Bai-Da ZHANG[1], Yu-Hua TANG[1], Jun-Jie WU[1], Shuai XU[2]

National laboratory for Parallel and Distributed Processing, School of Computer, National University of Defense Technology (1)
Department of Information Engineering, Academy of Armored Force Engineering (2)

Opportunity Cost Model of the Task Scheduling in Heterogeneous Systems

Abstract. *With the micro-electronics technology has encountered a bottleneck, adding heterogeneous core has become the primary means of increasing processor speed. However, how to assign heterogeneous processor to maximize the performance becomes an urgent problem. The problem has been proved to be NP-complete problem, i.e. it cannot find the optimal solution in polynomial time. This article draws on the idea of the economy, given the concept of opportunity cost in heterogeneous systems, and were analyzed by the opportunity cost model for task scheduling on heterogeneous systems. On this basis, draw the basic principles of a number of task scheduling. Theory and simulation results show that the task assignment algorithm to achieve the desired performance.*

Streszczenie. *Przy projektowaniu układów mikroelektronicznych niejednorodny rdzeń umożliwia zwiększenie szybkości procesora. W artykule przedstawiono ideę uwzględnienia modelu kosztów do projektowania kolejności zadań w systemie niejednorodnym. (**Możliwość wykorzystania modelu kosztów w projektowaniu kolejności zadań w systemach niejednorodnych**)*

Keywords: Opportunity Cost, Comparative Advantage, Heterogeneous Systems, Task Scheduling
Słowa kluczowe: Koszt alternatywny; atutach; Różnica gou System.

Introduction

As feature sizes continue to decrease, the research of processor encounter a problem that performance is limited by increasing the processing capacity of a single processor core, or just improve the number of cores on the system. Thus, heterogeneous systems become the mainstream and heterogeneous systems achieve good performance in many applications.

With the widening range of application of computers, distributed and heterogeneous systems (DHS, the Distributed Heterogeneous System) are becoming an effective tool to solve complex application problems. A set of heterogeneous computers is used to solve a specific task in order to obtain the better performance. There may be have hundreds of tasks in the job queue of a computer system, therefore how to allocate processor time between processes, is undoubtedly an important issue. Traditional scheduling algorithm consists of first come first served, shortest job priority. However, in heterogeneous systems, the processor allocation to new problems, that is, the same task on different processors have very different execution time, so that the processor will be allocated directly related to the operation of the system as a whole time and efficiency [1-5]. The processor allocation problem, also called processor scheduling. Task scheduling problem in the DHS, to play the parallel performance of the system and maintain the load balancing is very important. The problem has been proved to be NP-complete problem, i.e. it cannot find the optimal solution in polynomial time. Therefore, tireless efforts are put in to design a scheduling algorithm which can get a better solution with limited cost. Commonly used method is heuristic and random search of approximate methods [3,5-9].

A good scheduling algorithm should be considered, which may contain features following [10-14]:

- **Resource utilization rate** - utilization of the CPU or other resources as high as possible and can work in parallel.
- **Response time** - the interactive user response time as small as possible, or as soon as possible to deal with real-time tasks.
- **Turnaround time** - batch job submission to the system to the job is completed the results obtained up to this time interval that the job turnaround time, job turnaround time, or average job turnaround time as short as possible.
- **Throughput** - the unit of time to deal with the number of jobs as much as possible.
- **Fairness** - to ensure that each user for each process to obtain a reasonable share of the CPU or other resources sharing.

In order to studying processor task scheduling problem of heterogeneous systems, we introduce the idea of economics, which contains opportunity cost and comparative advantage, which is trade can improve economic status of all members [15,16]. Corresponding to the heterogeneous systems, which mean a reasonable task scheduling, the throughput of each processor can improve.

The creative research of this paper is the concept of opportunity cost in heterogeneous systems and applications to achieve at the task scheduling. The paper is organized as follows, Section 1 is introduction. Section 2 discusses the motivation of the paper. Scheduling algorithm is given in Section 3. Section 4 is the proof. Section5 describes the simulation experiments. Finally, we draw conclusions and discuss the future works in Section 6.

Motivations

First, task scheduling will be described formally. For a given task queue $T_1, T_2, T_3, ..., T_n$, task scheduling is to determine the time when task T_i is executed for the single processor. In homogeneous system, the processor is also need to be assigned when the same task queue is given. However, when considering heterogeneous system for the same scheduling problem, everything is changed. For example, there are a task queue and two processors, namely U_1 and U_2. Each task execution time on different processors is shown in Table 1:

Table 1. Task queue and execution time

Execution Time	T_1	T_2	T_3	T_4	T_5
U_1	5	17	27	21	17
U_2	3	21	11	32	15

If the first task of the queue is assign to the idle processor, then the execution is shown in Figure 1, i.e. tasks T_1, T_3, T_5 executed in processor U_1, while tasks T_2 and T_4 executed in processor U_2. Total computation time of U_1 was 49 while total computation time of U_2 was 53.

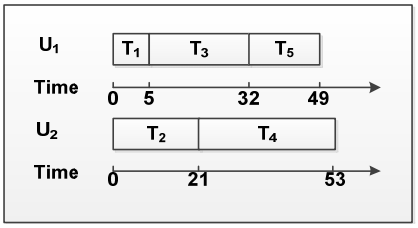

Fig.1. Scheduling method 1

If we adopt the scheduling policy shown in Figure 2, tasks T_1, T_3, T_5 executed in processor U_2, while tasks T_2 and T_4 executed in processor U_1, then total computation time of U_1 was 38 while total computation time of U_2 was 29. This is a surprising result! Because we just changed task allocation strategy, but the execution time of all processors have reduced!

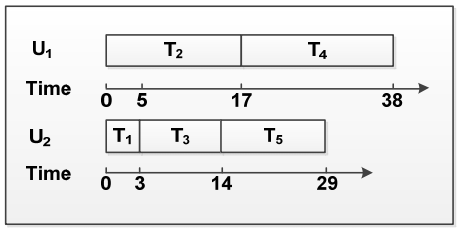

Fig.2. Scheduling method 2

The purpose of this paper is to introduce the approach of economics to solve the optimization problem of task scheduling in heterogeneous systems, and making the sum of the processor execution time decreases. It is to say that transplanting the approach how to organize production in economics to the task allocation in heterogeneous computing systems, which can be convenient scheduling arrangements for a task queue and can maximize performance of systems.

Solutions

In order to facilitate comparison, some index will be given first to evaluate scheduling strategies.

Definition 1(total execution time): Supposing a group of tasks T_i and a group of processors U_j, the total execution time is $T = \Sigma t_j$, where t_j is the execution time of processor j.

This is an index for heterogeneous computing systems. It is a function of task scheduling strategy. Take Figure 1 and Figure 2 as example, different scheduling strategy will lead to different total execution time. Smaller the index is, the better scheduling strategy is.

The following will give a simple example; it will be used to illustrate some concept.

There are two processors, namely U_1 and U_2 (we can suppose they are a GPU and a CPU respectively), and two tasks, namely A and B (for ease of understanding, imaging feature extraction and pattern recognition). Their execution time is shown in Table 2

Table 2. Execution Time (unit: seconds)

Execution Time	*A*	*B*
U_1	1	2
U_2	2	6

It can be seen from the table, U_2 can execute two tasks more quickly than processor U_1. Then what strategy should be used to schedule processors to maximize the performance of the system? Next, scheduling strategy will be introduced. This can be disassembled into two parts, i.e. constructing opportunity cost table and scheduling.

(1). Constructing Opportunity Cost Table

Definition 2 (Opportunity Cost): The opportunity cost of processor U run task A is the number of task B that U can execute in the time when a task A is executing.

For the example given in Table 2, we can construct the opportunity cost table shown in Table 3. Opportunity cost of U_1 run task A, for example, is 1/2 B. Because the time executing a task A is 1 second, and the time executing a task B is 2 seconds.

Table 3. Opportunity cost table of Table 2

Opportunity Cost	*A*	*B*
U_1	0.5 *B*	2 *A*
U_2	0.33 *B*	3 *A*

Definition 3 (Comparative Advantage): U_1 have comparative advantage when it execute task A compared with U_2 execute task B, if and only if $C_1<C_2$, where C_1 and C_2 are the opportunity cost that processor U_1 run task A and processor U_2 run task A respectively.

For example, in Table 2, processor U_1 running task B has comparative advantage. Because the opportunity cost of processor U_1 running task B is 2 task A, but processor U_2 is 3 task B. Similarly, processor U_2 running task A has comparative advantage.

It can be seen from Table 3 that the opportunity cost of two task of the same processor is reciprocal. Thus it can be inferred that if processor U_1 running task A has comparative advantage, then processor U_2 running task B must have comparative advantage. That is, each processor can contribute to performance of system.

(2) Scheduling Method

Scheduling method is based on the following analysis.

Definition 4(possible boundary of computing): Supposing only processor U_1 existing in the system, then 60 task A or 30 task B *or* some combinations of task A and B can be completed within 60 seconds time. This can be shown in Figure 3 as the upper diagonal line, named as possible boundary of computing. Similarly, U_2's possible boundary of computing can be drawn in Figure 3.

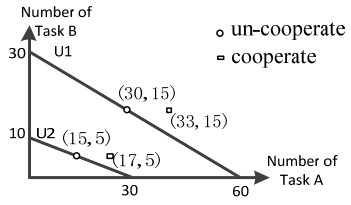

Fig.3. Possible boundary of computing of processors

Processor U_1 can only reach the point in the triangle that surrounded by A axis, B axis and U_1's possible boundary of computing when processor U2 is absence. However, if processor U_1 and U_2 can work together to finish each other's task, it is possible to break through this boundary. For example, processor U_1 need to execute 30 task A and 15 task B, and processor U_2 need to perform 15 task A and 5 task B. Because U_1 has a comparative advantage to execute task B, then U_1 help U_2 to execute 5 task B, and U_2 help U_1 to execute 13 task A. The total execution time is shortened. The total time of processor U_1 reduced from 60 to 57 and the total time of processor U_2 reduced from 60 to 56.

It can be seen from the above analysis, two different processors, in the case of mutual cooperation, their own are able to accomplish more tasks. This conclusion is also true for more processors and tasks. Following section will give the proof that relation of comparative advantage is a full

order relation, which means that we can assign tasks according to comparative advantage relations.

Table 4. Time and number of tasks under different scheduler

	number of Task *A*		number of Task *B*		total time	
	un-coop erate	coo pera te	un-coop erate	coo per ate	un-coop erate	coop erate
U_1	30	17	15	20	60	57
U_2	15	28	5	0	60	56

Proving

In this section, we'll give out the proving that comparative advantage relations are full order relation. Because any pair of element of a set that have total ordering relation are comparable and order can be used as the guideline to scheduling.

First, a quaternary relation *R(U,V,A,B)* is given, where U and V belong to the processor set, A and B belong to the task set. The value *R(U, V, A, B)*=1 if and only if the processor *U* perform task *A* has a comparative advantage to processor *V* execute task *B*, and *R(U, V, A, B)=0* for others.

we define other two dualistic relations, which are $S_{UV}(A, B)=R(U,V,A,B)$ and $T_{AB}(U,V)= R(U,V,A,B)$. And the proving process will be divided into two parts, namely relation S_{UV} is full order relation and relation T_{AB} is full order relation.

Full order relation can be proved directly by its definition. A relation is full order relation means that if we denote this relation as '≤', then following statements will be true for any a, b and c:

- If a ≤ b and b ≤ a then a = b (antisymmetry)
- If a ≤ b and b ≤ c then a ≤ c (transitivity)
- a ≤ b or b ≤ a (completeness)

4.1 S_{UV} is full order relation

Completeness and anti-symmetry of S_{UV} can be gotten from the definition of S_{UV} directly. Transitivity of the relation S_{UV} can be proved as the following: Supposing *A*, *B*, and *C* are tasks, *U* and *V* are processors, then opportunity cost table can be constructed as Table 5 by the definition, where X_{ij} is the number of task *j* that processor *U* can execute in the time of executing a task *i*, Y_{ij} is the number of task *j* that processor U can execute in the time of executing a task *i*.

Table 5. Opportunity cost of relation S_{UV}

Opportunity Cost	*A*	*B*	*C*
U	X_{AB}/X_{AC}	X_{BA}/X_{BC}	X_{CA}/X_{CB}
V	Y_{AB}/Y_{AC}	Y_{BA}/Y_{BC}	Y_{CA}/Y_{CB}

Supposing $S_{UV}(A, B)$ = 1, and $S_{UV}(B, C)$ = 1, then inequalities $X_{AB} < Y_{AB}$ and $X_{BC} < Y_{BC}$ can be gotten by the definition of relation S_{UV} directly. It is easy to know that $X_{AC} = X_{AB} * X_{BC}$, $Y_{AC} = Y_{AB} * Y_{BC}$, then inequalities $X_{AC}<Y_{AC}$ can be inferred, i.e. $S_{UV}(A, C)$ = 1. The transitivity of the relationship S_{UV} is proved.

In summary, the relationship $S_{UV}(A, B)$ is a total order.

4.2 T_{AB} is full order relation

Completeness and anti-symmetry of T_{AB} can be gotten from the definition of T_{AB} directly. Transitivity of the relation T_{AB}, can be proved as the following. U_1, U_2 and U_3 are processors that be chosen randomly. *A*, *B* are tasks. Then opportunity cost table can be constructed in Table 6 according to the definition, where. X_{ij}, Y_{ij}, Z_{ij} are the number of task *j* that processor U_1, U_2, U_3 can execute respectively in the time that a task *i* is executed.

Table 6. Opportunity cost of relationship T_{AB}

Opportunity cost	*A*	*B*
U_1	X_{AB}	X_{BA}
U_2	Y_{AB}	Y_{BA}
U_3	Z_{AB}	Z_{BA}

If $T_{AB}(U_1, U_2)$ = 1 and $T_{AB}(U_2, U_3)$ = 1, then inequalities $X_{AB} < Y_{AB}$ and $Y_{AB} < Z_{AB}$ can be gotten by the definition of T_{AB}. Inequality $X_{AB}<Z_{AB}$ can be inferred logically, i.e. $T_{AB}(U_1, U_3)$ = 1. The transitivity of the relationship T_{AB} is proved.

In summary, the relationship $T_{AB}(U, V)$ is a total order.

Experiments

In this paper, validation of the task scheduling algorithm presented is done by simulation experiments The process can be described as below, the system contains five different processors U_i (i = 1, 2, ..., 5). And 8 tasks, which are *A*, *B*, ..., *H* will be executed on this system. The execution time of each task on different processors is shown in Table 7. The method used by the task scheduling is the same for any other value.

Table 7. Task execution time on each processor

Execution Time	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
U_1	45	5	10	10	35	40	40	45
U_2	50	15	50	25	5	40	5	35
U_3	10	30	50	50	45	20	15	20
U_4	50	50	25	40	50	35	5	50
U_5	35	50	45	50	35	10	5	5

For any pair of tasks, such as *A* and *B*, and all processors U_i, opportunity costs table can be constructed According to the definition directly as Figure 4 shows. According to the opportunity cost table of task *A*, we can order all processor U_i. The result is $U_1<U_2<U_4<U_5<U_3$. From the result, U_3 is the best processor to execute task *A*. When there is a task *A* waiting to execute and U_3 is idle, the task should be executed on processor U_3.

Opportunity Cost	A/B
U_1	9
U_2	10/3
U_3	1/3
U_4	1
U_5	7/10

Priority	U_1	U_2	U_3	U_4	U_5
U_1	*	>	>	>	>
U_2	<	*	>	>	>
U_3	<	<	*	<	<
U_4	<	<	>	*	>
U_5	<	<	>	<	*
$U_1<U_2<U_4<U_5<U_3$					

Fig.4. Opportunity cost and processors ordering

Similarly, priority order can be constructed for any other task pairs. Such a task, the processor will have a priority ranking, and thus the scheduling of all tasks can be achieved.

In order to compare the performance of the proposed algorithm, the two scheduling strategies is implemented. They are as follows:

(1) First-In First-out strategy (FIFO for short): to maintain a task arrives in the queue sorted by arrival time, processor idle scheduling task execution queue first.

(2) Opportunity Cost Model strategy (OCM for short): A queue is maintained for each task. When a processor is idle, a task which it can do better is executed.

Figure 5 shows the generated task arrival sequence comparison chart for two methods mentioned above. Task to generate a completely random manner, which includes two meanings: the type and number of tasks is random, as well as interval between each batch task arrival is random. The result shows that 19.6% of execution time is saved when OCM strategy is used.

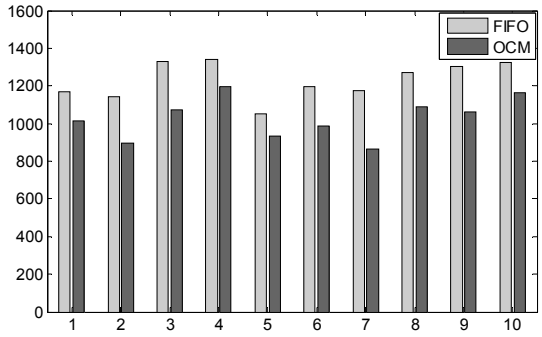

Fig.5. Total execution time of the two scheduling strategies

Conclusions

Heterogeneous system is now the direction of development, but the scheduling of heterogeneous systems lack the guidance of the mathematical model. This article introduced the concept of opportunity cost and comparative advantage to task scheduling of heterogeneous systems. The theoretical analysis and experimental verification shows that heterogeneous systems can improve the efficiency of each processor by giving priority to the works they are good at. Through this analysis, we can see that the concept of opportunity cost of task scheduling is helpful.

Acknowledgement

This work was supported in part by the Young Scientists Fund of the National Natural Science Foundation of China (Grant Nos 61003082, 60903059), the National Natural Science Foundation of China (Grant Nos 60873014), and the Science Fund for Creative Research Groups of the National Natural Science Foundation of China (Grant Nos 60921062).

REFERENCES

[1] Dj.M. Maric, P.F. Meier and S.K. Estreicher: Mater. Sci. Forum Vol. 83-87 (1992), p. 119

[2] M.A. Green: High Efficiency Silicon Solar Cells (Trans Tech Publications, Switzerland 1987).

[3] Y. Mishing, in: Diffusion Processes in Advanced Technological Materials, edtied by D. Gupta Noyes Publications/William Andrew Publising, Norwich, NY (2004), in press.

[4] G. Henkelman, G.Johannesson and H. Jónsson, in: Theoretical Methods in Condencsed Phase Chemistry, edited by S.D. Schwartz, volume 5 of Progress in Theoretical Chemistry and Physics, chapter, 10, Kluwer Academic Publishers (2000).

[5] R.J. Ong, J.T. Dawley and P.G. Clem: submitted to Journal of Materials Research (2003)

[6] P.G. Clem, M. Rodriguez, J.A. Voigt and C.S. Ashley, U.S. Patent 6,231,666. (2001)

[7] Information on http://www.weld.labs.gov.cn

[1] Silberschatz A, Galvin P B, Gagne G, et al. Operating system concepts, 4[M]. [S.l.]: Addison-Wesley, 1998.

[2] Tanenbaume, S A. Modern operating systems, 2[M]. [S.l.]: Prentice Hall New Jersey, 1992.

[3] Lo, M V. Heuristic algorithms for task assignment in distributed systems[J]. Computers, IEEE Transactions on, 1988, 37(11): 1384-1397.

[4] Casavant T L, Kuhl J G. A taxonomy of scheduling in general-purpose distributed computing systems[J]. Software Engineering, IEEE Transactions on, 1988, 14(2): 141-154.

[5] Shen C C, Tsai W H. A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion[J]. Computers, IEEE Transactions on, 1985, 100(3): 197-203.

[6] Grimshaw A S, Weissman J B, West E A, et al. Metasystems: an approach combining parallel processing and heterogeneous distributed computing systems[J]. Journal of Parallel and Distributed Computing, 1994, 21(3): 257-270.

[7] Chafil H, Devito Z, Moors A, et al. Language virtualization for heterogeneous parallel computing[Z]. [S.l.]: [s.n.], 2010: 835-847.

[8] Buyya R, others. High performance cluster computing: architectures and systems (volume 1)[J]. Prentice Hall, Upper Saddleriver, Nj, USA, 1999, 1(期缺失): 999.

[9] Sunderam V S, Geist G A. Heterogeneous parallel and distributed computing[J]. Parallel Computing, 1999, 25(13/14): 1699-1721.

[10] Wang L, Siegel H J, Roychowdhury V P, et al. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach[J]. Journal of Parallel and Distributed Computing, 1997, 47(1): 8-22.

[11] Saha D, Menasce D, Porto S, et al. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures[J]. Journal of Parallel and Distributed Computing, 1995, 28(1): 1-18.

[12] IVerson M A, F O Z, Follen G J. Parallelizing existing applications in a distributed heterogeneous environment[Z]. [S.l.]: [s.n.], 1995.

[13] Lastovestky A, Reddy R. On performance analysis of heterogeneous parallel algorithms[J]. Parallel Computing, 2004, 30(11): 1195-1216.

[14] Clematis A, Corana A. Modeling performance of heterogeneous parallel computing systems[J]. Parallel Computing, 1999, 25(9): 1131-1145.

[15] Case K E, Fair R C. Principles of microeconomics[M]. [S.l.]: Pearson Education, 2007.

[16] Mankiw, G N. Principles of economics[M]. [S.l.]: South-Western Pub, 2011.

***Authors**: Bai-Da ZHANG and Jun-Jie WU are with the National laboratory for Parallel and Distributed Processing, School of Computer, National University of Defense Technology, Changsha 410073, P.R. China. Prof. Yu-Hua TANG is with the Department of Computer Science and Technology, School of Computer, National University of Defense Technology, Changsha 410073, China. Shuai XU is with the Department of Information Engineering, Academy of Armored Force Engineering, Beijing 100000, P.R. China*

(E-mail: zhangbaida@gmail.com)