INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211

A Mapping Heuristic Approach for Scheduling the Tasks on Parallel Multiprocessors

Preeti Gupta
Department of Computer Science
Guru Nanak Dev University Amritsar, India

Abstract:

Multiprocessor scheduling is an NP-hard problem, no exact tractable algorithm exist. Many algorithms to schedule DAGs on multiprocessors have been proposed but this paper is an attempt to implement the scheduling algorithm called mapping heuristic of APN class. This paper also gives the scheduling trace of the algorithm which will describe how the tasks are allocated to the different processors with the help of Gantt chart.

Key words: *Multiprocessor, DAG, Algorithm, List scheduling, parallel processing.*

1.Introduction

The problem of finding an optimal schedule for any DAG on an arbitrary multiprocessor topology has been shown to be NP-complete. This paper presents an algorithms that allocates the tasks to the homogeneous processors which are represented by directed acyclic graph (DAG) or task graph with the objective of minimizing the overall finish-time by proper allocation of the tasks to the processors and arrangement of execution sequencing of the tasks such that system throughput is maximized. Scheduling is done in such a manner that the precedence constraints among the tasks are preserved [1], [3]. The overall finish-time of a parallel program is commonly called the schedule length or makespan. The rest of this paper is organized as follows. In next section, DAG model has been described. The APN scheduling algorithms is discussed in Section 3. The experimental set up and results are presented in Section 4 and Section 5 concludes the paper.

2.DAG Model

The DAG is a generic model of a parallel program consisting of a set of processes among which there are dependencies. In static scheduling, a parallel program can be represented by a directed acyclic graph (DAG) G = (V, E) where V is a set of v nodes and E is a set of e directed edges. A node in the DAG represents a task which in turn is a set of instructions. The weight of a node is called the computation cost and is denoted by $w(n_i)$. The edges in the DAG is denoted by (n_i, n_j) which correspond to the communication messages and precedence constraints among the nodes [8], [10]. The weight of an edge is called the communication cost of the edge and is denoted by $c(n_i, n_j)$. The source node of an edge is called the parent node while the sink node is called the child node. A node with no parent is called an entry node and a node with no child is called an exit node. The communication-to-computation-ratio (CCR) of a parallel program is defined as its average edge weight divided by its average node weight. The terms node and task are used interchangeably.

The precedence constraints of a DAG dictate that a node cannot start execution before it gathers all of the messages from its parent nodes. The communication cost between two tasks assigned to the same processor is assumed to be zero. The node and edge weights are usually obtained by estimation at compile-time [2], [3], [4].

3.APN Scheduling

In this, list scheduling, scheduling attributes, the APN class of DAG scheduling algorithm called mapping heuristic (MH) has been surveyed. In this, MH (Mapping Heuristic) scheduling algorithm [4] has been described which is based on list scheduling.

3.1.List Scheduling

The basic idea of list scheduling is to make an ordered list of nodes by assigning them some priorities and then repeatedly execute the following two steps until a valid schedule is obtained:

- A task prioritizing phase
- A processor selection phase

The priorities are determined statically before the scheduling process begins [8], [9]. In the scheduling process, the node with the highest priority is chosen for scheduling. In the second step, the best processor which allows the earliest start time is selected to accommodate this node. Scheduling algorithms are based on employing variations in the priority assignment methods such as HLF (Highest level First), LP (Longest Path), LPT (Longest Processing Time) and CP (Critical Path). Static priority assignment may not always order the nodes for scheduling according to their relative importance. The drawback of the static approach is that an inefficient schedule may be generated if a relatively less important node is chosen for scheduling before the more important ones. Static priority assignment fails to capture the variation in relative importance of nodes during the scheduling process. In order to avoid scheduling less important nodes before the more important ones, node priorities need to be determined dynamically during the scheduling process. The priorities of nodes are re-computed after a node has been scheduled in order to capture the changes in the relative importance of nodes. The following three steps are repeatedly executed:

- Determine new priorities of all unscheduled nodes.
- Select the node with the highest priority for scheduling.
- Select the most suitable processor to accommodate this node. Scheduling algorithms which employ the above three-step approach can generate better schedules but can increase the complexity of the algorithm.

3.2.Scheduling Attributes

The main scheduling attributes [9], [11] used in DAG for assigning priority while evaluating the algorithms are as follow:

3.2.1.T-Level

T-level of the node n_i in DAG is the length of the longest path from entry node to n_i not including n_i. It is the sum of all the nodes computational costs and edges weights along the path.

T-level (n_i) = max (t-level(n_m) + w_m + $c_{m,i}$)

where n_m ε predecessors of n_i, w_m stands for computational cost, $c_{m,i}$ stands for the communication cost and

t-level (n_{entry}) = 0.

3.2.2.B-Level

B-level of node n_i in DAG is the length of the longest path from n_i to the exit node. It is the sum of all nodes computational costs and edges weights along the path.

b-level(n_i) = w_i + max (b-level(n_m) + $c_{m,i}$)

where n_m ε successors of n_i, w_m stands for computational cost, $c_{m,i}$ stands for the communication cost and b-

level(n_{exit}) = w(v_{exit}).

3.2.3.Sl (Static Level)

If the edges weights are not taken while considering the b-level then it is called Static Level.

SL(n_i) = w_i + max(SL(n_m))

where n_m ε successors of n_i and SL (n_{exit}) = w(v_{exit}).

3.2.4.Cp (Critical Path)

It is the length of the longest path from standing node to the exit node in DAG.

3.2.5.Est (Earliest Starting Time)

Earliest Starting Time is same as the t-level.

EST(n_i) = max(EST(n_m) + w_m + $c_{m,i}$)

where n_m ε predecessors of n_i, wm stands for computational cost, $c_{m,i}$ stands for the communication cost and

EST(n_{entry}) = 0.

3.2.6.Lst (Latest Starting Time)

Latest Starting Time of node is computed by following the path starting from exit node upwards till the desired node is reached.

LST (n_i) = min (LST (n_m) – $c_{m,i}$) – w_i

where n_m ε successors of n_i, w_m stands for computational cost, $c_{m,i}$ stands for the communication cost and LST (n_{exit}) = EST (n_{exit}).

3.2.7.DL (Dynamic Level)

Dynamic level of the node is calculated by subtracting the Earliest Start Time from the Static Level.

DL = SL-EST

where SL stands for static level and EST stands for early start time.

3.3.Mh (Mapping Heuristic) Algorithm

The MH (Mapping Heuristic) algorithm [4] first assigns priorities by computing the sl of all nodes in decreasing priorities.

The MH algorithm is briefly described below.

- Compute the SL of each node n_i in the task graph.
- Initialize a ready node list by inserting all entry nodes in the task graph. The list is ordered according to node priorities, with the highest priority node first.

Repeat

- n_i ¬ the first node in the list
- Schedule n_i to the processor which gives the smallest start-time. In determining the start-time on a processor, all messages from the parent nodes are scheduled and routed by consulting the routing tables associated with each processor.
- Append all ready successor nodes of n_i, according to their priorities, to the ready node list.Until the ready node list is empty.

4.Experimental Setup And Results

This section will focus on experiment setup in which Mapping Heuristic (MH) APN algorithm is coded in C for 7 nodes.

Directed Acyclic Graph (DAG) with 7 nodes is considered as shown in figure 7.1(a). In this, task n_1 is the Entry task and task n_7 is Exit task. Task n_2, n_3 and task n_4 are dependent upon task n_1 and cannot be executed until task n_1 is not completed. Similarly task n_5 cannot be executed until tasks n_2 and n_4 are not completed and tasks n_6 cannot be executed until tasks n_2, n_3 and n_4 are not completed. Thus task n_7 cannot be executed until tasks n_5 and n_6 are completed.

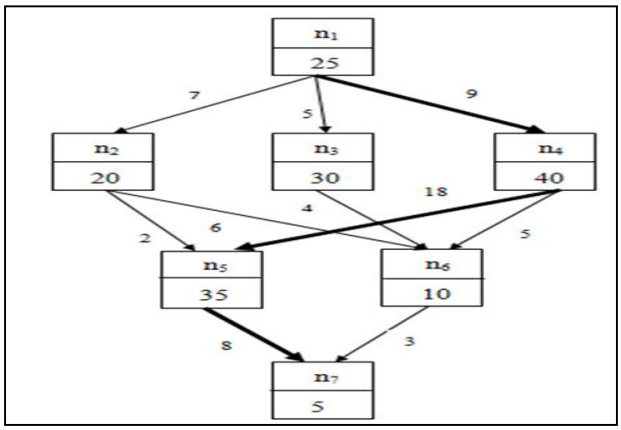

Figure 1: DAG with 7 nodes

4.1.Implementation of MH algorithm

The first step is to enter the number of nodes. The second step is to enter the network diagram for 7 nodes in which only non-zero values are entered and -1 is printed to exit. Similarly all the values for the network diagram are entered. Then the third step is the calculation of critical path which n_1, n_4, n_5, n_7.

The fourth step is to enter the weights of each node. After entering the weights of each node, the next step is to enter the sl values of each node. Then after this, DAG table or priority table is displayed as shown in table I which includes various scheduling attributes such as sl-level, t-level, b-level, p-level and alap. For each node, all the scheduling attributes are calculated, since each of the algorithms requires different attributes for mapping tasks to processors.

DAG TABLE					
N_i	**Sl**	**t-level**	**b-level**	**ALAP**	**p-level**
n_1	105	0	140	0	0
n_2	60	32	70	70	1
n_3	45	30	52	88	1
n_4	55	34	106	34	1
n_5	40	92	48	92	2
n_6	15	79	18	122	2
n_7	5	135	5	135	3

Table 1: DAG table

By using the DAG table, the final schedule list for MH algorithm is n_1, n_3, n_4, n_2, n_5, n_6, n_7.

For the DAG shown in Fig. 1, the scheduling trace of MH algorithm for 3 processors is given in Table 2.

In the table, the execution start times of each node on all available processors at each step are given and the nodes on the list are scheduled one by one to the processor that allows the earliest execution start time.

Steps	Nodes	P1	P2	P3	Selected P
1	n_1	0	0	0	P1
2	n_3	25	30	30	P1
3	n_4	55	34	34	P2
4	n_2	55	74	32	P3
5	n_5	92	74	92	P2
6	n_6	79	109	79	P3
7	n_7	117	109	117	P2

Table 2: Scheduling trace of MH algorithm for 3 processors

The Gantt chart for MH algorithm for 3 processors is shown in figure 2. The MH algorithm uses sl attribute to schedule the tasks in the list. Task 1 is schedule to P1 first as its starting time is 0 seconds and it completes at 25 seconds. Since task 3 is dependent on task 1 so task 3 waits till task 1 completes and task 3 starts right where task 1 completes and task 3 completes at 55 seconds. Task 4 is scheduled to P2 because its earliest starting time is 34seconds. After task 4, task 2 is schedule on P3, since P3 is the only processor left on which there is no task scheduled.
Task 5 depends on task 2 and 4 so it has to wait till all of them are completed.
Task 5 is scheduled on P2 because its earliest starting time is 74 seconds. Task 6 can be scheduled on P1 or P3 as both are available so task 6 is schedule on P3 and task 7 is scheduled on P2.

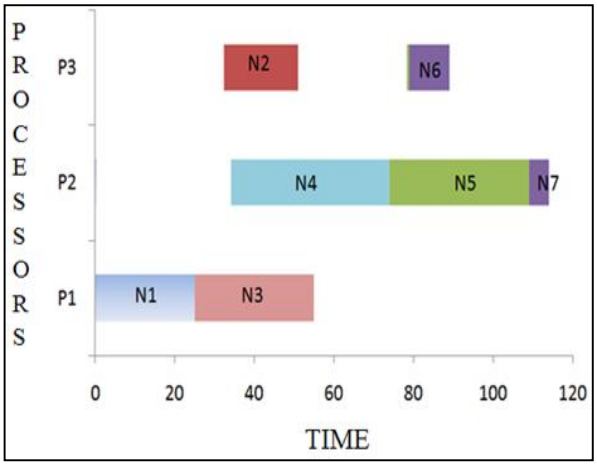

Figure 2: Gantt Chart of MH algorithm for 7 Nodes

5.Conclusion

A generalized mapping scheme for a parallel computing environment is proposed. The strategy uses the knowledge from the given algorithm and the given architecture to guide the mapping. In this, Mapping Heuristic (MH) APN scheduling algorithms has been implemented for task scheduling in parallel multiprocessor system including the communication delays to reduce the completion time and to increase the throughput of the system.

6.Reference

1. Shahid H. Bokhari, "On the Mapping Problem", IEEE Transactions on Computers, vol. C-30, 1981, pp. 207- 214.
2. A.F. Bashir, V. Susarla, and K. Vairavan, "A Statistical Study of the Performance of a Task Scheduling Algorithm," IEEE Transactions on Computers, vol. C-32, no. 8, Aug.1983, pp. 774-777.
3. J.K. Lenstra, A.H.G. Rinnooy Kan, "An Introduction to Multiprocessor Scheduling", Technical Report, CWI, Amsterdam, 1988.
4. H. El-Rewini and T.G. Lewis, "Scheduling Parallel Programs onto Arbitrary Target Machines," Journal of Parallel and Distributed Computing, vol. 9, no. 2, Jun. 1990, pp. 138-153.
5. V. Chaudhary and J. K. Aggarwal, "Generalized mapping of parallel algorithms onto parallel architectures," in Proc. Int. Conf. Parallel Processing, Aug, 1990, pp. 137- 141.
6. G.C. Sih and E.A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 2, Feb. 1993, pp. 75-87.
7. Viph Chaudhary, Member, IEEE, and J. K. Agganval, Fellow, "A Generalized Scheme for Mapping Parallel Algorithms", IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 3, march 1993.
8. A.A. Khan, C.L. McCreary, and M.S. Jones, "A Comparison of Multiprocessor Scheduling Heuristics," Proceedings of International Conference on Parallel Processing, vol. II, Aug. 1994, pp. 243-250.
9. I. Ahmad, Y.-K. Kwok, and M.-Y. Wu, "Analysis, Evaluation, and Comparison of Algorithms for Scheduling Task Graphs on Parallel Processors," International Symposium on Parallel Architectures, Algorithms, and Networks, Beijing, China, pp. 207-213, Jun. 1996.
10. Yu-Kwong Kwok, "Benchmarking and Comparison of the Task Graph Scheduling Algorithms", Journal of Parallel and Distributed Computing 59, pp. 381-422, 1999.
11. T. Hagras, J.Janecek, "Static vs. Dynamic List Scheduling Performance Comparison", Acta Polytechnica, Vol. 43, No. 6, 2003.