Bull. Nov. Comp. Center, Comp. Science, 27 (2008), 41–50
© 2008 NCC Publisher

Equilibrium prices in economy of the GRID resource allocation: Part two

S.V. Bredikhin, A.B. Khutoretsky, E.A. Tiunova

Abstract. We consider a market model, which has arised under conditions of the paid usage of CPU time. A new algorithm for calculating the price of the local (for one processor) equilibrium is offered under the assumption that the users' preferences are described by the Cobb–Douglas utility functions. An example shows that equilibrium in a multiprocessor system does not necessarily exist. We describe a process, in which each job may move from one processor to another to increase utility. This process is finite and results in a Nash equilibrium for the corresponding game.

1. Introduction

The problem of financing the operating and development costs for a distributed information system can be solved by providing the paid services to users. Economic mechanisms of pricing are applicable for the resource management in these systems.

In [1], the problem of distributing the processor time in a multiprocessor system is considered under the assumption that the users' preferences are described by utility functions in the Cobb–Douglas form. The authors introduce definitions of the local equilibria (for one processor) and equilibria in a multiprocessor system and offer an algorithm for calculating the equilibrium for an arbitrary group of users at a single processor. Nevertheless, the existence of equilibrium for the whole system remains an open question. In this paper, the dynamic distribution and redistribution of jobs among processors is described. The authors hazard a conjecture that this process is finite.

Our objective is to continue an analysis of local equilibria and equilibria in a multiprocessor system under the same assumption. In Section 2, we construct a new algorithm for calculating the local equilibrium, which is more convenient and better interpreted than that in [1]. In Section 3, we give an example of a two-processor system having no equilibrium. A possibility of such a situation stimulates an interest to other methods of an expedient distribution of jobs among processors. In Section 4, under the assumption that each job can move from one processor to another, if it is profitable for the user, we show that the arising process of reallocations is finite and results in a Nash equilibrium of the corresponding game.

2. A market model

The following market model refines the one described in [2].

The supplier $j \in \{1, \ldots, n\}$ owns the processor of speed V_j (cycles per unit period) and selects the supply q_j for the next unit period. The supplier's cost function is $C_j(q_j) = c_j \cdot q_j$ if $q_j \le V_j$, otherwise $C_j(q_j) = \infty$. Let p_j be the price for the processor j's services within a certain unit period (i.e., the value of one cycle). The supplier maximizes the profit:

$$(p_j - c_j)q_j \to \max \quad \text{s.t.} \quad q_j \in [0; V_j]. \tag{1}$$

The user $i \in \{1, \ldots, m\}$ has a job of scope $Q_i > 0$ and a budget $B_i > 0$. For the next unit period he selects a triple $(j(i), s_i, z_i)$, where $j(i) \in \{1, \ldots, n\}$ is the number of a processor for solving the job; $s_i \in [0; Q_i]$ is the speed of the job solution; $z_i \ge 0$ is a remainder of the user's budget by the beginning of the following unit period. The user may change a processor only at the boundary of a unit period. This move takes neither time nor money. The user's utility function (in the Cobb–Douglas form) is:

$$U_i(s_i, z_i) = s_i^{a_i} m_i^{b_i}, \tag{2}$$

where $a_i > 0$ and $b_i > 0$. For a price vector $\boldsymbol{p} = (p_j)_j$ the user i's budget restriction is

$$s_i p_{j(i)} + z_i \le B_i. \tag{3}$$

It is obvious that the solution of supplier's problem (1) is the following point-to-set mapping $S_j(p_j)$:

$$S_j(p_j) = \begin{cases} \{0\}, & p_j < c_j; \\ [0; V_j], & p_j = c_j; \\ \{V_j\}, & p_j > c_j. \end{cases} \tag{4}$$

In other words, if the price does not cover the cost, then the supply is absent; if the profit is positive, then the supplier would like to operate at the full power; if the price equals constant marginal costs, then all possible values of the supply are indifferent for the supplier.

The user i with prices $\boldsymbol{p} = (p_j)_1^n$ maximizes his utility function under the budget restriction. For each j, he searches a pair (s_i^j, m_i^j), which is optimal in the case of selecting the processor j. Then he selects $j(i)$ such that $U_i(s_i^{j(i)}, m_i^{j(i)}) \ge U_i(s_i^j, m_i^j)$ for all j. Let us set $(s_i, z_i) = (s_i^{j(i)}, m_i^{j(i)})$.

At fixed j, the user's problem has the following form: $U_i(s, z) \to \max$ subject to conditions (3) and $s \in [0; Q_i]$, $m \ge 0$. In [1, §2.3.2] it was shown that one can calculate the optimal pair $(s_i^j(p_j), m_i^j(p_j))$ as follows:

If $p_j = 0$, then $(s_i^j(p_j), m_i^j(p_j)) = (Q_i, B_i)$. If $p_j \neq 0$, then

$$s_i^j(p_j) = \begin{cases} Q_i, & p_j \in [0; k_i B_i/Q_i], \\ k_i B_i/p_j, & p_j > k_i B_i/Q_i, \end{cases} \tag{5}$$

where $k_i = a_i/(a_i + b_i)$. Respectively,

$$m_i^j(p_j) = B_i - s_i^j(p_j). \tag{6}$$

A maximum utility at the price p_j is $\varphi_i(p_j) = U_i(s_i^j(p_j), m_i^j(p_j))$:

$$\varphi_i(p_j) = \begin{cases} Q_i^{a_i}(B_i - p_j Q_i)^{b_i}, & p_j \in [0; k_i B_i/Q_i], \\ (k_i B_i/p_j)^{a_i}((1-k_i)B_i)^{b_i}, & p_j > k_i B_i/Q_i. \end{cases} \tag{7}$$

The function $\varphi_i(\cdot)$ is continuous and monotonously decreases. Hence, the user will choose a processor with the minimal price.

3. The local equilibrium

For any set $N(j)$ and any $p \geq 0$ let us set $D(p) = \sum_{i \in N(j)} s_i^j(p)$.

Definition. The processor j is in the local equilibrium with price p and a group of users $N(j)$ if $D(p) \in S_j(p)$.

It is easily seen that the functions $s_i^j(\cdot)$ are non-increasing. In the further reasoning, we always assume $i \in N(j)$. Set $L(i) = k_i B_i/Q_i$, and let $\lambda(1) < \ldots < \lambda(r)$ be a sequence of all pairwise distinct values of $L(i)$. According to (5)

$$D(p) = \sum_{\{i|p<L(i)\}} Q_i + \sum_{\{i|p \geq L(i)\}} k_i B_i/p. \tag{8}$$

In [1, Theorem 2] it was proven that the local equilibrium always exists, and the equilibrium distribution of the processor capacity among users (within a unit period) is uniquely determined. In the same paper, an algorithm is proposed for calculation of equilibrium for each processor j. In a nontrivial case, when $D(c_j) > V_j$ (the processor is overloaded), the algorithm includes the calculation of

$$p_{jk} = \frac{\sum_{\{i|L(i) \leq \lambda(k)\}} k_i B_i}{V_j - \sum_{\{i|L(i) \geq \lambda(k+1)\}} Q_i} \tag{9}$$

for each $k \in \{1, \ldots, r\}$. The proof of the above-mentioned theorem shows that in the case in question the equilibrium price exists and is unique. Therefore, there exists a unique value k such that $p_{jk} \in [\lambda(k); \lambda(k+1))$. This p_{jk} is just the desired equilibrium price p^*.

The computational complexity of the described mechanism is proportional to r, and r is no more than the number of jobs. Below we propose another algorithm of the same complexity for calculating the equilibrium price. This second algorithm realizes a simple idea of the proportional distribution. It is more convenient in certain cases.

4. An algorithm of CPU capacity distribution

Construction of an equilibrium in the case of $D(c_j) \le V_j$ (the processor j is not overloaded) is trivial. So, consider the case

$$D(c_j) > V_j. \tag{10}$$

From now, we fix j, i.e., choose one of the processors, and omit the index j. Let us assume that the users with numbers $i \in N$ have chosen this processor for the next unit period. At the preliminary Step 0, we set $I_1 = N$ (a set of jobs being active at Step 1), $W_1 = V$ (the number of unallocated CPU cycles) and $q_i = 0$ (the preliminary distribution of CPU cycles) for all $i \in N$.

Step $k > 0$. At Step $k-1$, the set I_k and the value W_k were determined. For all $i \in I_k$, set $\alpha_i^k = k_i B_i / \sum_{r \in I_k} k_r B_r$. At the current step, the algorithm distributes W_k among the active jobs $i \in I_k$ as follows.

Calculate a preliminary distribution $x_i^k = \alpha_i^k W_k$, $i \in I_k$. Define $J_k = \{i \in I_k \mid x_j^k \ge Q_i\}$ (a set of those jobs, for which the number of the preliminary allocated CPU cycles covers the demand), $I_{k+1} = I_k \setminus J_k = \{i \in I_k \mid x_i^k < Q_i\}$ (a set of jobs being active at the following step), $W_{k+1} = W_k - Q_i$. The job $i \in N$ receives the CPU capacity

$$q_i^k = \begin{cases} Q_i, & i \in J_k, \\ x_i^k, & i \in I_{k+1}. \end{cases} \tag{11}$$

The procedure terminates if $x_i^k \le Q_i$ for all $i \in I_k$.

For each k, let us define

$$p_k = \sum_{i \in I_k} k_i B_i / W_k \tag{12}$$

(the price of one cycle at the step k).

Note that at Step k, we distribute W_k (the unallocated cycles) among active jobs $i \in I_k$ in proportion to the values $k_i B_i$, where $k_i B_i$, according to (5), is the maximal resource cost acceptable for the user i.

Theorem 1. *The described algorithm of the CPU capacity distribution is finite.*

Proof. Let $M_s = \{i \in I_s \mid x_i^s > Q_i\}$. The condition of stopping the algorithm may be written down in the form $M_k = \emptyset$. Let us prove that $M_k = \emptyset$ at some step $k \le m$, where $m = |N|$ (the number of those users who have selected the considered processor for the next unit period). Indeed, for $s \ge 1$, we have $M_s \subseteq J_s \subseteq I_s$ and $I_{s+1} = I_s \setminus J_s$. If $M_s \ne \emptyset$, then $I_{s+1} \subset I_s$. Therefore, $N = I_1 \supset I_2 \supset \ldots$, and so the sequence $I_1, I_2, \ldots$ contains no more than $|N|$ elements. □

Lemma. *A sequence of prices p_k is non-increasing.*

Proof. Let us write the expression for p_{k+1}:

$$p_{k+1} = \frac{\sum_{i\in I_{k+1}} k_i B_i}{W_{k+1}} = \frac{\sum_{i\in I_k \setminus J_k} k_i B_i}{W_k - \sum_{i\in J_k} Q_i} = \frac{\sum_{i\in I_k} k_i B_i - \sum_{i\in J_k} k_i B_i}{W_k - \sum_{i\in J_k} Q_i}. \quad (13)$$

It follows from (12) and (13) that $p_{k+1} \le p_k$ is equivalent to the inequality

$$\frac{\sum_{i\in J_k} k_i B_i}{\sum_{i\in I_k} k_i B_i} W_k \ge \sum_{i\in J_k} Q_i. \quad (14)$$

By definition of the set J_k, for each $i \in J_k$ we have

$$\frac{k_i B_i}{\sum_{i\in I_k} k_i B_i} W_k = x_i^k \ge Q_i. \quad (15)$$

Summing these inequalities for all $i \in J_k$ we obtain (14). Consequently, $p_{k+1} \le p_k$. □

Denote by τ the number of the last step of the above algorithm (this step exists according to Theorem 1). Let N be a set of numbers of all users, who have selected the considered processor for the next unit period, $J^* = \cup_{k=1}^{\tau} J_k$, $W^* = V - \sum_{i\in J^*} Q_i$, $I^* = N \setminus J^*$, $p^* = p_\tau = \sum_{i\in I^*} k_i B_i / W^*$.

Theorem 2. *The local equilibrium price is equal to p^*.*

Proof. The user i's demand $s_i^j(p_j)$ for the capacity of the processor j within the considered unit period is described by formula (5). Let us fix i. Two cases are possible at the last step τ of the above algorithm of the CPU capacity distribution.

1) The user i obtains Q_i cycles at the step τ. Then $i \in J_k$ at some step $k \le \tau$ and inequality (15) holds, i.e., $p_k \le k_i B_i / Q_i$. From Lemma 1 it follows that $p^* \le p_k$ for all $k \le \tau$. Hence, $p^* \le k_i B_i / Q_i$, and $s_i^j(p^*) = Q_i$ by (5).

2) The user i obtains $k_i B_i/p^* < Q_i$ CPU cycles at the step τ. Then $i \notin J_\tau$ and (15) with $k = \tau$ is not true, i.e., $p^* = p_\tau > k_i B_i/Q_i$ and $s_i^j(p^*) = k_i B_i/p^*$ by (5).

Thus, the CPU capacity allocated to a user by the algorithm is equal to the demand of this user under the price p^*, and the algorithm ensures $\sum_i s_i^j(p^*) = V_j$ (the full processor capacity is distributed). Function (8) is non-increasing, therefore from (10) follows $p^* > c_j$; then the supply is V_j according to (4). Hence, the total demand equals the supply. Therefore, p^* is an equilibrium price. □

5. Equilibrium in a multiprocessor system

It is natural to say that a multiprocessor system is in equilibrium if

- the local equilibria are established on all processors, and
- each user is "assigned" to the same processor he would choose under the prices of these equilibria.

At the end of Section 2, we noted that it is profitable for a user to use a processor with the minimal price for his job. Let $J(\boldsymbol{p}) = \{j \mid \forall k(p_j \le p_k)\}$ be a set of numbers of all such processors. The user i's demand is described by the set of pairs

$$D_i(\boldsymbol{p}) = \{(j, s_i^j(p_j)) \mid j \in J(\boldsymbol{p})\}. \tag{16}$$

Definition. A price vector $\boldsymbol{p} = (p_j)_j$ and a partitioning $(N(j))_j$ of a set of all jobs is an equilibrium if each processor j is in the local equilibrium with the price p_j and the group of users $N(j)$, and $(j, s_i^j(p_j)) \in D_i(\boldsymbol{p})$ for all $i \in N(j)$.

It was shown in [1, §2.4] that being in equilibrium, the prices p_j coincide for all j such that $N(j) \neq \emptyset$. That is, all jobs are served at the uniform equilibrium price p^*. In addition, from $N(j) = \emptyset$ it follows that $p^* \le p_j \le c_j$. This means that in equilibrium the processors with high operating costs are not used if all jobs can be served at a lower price. The fact is, we do not know the existence conditions for equilibria in multiprocessor systems.

6. The Nash equilibrium

Functioning of a multiprocessor system can be described as follows.

At the moment when a new job (with number s) appears, the other jobs are distributed among processors yet. Let $N(j)$ be a set of all jobs assigned to the processor j and being not completed during the current time unit. The job s chooses a processor and capacity for the next time unit. The rule of choosing follows from the fact that the function $\varphi_i(p_j)$ defined at the

end of Section 2 is decreasing (see (7)): the job prefers the processor j, for which the price of the local equilibrium with a group of users $N(j) \cup \{s\}$ (if the job j chooses the processor j) is minimal. It may be supposed that the system defines this price itself (for example, by application of the mechanism described in Section 4) and, respectively, allocates the job (such an allocation does not contradict to the user's preferences). Nevertheless, after the job s is allocated to the processor j with a new price p_j of the local equilibrium, it may turn out that some job $i \in N(j)$ "wants" to move to the processor $k \neq j$, because the local equilibrium price for the processor k with a set of jobs $N(k) \cup \{i\}$ is less than p_j.

To formulate the process of jobs reallocation among processors (a moving process), in any case, the equilibrium price should be uniquely determined. Let $P_j(M)$ be a set of local equilibrium prices for the processor j with the set of jobs M. It is proved in [1, Theorem 2] that $P_j(M) \neq \emptyset$ for any M and the equilibrium price is ambiguously determined only in the following two cases:

(A) If $M = \emptyset$ and $c_j > 0$, then $P_j(M) = [0; c_j]$.

(B) If $\sum_{i \in M} s_i^j(c_j) = V_j$ and $c_j < \min\{k_i B_i / Q_i \mid i \in M\} = \lambda$, then $P_j(M) = [c_j; \lambda]$.

To check, whether a job $i \in N(j)$ passes to the processor $k \neq j$, it is necessary to compare $p_j(N(j))$ and $p_k(N(k) \cup \{i\})$. In this case, $N(j) \neq \emptyset$ and $N(k) \cup \{i\} \neq \emptyset$, therefore we do not need to consider the case (A). Nevertheless, for completeness assume $p_j(\emptyset) = c_j$. In the case (B), we choose the minimal equilibrium price c_j, since it stimulates reallocation of jobs to those processors, which are less loaded and have lower starting prices. In other words, in the case $c_j \in P_j(M)$, we set $p_j(M) = c_j$. Using this definition of $p_j(M)$ we prove the finiteness of the moving process.

Theorem 3. *If the equilibrium price at each processor j for any set of jobs M takes the value $p_j(M)$, and a set of all jobs is invariable, then the moving process is finite.*

Proof. Assume that jobs are distributed for the nearest unit period. All new jobs have come to the system and all completed jobs have left it before starting the moving process. The number of jobs allocations among processors is finite. A uniquely defined vector of the equilibrium prices $p = (p_j(N(j))_j$ corresponds to each distribution $(N(j))_j$. Therefore, a set of possible equilibrium prices is finite. Let us arrange these values in increasing order: $\pi_1 < \pi_2 < \ldots < \pi_{r-1} < \pi r$. Let m_k be a number of all jobs using the processors with prices of the local equilibrium higher than π_k. Let us consider the processor a with the price of the local equilibrium π_k and a set of users $N(a)$. We assume the job s to move from the processor a to another

processor b. Then $p_b(N(b) \setminus \{s\}) < \pi_k$ (otherwise the job s would not move to the processor b). After the movement of the job s, the price for services of the processor a will decrease to adapt to the set of jobs $N(a) \setminus \{s\}$. It is clear that $p_a(N(a) \setminus \{s\}) < \pi_k$ if $\pi_k > c_a$, and $p_a(N(a) \setminus \{s\}) = \pi_k$ if $\pi_k = c_a$. In any case, the number of users served at prices not lower than π_k will decrease. Therefore, no more than mr reallocations will occur, where m is the number of jobs, and r is the number of equilibrium prices $p_j(M)$ (for all M). □

The job moving process results in the final distribution when none of the users want to change a processor. This distribution is not necessarily an equilibrium one in terms of the definition given in Section 5. But it is a Nash equilibrium in the following game form.

The players are m users and n suppliers. Each user wants to execute one job, each supplier has one processor. Let N be a set (of numbers) of all players, $|N| = m + n$. The user $i \in \{1, \ldots, m\}$ reports to the system the volume Q_i of his job and willingness to pay $k_i B_i$; the supplier $j \in \{m+1, \ldots, m+n\}$ reports the processor capacity V_j (cycles per unit of time) and the reservation price c_j. User i chooses a number $j(i) \in \{m+1, \ldots, m+n\}$ of the processor, which he wants to use within the nearest unit period; supplier j chooses the capacity $y_j \in [0; V_j]$ he would like to place at the disposal of the system for the same period. Thus, the strategy z_k of the player k is $j(k)$, if $1 \le k \le m$, and y_k if $m < k \le m + n$. The situation or the strategy profile is a vector $\boldsymbol{z} = (z_1, \ldots, z_n)$ including one strategy for each player. Given the strategy profile, the users pay for resources at the prices of the corresponding local equilibria (see Section 3).

Now let us define the payoff functions. Given the strategy profile $\boldsymbol{z}$, let $N_j(\boldsymbol{z}) = \{i \mid 1 \le i \le m,\ z_i = j\}$ be a set of all users choosing the processor j in the situation $\boldsymbol{z}$. For each $j \in \{m+1, \ldots, m+n\}$, the system calculates the local equilibrium price $p_j = p_j(N_j(\boldsymbol{z})) \ge c_j$ for the maximal possible supply V_j and the group of users $N_j(\boldsymbol{z})$. If the demand for the processor j capacity under the price p_j is not equal to the supply y_j (corresponding to situation $\boldsymbol{z}$), then the gains of supplier j and all the users $i \in N_j(\boldsymbol{z})$ are equal to zero, $F_j(z) = F_i(\boldsymbol{z}) = 0$. Otherwise, according to (1), (7), the supplier's gain equals his profit, $F_j(\boldsymbol{z}) = (p_j - c_j)y_j$, and the user's gain equals the obtained utility, $F_i(\boldsymbol{z}) = U_i(s_i^j(p_j), B_i - p_j s_i^j(p_j))$ for $i \in N_j(\boldsymbol{z})$.

Let us denote by $\boldsymbol{z}_{-i}$ the collection of all players', except the player i, strategies in the situation $\boldsymbol{z}$. Then $\boldsymbol{z} = (z_i, \boldsymbol{z}_{-i})$. If in the situation $\boldsymbol{z}^0$, there are $F_i(z_i^0, \boldsymbol{z}_{-i}^0) \ge F_i(z_i, \boldsymbol{z}_{-i}^0)$ for each i and any strategy z_i of the player i, then $\boldsymbol{z}^0$ is a Nash equilibrium.

To each step of the moving process there corresponds some distribution $(N(j))_j$ of the jobs among processors. In turn, each distribution $(N(j))_j$ creates the strategy profile $\boldsymbol{z}$ in the considered game as follows.

For all i $(1 \leq i \leq m)$ we set $z_i = j$, if $i \in N(j)$. Then, knowing the sets $N(j)$, we calculate the local equilibrium prices $p_j = p_j(N(j))$ for all processors. At these prices each supplier j $(m < j \leq m + n)$ chooses the supply $S_j(p_j)$ according to (4); we set $z_j = S_j(p_j)$.

Theorem 4. *The strategy profile that corresponds (as described above) to the final distribution of the moving process is a Nash equilibrium in the considered game.*

Proof. Let $(N(j))_j$ be the final distribution of the moving process and $\boldsymbol{z}$ is the corresponding strategy profile. Let us show that it is not profitable for each player i to deviate from the strategy z_i if all other players $k \neq i$ choose strategies z_k. Let $p_j = p_j(N(j))$ be the local equilibrium price for the processor j. It follows from the definition of $p_j(M)$ (in the beginning of this section) that $p_j \geq c_j$.

If $p_j = c_j$, then all possible values of y_j give zero profit to the supplier j, and rhus z_j is one of his best responses to the other player's strategies. If $p_j > c_j$, then supply $y_j = V_j$ corresponds to the local equilibrium for the processor j, so, $z_j = V_j$; just this strategy will be chosen by supplier j, since it maximizes his profit, see (1). Therefore, the supplier is not interested in deviation from the strategy z_j.

Let $j(i)$ be the number of a processor, for which the job i is assigned in the considered final distribution. The user i's payoff function $F_i(\boldsymbol{z})$ is his indirect utility function, it depends only on the equilibrium price for the processor $j(i)$ and does not increase in price (see, e.g. [3, proposition 3.D.3]). If the moving process is completed, then the reallocation of any job i subject to an invariable allocation of all other jobs will not result in a decrease of the equilibrium price, by which the job i will be served. Therefore, the player i's payoff function will not increase, and so the consumer i is not interested in deviation from the strategy $z_i = j(i)$.

Therefore, for each player k, the strategy z_k is one of the best responses to the other players' strategy profile, so, $\boldsymbol{z}$ is a Nash equilibrium. □

The statement, converse to Theorem 4, is evident: if the moving process starts with the distribution of jobs corresponding to a Nash equilibrium, then no movements will occur and the initial distribution is also the final one.

7. Conclusion

The equilibrium allocation in a single-processor system gives, in fact, a schedule of the jobs execution. In a multiprocessor system, it is possible to use the local equilibrium prices for an expedient time distribution by, for

example, the moving process described in this paper. This process results in a Nash equilibrium.

References

[1] Bredikhin S.V., Khutoretskiy A.B., Savchenko I.Yu., Vyalkov I.A. Two models of adjusting for distribution of computational resources // Siberian J. Industrial Math. — 2006. — Vol. IX, No. 1 (25). — P. 28–46 (In Russian).

[2] Bredin J., Kotz D., Rus D. Utility Driven Mobile-Agent Scheduling. — Hanover (Germany): Dartmouth College, 1998. — (Technical Report; PCS-TR98-331).

[3] Mas-Colell A., Whinston M.D., Green J.R. Microeconomic Theory. — Oxford (USA): Oxford Univ. Press, 1995.