Volume 4, Issue 3, March 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Analysis & Comparison of BNP Class of Algorithms Based on Matrics

Manju Dandi*

CSE & GNDU, Amritsar

India

Abstract— Parallel processing is a technique of executing the multiple tasks concurrently on different processors. Parallel processing is used to solve the complex problems that require vast amount of processing time. Task scheduling is one of the major problems of parallel processing. In this paper we survey algorithms that allocate a parallel program represented by an edge-weighted directed acyclic graph (DAG), also called a task graph or macro dataflow graph, to a set of homogeneous processors with the objective of minimizing the completion time. We examine BNP class of algorithms and then compare the performance of a class of scheduling algorithms known as the bounded number of processors (BNP) scheduling algorithms. Comparison is based on various scheduling parameters such as makespan, speed up, processor utilization and scheduled length ratio.

Keywords— Task Scheduling, Bounded Number of Processor (BNP) Scheduling, HLFET, MCP, ETF, DLS

I. INTRODUCTION

The task scheduling in multiprocessor system is also known as multiprocessor scheduling [1], it is used in a large number of computer applications. In parallel processing, it needs to minimized execution time with respect to a minimum number of processors. The major objective of task scheduling in multiprocessor system is to minimize the scheduling length. The scheduling length is also called makespan. The task scheduling in multiprocessor system has been proven to be NP-Complete except for few restricted cases. The scheduling problem is NP-Complete [4], [10] into simple cases: scheduling tasks with uniform weights to an arbitrary number of processor and scheduling tasks with weights equal to one or two units to two processors. Traditional static scheduling [8], [9] algorithms attempt to minimize the schedule length through iterative local minimization of the start times of individual tasks. This paper presents three task scheduling algorithms which can schedule directed acyclic graphs (DAGS) with a complexity of O (v2 log U), where w is the number of tasks in the DAG [1], [10]**.** The algorithms schedule the tasks and they are suitable for graphs with arbitrary computation and without communication costs, and are applicable to systems with homogeneous [9] fully connected processors. The performances of the algorithms have been observed by comparing bounded number of processor (BNP) scheduling algorithms in terms of the schedule length of parallel and sequential processing. In this paper we try to answer some of these questions by examining a number of recently proposed algorithms. These algorithms can be classified into following categories:

A. Bounded Number of Processor (BNP) Scheduling

These algorithms schedule the DAG to a bounded number of processor directly. It is based on the list scheduling technique. List scheduling is a class of scheduling heuristics in which the nodes are assigned priorities and placed in a list arranged order of priority.

B. Unbounded Number of Clusters (UNC) Scheduling

These algorithms schedule the DAG to a bounded number of clusters. The processors are assumed to be fully connected. The technique employed by these algorithms is also called clustering.

C. Task Duplication Based (TDB) Scheduling

The principle behind the TDB algorithm is to reduce the communication overhead by redundantly allocating some tasks to multiple processors. In duplication different strategies can be employed to select ancestor nodes for duplication.

D. Arbitrary Processor Network (AP) Scheduling

The algorithms in these account specific architectural features such as the number of processor as well as their interconnection topology. These algorithms can schedule tasks on the processors and messages on the network communication links.

II. DAG MODEL

A parallel program can be represented by a weighed Directed Acyclic Graph (DAG)[1], in which the vertex/node weights represent task processing time and the edge weights represent data dependencies as well as the communication time between tasks. The communication time is also referred as communication cost. Directed Acyclic Graph (DAG) is a directed graph that contains no cycles. A rooted tree is a special kind of DAG and a DAG is a special kind of directed graph. Directed Acyclic Graph (DAG) G = (V, E), where V is a set of v nodes/vertices and E is a set of e directed edges. 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. Some variations in the generic DAG model are:-

A. Accurate Model

In an accurate model [10], the weight of a node includes the computation time, the time to receive messages before the computation, and the time to send messages after the computation the weight of an edge is a function of the distance between the source and destination nodes, and therefore, depends on the node allocation and network topology. It also depends on network contention, which can be difficult to model. When two nodes are assigned to a single processor, the edge weight becomes zero, so as the message receiving time and sending time.

B. Approximate Model 1

In this model, the edge weight is approximated by a constant, independent of the message transmission distance and network contention. A completely connected network without contention fits this model.

C. Approximate model 2

In this model [6], [10], the message receiving time and sending time are ignored in addition to approximating the edge weight by a constant.

III. BASIC TECHNIQUES IN DAG SCHEDULING

Most of the scheduling algorithms are based on the list scheduling technique. In list scheduling, the nodes are assigned priorities and placed in a list arranged in a descending order of priority. A node with higher priority is examined for scheduling before a node with lower priority. If more than one node has the same priority, ties are broken using specific method. Two frequently used attributes for assigning priority are:

A. t-level

The t-level of a node T_i is the length of a longest path (there can be more than one longest path) from an entry node to T_i (excluding T_i). Here, the length of a path is the sum of all the node and edge weights along the path. The t-level of T_i highly correlates with T_i's earliest start-time, denoted by TS(T_i), which is determined after T_i is scheduled to a processor.

B. b-level

The b-level of a node T_i is the length of a longest path from node Ti to an exit node. The b-level of a node is bounded from above by the length of a critical path. A critical path (CP) of a DAG, which is an important structure in the DAG, is a longest path in the DAG. Clearly a DAG can have more than one CP. Consider the task graph shown in Fig. 1. In this task graph, nodes T_1, T_2, T_4, T_6 are the nodes of the only CP and are called CPNs (Critical-Path Nodes).

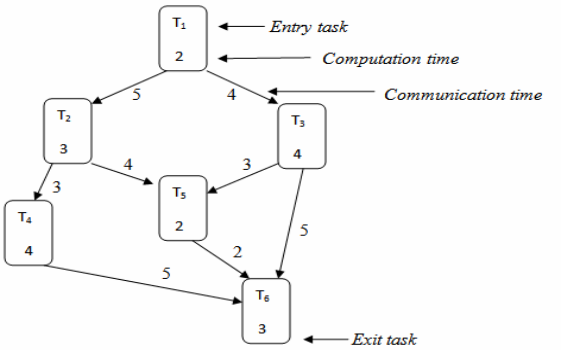

Fig. 1 An example of a DAG

It should be noted that some scheduling algorithms do not take into account the edge weights in computing the b-level. In such a case, b-level is called static b-level [2] or static level (SL). If an algorithm uses a static attribute, such as b-level or static b-level, to order nodes for scheduling, it is called a static algorithm; otherwise, it is called a dynamic algorithm. Note that the procedure for computing the t-levels can also be used to compute the start times of nodes on processors during the scheduling process. Indeed, some researchers call the t-level of a node the ASAP (As-Soon-As-Possible) start-time because the t-level is the earliest possible start-time. Some of the DAG scheduling algorithms employs an attribute called ALAP (As-Late-As- Possible) start-time. The ALAP start-time of a node is a measure of how far the node's start-time can be delayed without increasing the schedule length.

We use following DAG in scheduling algorithms comparison (DAG shown in Fig. 2). The values of the priorities discussed above are shown in TABLE I.

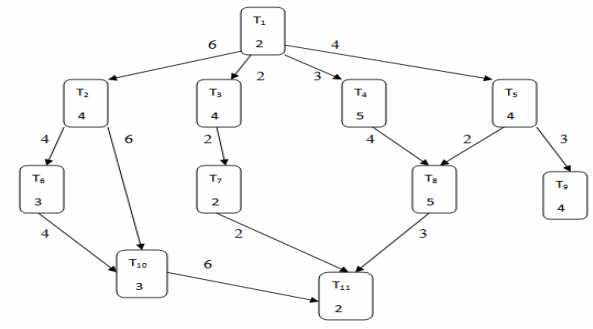

Fig. 2 DAG that is used in algorithm comparison

TABLE II: PRIORITY ATTRIBUTES OF EVERY TASK (TI)

Task (Ti)	t-level	b-level	ALAP	Static Length (SL)
T1	0	34	0	14
T2	8	26	8	12
T3	4	12	22	8
T4	5	19	15	12
T5	6	16	18	11
T6	16	18	16	8
T7	10	6	28	4
T8	14	10	24	7
T9	13	4	30	4
T10	23	11	23	5
T11	32	2	32	2

IV. BNP CLASS OF SCHEDULING ALGORITHMS

Most BNP scheduling algorithms are based on the list scheduling technique. The BNP classes of scheduling [1], [2] algorithms are developed for homogeneous and limited number of processors. It also considered that all processors are fully connected. Now we have to analysis all four scheduling algorithms as per their priority attribute and scheduling length.

A. The Highest Level First with Estimate Times (HLFET) algorithm

This is the simplest algorithm among the BNP [4] class of scheduling algorithms. It was developed for fully connected identical processors. In this algorithm, priority of the task is decided by using static level (SL) attribute. The major problem with this algorithm, it does not use communication cost during the entire process. i.e. ignore the communication cost. The HLFET algorithm is given below.

Step1. Calculate the static level (SL) of each task.

Step2. Make a ready list in a descending order of static level (SL). Initially, the ready list contains only the entry task. Ties are broken randomly.

Repeat

Step3. Schedule the first task in the ready list to a processor that allows the earliest execution, using the non-insertion approach.

Step4. Update the ready list by inserting the tasks that are now ready.

Until all tasks are scheduled

The time complexity of HLFET is O (v^2).

B. The Modified Critical Path (MCP) Algorithm

The MCP uses ALAP attribute to calculate the priority of each task. It can be computed by using formula :ALAP= CP - b-level (Ti), where CP is longest path from entry to exits task of the DAG and b-level(Ti) is bottom level of each task {Ti: i=1,2…n}. It includes the communication time in the scheduling process. The major problem with the MCP algorithm that it includes the communication time for computing the priority of tasks but does not assign task priorities accurately. The MCP algorithm is given below:

Step1. Compute the ALAP time of each task.

Step2.For each task, create a list, which consists of the ALAP times of the tasks itself and all its children in a descending order.

Step3. Sort these lists in an ascending lexicographical order. Create a task list according to this order.

Repeat

Step4. Schedule the first task in the tasks list to a processor that allows the earliest execution, using the insertion approach.

Step5. Remove the task from the task list.

Until the task list is empty.

The time complexity of MCP algorithm is O (v^2logv).

C. The Earliest Time First (ETF) Algorithm

The ETF algorithm uses static level (SL) attribute to compute the priority of each task of graph. It is not need to be schedule task with highest priority due to each step of scheduling. The ETF algorithm is compute the earliest start time of each task of graph and select the smallest among them for schedule. If two or more task having same earliest start time, then ETF algorithm selects the task as per static level (SL) with the highest priority. The major problem ETF algorithm that it cannot be reduces partial scheduling length at each step of scheduling process. The ETF algorithm is explained below:

Step1. Compute the static level of each task.
Step2. Initially, the pool of the ready tasks includes only the entry task.
Repeat
Step3.Calculate the earliest start time on each processor for each task in the ready pool. Pick the task-processor pair that gives the earliest time using the non-insertion approach. Ties are broken by selecting the task with a higher static level (SL). Schedule the node to the corresponding processor.
Step4 Add the newly ready task to the ready task pool.
Until all tasks are scheduled.
The time complexity of ETF algorithm [7] is O (pv2).

D. The Dynamic Level Scheduling (DLS) Algorithm

The DLS algorithm [4], [7] determines the priority of task by using dynamic attribute is called Dynamic Level (DL). It is computed by static level and earliest time first. Formally, the DL is defined as the difference between static level and earliest start time of every task. After computed DL of each task, then select the task with the highest DL value. The problem with DLS algorithm, it does not maintain a scheduling list during scheduling process. The DLS algorithm is explained below:

Step1. Calculate the static level of each task.
Step2. Initially, the ready node pool includes only the entry tasks.
Repeat
Step3. Calculate the earliest start time for every ready task on each processor. Hence, compute the DL of every node-processor pair by subtracting the earliest start time from static level of task.
Step4. Select the node-processor pair that gives the largest DL Schedule the task to the corresponding processor.
Step5. Add the newly ready tasks to the ready pool
Until all task are scheduled
The time complexity of DLS algorithm is O (pv3)

V. AN ILLUSTRATED EXAMPLE

Consider DAG or task graph shown in Fig. 2. After implement above-mentioned BNP algorithms, we will get following scheduling lengths of each algorithm as shown in TABLE II.

TABLE II
SCHEDULE LENGTH

BNP Algorithms	**Schedule Length**
HLFET	22
MCP	24
ETF	25
DLS	24

VI. COMPARISON MATRICES

In this section, we will use four types of matrices to compare the BNP class of scheduling algorithms which are based on scheduling length. They are speedup, efficiency, normalized scheduling length (NSL) and load balancing.

A. Speedup

Speedup [4] is the ratio between sequential execution time and parallel execution time where the sequential time execution time is sum of total computation time of each task and parallel time execution is the scheduling length on limited number of processors. Speedup= Sequential time execution/ Parallel time execution

B. Efficiency

The efficiency of a parallel program is a measure of processor utilization. Efficiency=Speedup/ number of processors.

C. The Normalized scheduling length (NSL)

The Normalized scheduling length (NSL) of a scheduling algorithm is given by NSL=Scheduling length of particular algorithm / Max { sum of computation costs along a path}.

D. Load Balancing

The load balancing is calculated by the ratio of scheduling length to average execution time over all processors. Load Balance= Scheduling length /Average. Where Average = sum of processing time each processor / number of processors.

VII. COMPARISON OF BNP CLASS OF SCHEDULING

The BNP class of scheduling algorithms will be compared based on comparison matrices. *HLFTE Scheduling[5]:* The HLFET algorithm has the scheduling length is 22, So that we can calculate the matrices based on scheduling length. Speedup= Sequential Execution Time/ Parallel Execution Time= 38/22=1.727 Efficiency= (Speedup/Number of processors)*100= (1.727/4)x 100=43.18% Load Balance=Scheduling length/Average=22/14=1.571 NSL=Scheduling length/ Max {sum of computation cost along path} =22/34=0.647

Similarly we can also calculate the matrices of other three BNP class of scheduling algorithm. *The MCP algorithm:* The MCP algorithm has scheduling length is 24 so that Speedup=1.583, Efficiency=39.58%, Load Balance=1.655, NSL=0.706 The ETF algorithm: The ETF algorithm has scheduling length is 25 so that

Speedup=1.520, Efficiency=38 %, Load Balance=1.666, NSL=0.735 The DLS Algorithm: The DLS algorithm has scheduling length is 24 so that

Speedup=1.583, Efficiency=39.58 %, Load Balance =1.846, NSL=0.706 In the TABLE III, we have given the comparison based on the matrices. The HLFET algorithm gives minimum scheduling length, maximum speedup, maximum efficiency, minimum load balance and NSL. The ETF algorithm gives maximum scheduling length, minimum speedup, minimum efficiency and maximum NSL. Both MCP and DLS algorithms have same scheduling length, speedup, efficiency and NSL but differ in load balancing.

TABLE IV
COMPARISON BASED ON THE COMPARISON MATRICES

S. No	Algorithms	Speedup	Efficiency	Load Balance	NSL
1	HLFET	1.727	43.18%	1.571	0.647
2	MCP	1.583	39.58%	1.655	0.706
3	ETF	1.520	38%	1.666	0.732
4	DLS	1.583	39.58%	1.846	0.706

VIII. CONCLUSIONS

In this paper, we have studied BNP (Bounded Number Processors) Classes of scheduling: HLFET, MCP, ETF and DLS scheduling algorithms. We compared their performance based on speedup, efficiency, load balancing and normalized schedule length. The HLFET scheduling are the highest speedup and efficiency than MCP, ETF, and DLS scheduling. The MCP and DLS scheduling are same speedup and efficiency. The DLS scheduling gives better load balance than HLFET, MCP and ETF scheduling. The ETF scheduling gives better NSL than HLFET, MCP, DLS scheduling. All these comparison, we have studied on eleven tasks DAG model. The future wok would be, we will also comparison other class of multiprocessor scheduling algorithms based on matrices with more task graph.

ACKNOWLEDGMENT

The author of this paper is highly grateful to Dr. Gurvinder Singh, Associate Professor & Head, DCSE, GNDU, Amritsar and Mr. Balvinder Singh, PHP Developer at Techies India Incorporation, Ludhiana for his precious guidance from time to time, without which it would not have been possible for to accomplish this work.

REFERENCES

[1] Yu-Kwong Kwok and Ishfaq Ahmad," Static Scheduling Algorithms for Allocating Directed Task Graphs to Mulitprocessors", ACM Computing Surveys, Vol.31 N0.4, December 1999.

[2] Ishfaq Ahmad, Yu-Kwong Kwok and Min-You Wu," Amalysis, Evaluation, and Comparison of Algorithms for Scheduling Task Graphs on Parallel Processors ",IEEE,1996.

[3] T. Hagras, J. Janecek," Static vs. Dynamic List-Scheduling Performance Comparison", Acta Polytechnica Vol. 43 No. 6/2003.

[4] Ranjit Rajak," Comparsion Of Bounded Number Of Processors (BNP) Class Of Scheduling Algorithms Based On Matrics", Computer Science and Telecommunications 2012|No.2(34).

[5] Parneet Kaur, Dheerandra Singh, Gurvinder Singh, "Analysis Comparison and Performance Evaluation of BNP Scheduling Algorithm in parallel Processing", International journal of Knowledge engineering.

[6] By Er. Navneet Singh, Er. Gagandeep Kaur, Er. Parneet Kaur & Dr. Gurdev Singh," Analytical Performance Comparison of BNP Scheduling Algorithms", Volume 12 Issue 10 Version 1.0 year 2012.

[7] Manik Sharma, Dr. Gurdev Singh and Harsimran Kaur "A STUDY OF BNP PARALLEL TASK SCHEDULING ALGORITHMS METRIC'S FOR DISTRIBUTED DATABASE SYSTEM" International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.1, January 2012.

[8] Manik Sharma and Smriti," STATIC AND DYNAMIC BNP PARALLEL SCHEDULING ALGORITHIMS FOR DISTRIBUTED DATABASE", International Journal of Computers & Technology Volume 1 No.1 Dec. 2011.

[9] Gurvinder Singh, Kamaljit Kaur, Amit Chhabra, "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System", International Journal of Computer Science and Security (IJCSS), Volume (4): Issue (2).

[10] Ramandeep Kaur and Priyanka kachroo," EVALUATION AND COMPARISON OF MAKE SPAN TIME IN BNP SCHEDULING ALGORITHMS", Vol 1 Issue 6 June 2013, ISSN 2320 – 4486.

