A Model of Multiprocessor System with Communication Delays and Its Scheduling Method

Takashi Otsuka[1] , Hironori Youhata[3] , Qi-Wei Ge[2] , Mitsuru Nakata[2] ,
Yuu Moriyama[3] and Hirotoshi Tonou[3]
[1]Graduate School of Education, Yamaguchi University,
1677-1 Yoshida, Yamaguchi-shi 753-8513, Japan
[2]Faculty of Education, Yamaguchi University,
1677-1 Yoshida, Yamaguchi-shi 753-8513, Japan
[3]Fujitsu TEN Limited
1-2-28 Goshodori, Hyogo-ku, Kobe-shi 652-8510, Japan
E-mail : [12]{k004kn,gqw,mnakata}@yamaguchi-u.ac.jp
[3]tahahironori@hotmail.com, {moriyama,tonou}@me.ten.fujitsu.com

Abstract: This paper aims at developing a scheduling method for multiprocessor systems with communication time. In this paper, we firstly propose a model of multiprocessor system with communication time occurring in reading data. Then, for the proposed model, we propose a scheduling method (called *AMCN* scheduling method) that (i) divides a task graph to subgraphs so that a task (called node hereafter) and its successors and predecessors are as much as possible included in the same subgraph to shorten communication time; and (ii) uses a fixed processor to execute all the nodes of a subgraph. Finally, we do computational simulation experiments to evaluate our scheduling method.

1. Introduction

Multiprocessor systems have been widely used in a variety of computer systems as an effective way to decrease computation time of the program [1]. To maximally achieve the advantage of a multiprocessor system, it is desirable to find out an efficient way to schedule the processors in executing tasks of programs in order to attain the minimum execution time.

Multiprocessor scheduling problem is, given with processors and a program usually represented as a task graph (directed acyclic graph DAG), to determine the order of tasks'(called node hereafter) execution assigned to the processors to minimize the total execution time. However, this problem is known as an NP-hard problem[2]. Only for two special cases polynomial algorithms were found: the first is proposed by Hu [3] and to schedule execution of rooted task graphs with same node execution time by using arbitrary processors; the second is proposed by Coffman and Graham[4] and to schedule execution of general task graphs also with same node execution time but by two processors. For this reason multiprocessor scheduling is usually approached by heuristic methods. As typical list scheduling for task graph, *CP/MISF* proposed by Kasahara *et al.* [5] which provides a suboptimal list scheduling, and *SDS-Switching*[6] that we recently proposed. These focus on a multiprocessor system without communication time. However, communication time can not ignore in our systems. Therefore, our research aims at developing a scheduling method for the multiprocessor system with communication time.

The definitions of communication time are generally different by the models of multiprocessor system. For example, the model of the following multiprocessor systems can be given. (i) The model that a sending processor and a receiving processor consume communication time for data transmission. (ii) The model that a sending processor and a receiving processor are not consumed communication time for data transmission. (iii) The model that only a sending processor or a receiving processor consumes communication time for data transmission. This paper focuses on the model that only a receiving processor consumes communication time. Then, the sending processor can execute a node during data transmission.

This paper is organized as follows. Section 2 describes the basic definition about the model of a multiprocessor system and related definitions. Section 3 describes our scheduling method, which is called "*AMCN* scheduling method", for the proposed model. Section 4 gives the performance evaluation of *AMCN* scheduling method by computational simulation. We compare our scheduling method with *MCP* (*Modified Critical Path*)[7] that has already been proposed. The *MCP* is to improve *CP* (*Critical Path*)[2] that determines a execution node by a priority based on the longest length from each nodes to termination, and determines a execution node by the priority based on ALAP (*as late as possible*).

2. A Model of Multiprocessor System and Necessary Definitions

2.1 A model of multiprocessor system

Here, we propose a model of multiprocessor system with communication time as follows.

- The throughput of processors $p_k \in PE$ is homogeneous.
- Each processor has a local memory and a I/O system (which is implemented on a PCI board). The local memory stores output data of the processor. The I/O system obtains the data from the local memory of different processors by the communication.
- The communication time of reading and writing between a processor and its local memory is zero, and the communication time of reading from other processors via I/O system is not zero.
- Before executing a node, a processor must read data from

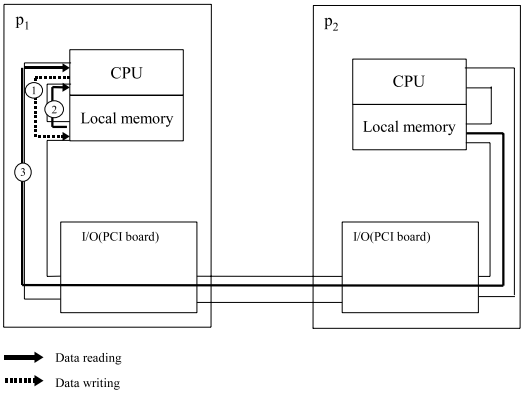

Figure 1. Data writing and reading

all the inputs. And a node cannot be executed until data reading is completed.

Figure 1 shows writing and reading of data. The dashed line ① in Figure 1 expresses the writing of data, which indicates writing of output data into own local memory. And, solid lines②,③ express reading of data. The line② indicates processor p_1 reads an output data of a node processed by itself from its local memory. The line③ indicates processor p_2 reads an output data of a node processed by p_2 through I/O.

2.2 Necessary definitions

Definition 1: Let directed graph G=(N, E, γ, c) be a task graph.

(1) G is DAG composed of node set N and directed edge set E. A node $z_i \in N$ expresses a task, and an edge $e_i \in E$ expresses the connection between two nodes.

(2) Except for s and t that have 0 execution time, each node z_i has random execution time $\gamma_i(\geq 0)$, and executions of these nodes are never interrupted.

(3) A directed edge, which connects between nodes, expresses a precedence constraint and communication channel between the nodes. If there exists a directed edge (z_i, z_j), which connects from z_i to z_j, then execution of z_j cannot be started until execution of z_i has finished. And, a directed edge (z_i, z_j) has weight $c(z_i, z_j)$ expressing the communication time when being processed by different processors. And, $c(s, z_i)$ and $c(z_i, t)$ are 0 for arbitrary z_i. □

Definition 2: Let G=(N, E, γ, c) be a task graph.

(1) The sets of immediate predecessors and successors of z_i are denoted as $IP(z_i)$ and $IS(z_i)$ respectively. Moreover, The set of all the directed paths $\rho_{i,j} = z_i z_{x_1} z_{x_2} \cdots z_j$ from z_i to z_j is denoted by $P_{i,j}$. The number of the nodes contained in $\rho_{i,j}$ is denoted by $|\rho_{i,j}|$.

(2) The set of nodes z_i executed by the processor p_k is denoted $PE\_Node_{p_k}$. At this time, the communication time related to z_f and z_i is denoted

$$C_{f,i} = \begin{cases} 0 & z_f \in PE\_Node_p \\ c(z_f, z_i) & otherwise \end{cases}$$

Communication time required for execution of z_i $tc(z_i)$ (Total Communication time) is

$$tc(z_i) = \sum_{z_f \in IP(z_i)} C_{f,i}. \qquad \square$$

Definition 3: (1) The value related to a directed path $\rho_{i,t}$, calculated by "(summation of execution time and weight of input edges of every node in $\rho_{i,t}$) - (weights of edges included in $\rho_{i,t}$)", is called *Approximate Modified Length* of $\rho_{i,t}$ and denoted by $AMLen(\rho_{i,t})$. The path $\rho_{i,t}$ with the maximum Approximate Modified Length among all the paths from z_i to t is called *Approximate Modified Longest Path* and denoted by $\rho_{i,t}^{aml}$. (2) The value related to $\rho_{i,t}^{aml}$, calculated by "$AMLen(\rho_{i,t}^{aml})$-(weights of edges between the nodes in $\rho_{i,t}^{aml}$)", is denoted by $AMLen^{+}(\rho_{i,t}^{aml})$. Approximate Modified Longest Path $\rho_{s,t}^{aml}$ from s to t is called *Approximate Modified Critical Path* and the node set denoted as $AMCP$. □

3. Proposal of Our Scheduling Method

We describe *AMCN* scheduling method for task graph G, when the number of processors is κ.

In order to shorten communication time, *AMCN* scheduling method divides a task graph into subgraphs so that a node and its successors and predecessors are as much as possible included in the same subgraph, and uses a fixed processor to execute all the nodes of a subgraph. The subgraph is called *Approximate Modified Critical Net* and denoted by $AMCN$.

(1) $AMCN$s ($AMCN_1, AMCN_2, \cdots$) are created sequentially by the following operations. (i) We firstly obtain $AMCP_1$ as well as $AMCN_1$ from G and search the immediate predecessors N_A=$\{z_h\}$ (except the nodes in $AMCN_1$) of the nodes in $AMCN_1$. For each node $z_h \in N_A$, if z_h satisfies

$$\sum_{z_i \in (IS(z_h) \cap AMCN)} c(h,i) \; - \sum_{z_j \in (IP(z_h) - AMCN)} c(j,h) \; - \; \gamma_h \geq 0$$

z_h (called *Added Adjusted Node*) is added to $AMCN_1$. When N_A=$\{\phi\}$, nodes s and t are removed from $AMCN_1$. (ii) Then we do the similar operation as done for $AMCN_1$ to obtain $AMCN_2$ under the remain graph $G - AMCN_1$, and so for $AMCN_3$, $AMCN_4$, $\cdots$. These operations are repeated as many times as the number of processors or till no new $AMCP$ can be found.

(2) Basically, processors ($p_1, p_2, \cdots$) are respectively fixed to executing $AMCN_1, AMCN_2, \cdots$. When there are no executable nodes in some $AMCN$s, say $AMCN_{j_1}$, $AMCN_{j_2}$, $\cdots$, $AMCN_{j_n}$ ($j_1 < j_2 < \cdots j_n$), then processors $\{p_{j_i}\}$ are assigned to the executable nodes of remaining graph (if any) in the priority order of $p_{j_n}, \cdots, p_{j_2}, p_{j_1}$.

3.1 Preprocessing

Here, we describes the preprocessing before scheduling.

Firstly, we calculate $AMLen(\rho_{i,t}^{aml})$ of each node z_i, and then based on $AMLen(\rho_{i,t}^{aml})$ we find *Approximate Modified Longest Path* for each nodes z_i and $AMCP$. Secondly, we calculate $AMLen^{+}(\rho_{i,t}^{aml})$ and treat it as the priority of node z_i in doing our scheduling. $AMLen^{+}(\rho_{i,t}^{aml})$ expresses total execution time of the nodes in $\rho_{i,t}^{aml}$ under the assumptions: (i) these nodes are assigned to a fixed processor; and (ii) all the

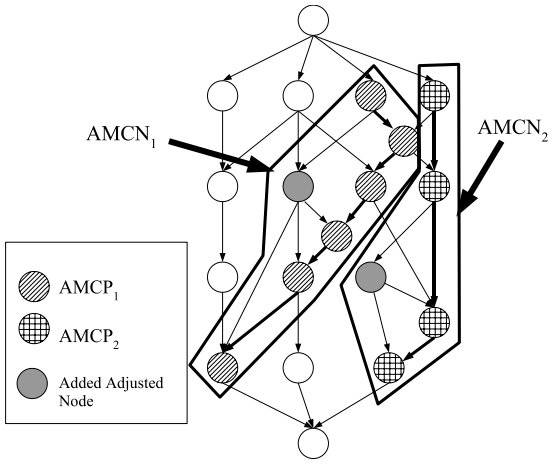

Figure 2. Example of $AMCN$

other nodes are assigned to the processors other than the fixed one. The larger the value of $AMLen^{+}(\rho_{i,t}^{aml})$, the higher the priority of the node z_i is.

$AMCN$s are obtained by the operations as stated in the beginning of this section. The computational complexity to calculate $AMCN$s and $AMLen^{+}(\rho_{i,t}^{aml})$ is $O(N^2)$. Figure 2 shows an example of $AMCN$.

3.2 *AMCN* scheduling method

Our *AMCN* scheduling method assigns a node z_f to an idle processor p_k according to the following Rules 1-3 (at time τ). The following notations are used hereafter.

- N_{ExeEd}: the set of nodes that have been executed.
- N_{ExeAbl}: the set of executable nodes, of which each z_f satisfies $IP(z_f) \in N_{ExeEd}$.
- PE_{Free}: the set of idle processors.
- $FreeNode$: the set of nodes that are not included in $\{AMCN_x\}$.

[Rule 1] If node $AMLen^{+}(\rho_{f,t}^{aml})$ satifies

$AMLen^{+}(\rho_{f,t}^{aml}) =$
$\max\{AMLen^{+}(\rho_{j,t}^{aml}) | z_j \in N_{ExeAbl} \wedge z_j \in AMCN_k\}$,

then the related node z_f is assigned to the processor p_k.

[Rule 2] If node $AMLen^{+}(\rho_{f,t}^{aml})$ satisfies

$AMLen^{+}(\rho_{f,t}^{aml}) =$
$\max\{AMLen^{+}(\rho_{i,t}^{aml}) | z_i \in N_{ExeAbl} \wedge z_i \in FreeNode \wedge \sum_{z_j \in (IS(z_i) \cap AMCN_k)} c(i,j) - tc(z_i) - \gamma_i \geq 0\}$,

then the related node z_f is assigned to the processor p_k.

[Rule 3] If node $AMLen^{+}(\rho_{f,t}^{aml})$ satisfies

$AMLen^{+}(\rho_{f,t}^{aml}) =$
$\max\{AMLen^{+}(\rho_{i,t}^{aml}) | z_i \in N_{ExeAbl} \wedge z_i \in FreeNode\}$
and
$|N_{ExeAbl} \cap FreeNode| +$
$|\{p_x | p_x \in PE_{Free} \wedge |AMCN_x \cap N_{ExeAbl}| \geq 1\}|$
$\geq |PE_{Free} \cap \{p_k \cdots p_\kappa\}|$

then the related node z_f is assigned to the processor p_k.

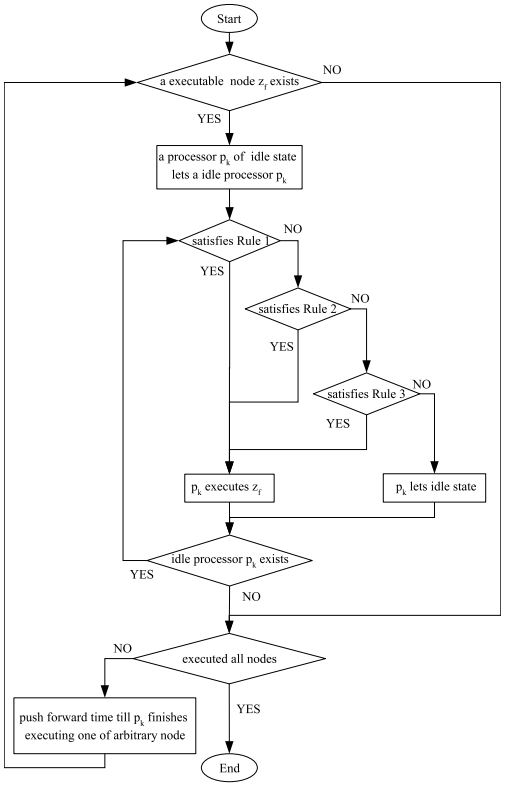

Figure 3. The flow of *AMCN* scheduling method

Rules 1-3 determine which executable node should be assigned to an idle processor p_k. These rules are applied in the order of Rule 1, Rule 2 and Rule 3. If a node z_f is determined to be assigned to processor p_k by Rule 1, then Rule 2 need not to be applied and so forth. If no nodes can be determined by Rules 1-3, processor p_k keeps idle state. Figure 3 shows the flow of our *AMCN* scheduling method.

4. Computational Simulation and The Evaluation Result

We have done computational simulation by using *AMCN* scheduling method as well as a proposed method called *MCP* (Modified Critical Path) [7], and compare the results of these two methods in order to evaluate our method.

4.1 The task graphs used in computational simulation

In our computational simulation, we used the task graphs $CSTGs$ (Communication Standard Task Graphs Set) and these $CSTGs$ are made by modifying the $STGs$ (Standard Task Graph Set) [8] without communication times. In the modification of $STGs$, (1) we added randomly generated communication times (from 1 to 10 unit time) to the edges of the $STGs$ under such a limitation that the edges connected from a same node are with the same communication time; (2) we increased node execution time to 10 times of the original for each node. Each task graph of $CSTG$ has the information, such as the number of immediate predecessor, immediate predecessor number, communication time, the total number of nodes, node number, and execute time of node. Figure

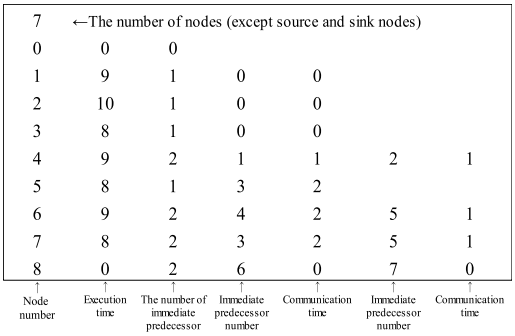

Figure 4. Example of $CSTG$

4 show an example of $CSTG$.

Computational simulation has been done for 6 types of $CSTGs$ with the number of nodes 50, 100, 300, 500, 750, 1000, of which each type contains 180 task graphs. And the number of processors, κ, is 2 and 4.

4.2 Computational simulation results

We compare the simulation results of our method with *MCP*'s using a standard of "improvement rate", $(T_{G_i}^{(1)} - T_{G_i}^{(\kappa)})/T_{G_i}^{(1)}$, where $T_{G_i}^{(1)}$ and $T_{G_i}^{(\kappa)}$ are the schedule lengths when executed by a single processor and κ processors for a task graph G_i.

Table 1, 2 show the averages of the improvement rates of each type of $CSTGs$ as well as the total averages respectively for the cases of 2 and 4 processors. Also the average of the cases of 2 and 4 processors is shown in Table 3. From these computational simulation results, we could find that *AMCN* scheduling method is better than *MCP* in the sense of total average of the improvement rate. In both cases of 2 and 4 processors, the average of *AMCN* is better than *MCP* for each type of graphs except the type of graphs with 300 node number.

However, we found the average improvement rate of *AMCN* is -2.81% when applied to the tasks with 1000 node number in the case of 2 processors, which means the schedule lengths of these graphs executed by 2 processors are even longer than that executed by a single processor. This phenomenon could also been found in the executions of many individual graphs.

5. Conclusion

We have proposed a model of multiprocessor systems with communication time occurring in reading data and for this kind of multiprocessor systems we have proposed a scheduling method *AMCN* trying to decrease the communication times.

To evaluate our *AMCN* scheduling method, we have done computational simulation and compare the results with a previously proposed method *MCP*. Our simulation results show: (1) *AMCN* scheduling method is generally better than *MCP* and can generate shorter schedules than *MCP*; (2) there are cases that the schedule lengths by our *AMCN* scheduling are even longer that executed by a single processor.

As the future works, we need to clarify the cause of the phenomenon as stated in the above (2) and improve our method.

Table 1. Improvement rate (two processors)

κ=2	*MCP*	*AMCN*
50	34.49%	**35.32%**
100	34.68%	**35.16%**
300	23.03%	**23.93%**
500	11.67%	**14.31%**
750	-3.30%	**3.56%**
1000	-18.37%	**-2.81%**
Total	13.70%	**18.24%**

Table 2. Improvement rate (four processors)

κ=4	*MCP*	*AMCN*
50	55.24%	**56.17 %**
100	58.89%	**59.19 %**
300	**53.71%**	53.64%
500	45.15%	**45.55%**
750	33.73%	**35.00%**
1000	22.18%	**26.34%**
Total	44.81%	**45.98%**

Table 3. Comprehensive evaluation

Scheduling method	Improvement rate
MCP	29.26%
AMCN	**32.11%**

References

[1] Mieko Kawajiri, Qi-Wei Ge, Mitsuru Nakata, Kiyoshi Yagi, Hirotoshi Tonou, "On Multiprocessor Scheduling for Automotive Control Systems", *Technical Report of IEICE*, vol.102, no.259, pp.7-10 (CST2002-11) (2003). in Japan

[2] E.G.Coffman, *Computer and Job-Shop Scheduling Theory*, John Wiley, New York (1976).

[3] T.C.Hu, "Parallel sequencing and assembly line problems", *Oper. Res.*, vol.9, pp.841-848 (1961).

[4] E.G.Coffman and R.L.Graham, "Optimal scheduling for two-processors systems", *Acta. Informatica*, vol.1, pp.200-213 (1972).

[5] Hironori Kasahara, Seinosuke Narita, "A Practical Optimal / Approximate Algorithm for Multi-Processor Scheduling Problem", *Technical Report of IEICE*, vol.J67-D, no.7, pp.792-799 (1984). in Japan

[6] Hironori Youhata, Takashi Otsuka, Qi-Wei Ge, Mitsuru Nakata, Hirotoshi Tonou, "A Proposal of Dynamic Multi-processor Scheduling by Considering Critical Path", *Technical Report of IEICE*, vol. 106, no. 502, pp. 7-12, (CST2006-39) (2007). in Japan

[7] Min-You Wu, Daniel D.Gajski, "Hypertool: A programming aid for message-passing systems", *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 3, pp. 330-343, July (1990).

[8] "Standard Task Graph Set", http://www.kasahara.elec.waseda.ac.jp/schedule/.