud computing services. The journal of supercomputing 54.2 (2010): 252-269.
6. Siar, Hajar, Kourosh Kiani, and Anthony T. Chronopoulos. An effective game theoretic static load balancing applied to distributed computing. Cluster Computing 18.4 (2015): 1609-1623.
7. Li, Kai, Yong Wang, and Meilin Liu. A non-cooperative game model for reliability-based task scheduling in cloud computing. arXiv preprint arXiv:1403.5012 (2014).
8. https://en.wikipedia.org/wiki/CloudSim
9. Anatoly, Doroshenko, Ignatenko Oleksii, and Ivanenko Pavlo. One model of optimal resource allocation in homogeneous multiprocessor system // Problems in programming 1 (2011): 29-39.
10. Andon, F. I., and O. P. Ignatenko. "Modeling conflict processes on the internet." Cybernetics and Systems Analysis 49.4 (2013): 616-623.
11. Ignatenko, O., Synetskyi, O. (2014). Evolutionary Game of N Competing AIMD Connections. In Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 325-342). Springer International Publishing.