Multiprocessor Scheduling Using Task Duplication Based Scheduling Algorithms: A Review Paper

Ravneet Kaur[1], Ramneek Kaur[2]
Department of Computer Science
Guru Nanak Dev University, Amritsar, Punjab, 143001, India.

ABSTRACT
Parallel processing is an efficient approach to meet the computational constraints of a large number of the current and emerging applications. Multiprocessor scheduling is the process of assigning tasks to multiple processors in a parallel fashion such that the time required for completion of all the processes is minimized. In this paper, various parallel scheduling strategies and scheduling algorithms has been studied. The task duplication based scheduling algorithms are studied in more detail. It provides greater efficiency and minimum makespan time as compared to other scheduling techniques. This paper also provides the list of performance metrics on the basis of which different algorithms can be compared.

Keywords: Parallel processing, Scheduling, DAG model, TDB Scheduling.

1. INTRODUCTION

During the past years, dramatic increases in computing speed have been observed. Most of these achievements in speed were due to the use of faster electronic components. So, scientist started to look for a new data processing approach, which is known as parallel processing. Parallel processing is an efficient approach to meet the computational constraints of a large number of the current and emerging applications. The reasons for the popularity and attractiveness of parallel processing are:

1. Reducing cost of computer hardware.
2. Can perform the applications which are beyond the limits of conventional computers

Scheduling and allocation is very important issue because an inappropriate scheduling of tasks can fail to exploit the optimum potential of the system. The objective of scheduling is to minimize the completion time of application by properly allocating the tasks to the processors [1].

The scheduling problem is NP-complete for most of its variants except for a few simplified cases [5], [6], [8], [9]. Therefore many heuristics with polynomial-time complexity have been suggested. However, these heuristics are greatly diverse in terms of their assumptions about the parallel program's structure and the target parallel architecture, and thus are difficult to explain. A wide variety of architectures have been developed employing various design methodologies. The architectural attributes such as system topology, routing strategy, overlapped communication and computation, etc., if taken into account, can result in different allocation decisions.

For more realistic cases, a scheduling algorithm needs to address a number of issues. It should exploit the parallelism by identifying the task graph and taking into consideration arbitrary computation, task granularity, load balancing and the number of processors, communication costs, and interprocessor communication. Moreover, a scheduling algorithm should be fast and economical in terms of number of processors used.

An execution scheduling consists of three components:

1. Performance of homogeneous of processor.
2. Mapping of tasks onto processors.
3. Sequence of the execution of the tasks on each processor.

All three components of the optimization problem have high dependencies on each other and could not be optimized separately.

2. RELATED WORK

There has been a significant research in the past to study the classic problem of task scheduling using the dag model [3],[4],[5],[7],[10],[11].The Multiprocessor scheduling environment uses more than one processor to execute its processes.

Its structure is either homogenous or heterogeneous.

1. Homogenous system: consists of the processors identical in terms of their functionality.
2. Heterogeneous system: consists of different processors that are capable of performing different tasks.

Various scheduling strategies found in literature are presented below:
Blazewicz et al. investigated the problem of scheduling a set of independent parallel tasks to identical processors under preemptive and non-preemptive scheduling assumptions [1], [16]. Du and Leung also explored the same problem but with more flexibility.
Wang and Cheng further extended the model [14]. They devised a list scheduling approach.

A. Preemptive Scheduling vs.Nonpreemptive Scheduling:
In preemptive scheduling, the execution of a task may be interrupted so that the unfinished Portion of the task can be reallocated to a different processor [15], [11], [12]. On the contrary, algorithms assuming nonpreemptive scheduling must allow a task to execute until completion on a single processor.

B. Parallel Tasks vs. Nonparallel Tasks:
A parallel task is a task that requires more than one processor at the same time for execution [17]. It investigated the problem of scheduling a set of independent parallel tasks to identical processors under preemptive and non preemptive scheduling assumptions. A task can be scheduled to no more than a certain predefined maximum number of processors.

C.Local vs. global
Local scheduling is used in scheduling concurrent processes to the time slices of a single processor. Global scheduling is the assignment of tasks to processors in parallel systems.

D.Static vs. dynamic
In static scheduling, all information regarding the precedence constrained task graph is known beforehand and is fixed [1]. In dynamic scheduling the task graph topology and labels are not known before the program executes due to branches and loops [5].

E. Adaptive vs. non-adaptive
A non-adaptive scheduler is a scheduler that doesn’t change its behavior according to the feedback from the system.
In contrast, adaptive scheduler changes its scheduling decisions according to previous and current behavior of the system.

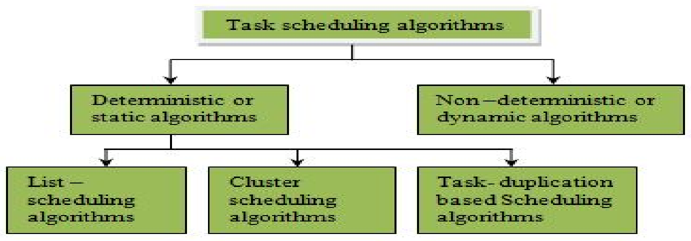

Task scheduling algorithms
Deterministic or static algorithms
Non–deterministic or dynamic algorithms
List – scheduling algorithms
Cluster scheduling algorithms
Task- duplication based Scheduling algorithms

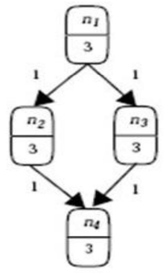
Figure 1 Classification of algorithms

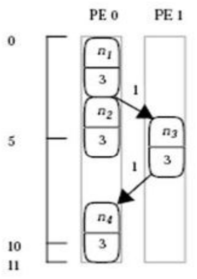
2.1. List-Scheduling Algorithms
One of the earliest proposed solutions to the task scheduling problem is the list-scheduling algorithms [19]. Interestingly, a majority of task scheduling algorithms developed for the homogeneous and the heterogeneous processors belong to the class of list-scheduling algorithms. This is probably due to their relative simplicity and low complexity in comparison with other approaches. The basic idea in the list-scheduling is to assign priorities to the tasks of the DAG and place the tasks in a list arranged in decreasing order of priorities. Some of the list-scheduling algorithms are: ISH *(Lewis et al.*) [20], HLFET, MCP (*proposed by Wu and Gajski*), ETF (*Hwang et al.)* etc.

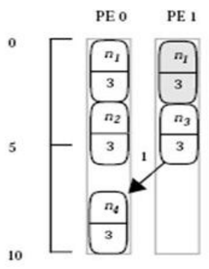
2.2. Clustering Algorithms
In parallel and distributed systems, clustering is an efficient way to reduce communication delay in DAGs by grouping heavily communicating tasks to the same labeled clusters and then assigning tasks in a cluster to the same processor. The clustering algorithms in general have two phases: the task clustering phase that partitions the original task graph into clusters and a post-clustering phase. Task cluster could be linear or non-linear**.** Some of these algorithms are: LC (*Kim and Browne),* EZ (*sarkar*), TRIPLET etc.

3. TASK DUPLICATION BASED SCHEDULING
The task duplication scheduling provides greater efficiency and minimum make span time as compared to other scheduling techniques [3], [7], [10], [14]. The main idea behind the task duplication based scheduling is utilizing processor idle time to duplicate predecessor tasks. This can avoid the transfer of data from a predecessor to a successor

thus reducing the communication cost, network overhead and potentially reduce the start times of waiting task. Task duplication-based scheduling is much useful for systems having high communication latencies and low bandwidths. Task duplication-based scheduling algorithms have been developed for homogeneous processors and heterogeneous processors in the past [10], [11].







(a) (b) (c)

Figure 2 (a) A simple task graph (b) Without duplication (c) With duplication(adapted from [21])

4. DESIGN METHODOLGY

Design methodology in this paper explains about the problem, its representation and the basic technique used in task duplication based scheduling. The detail is as follows:

4.1 Problem Definition

Multiprocessor task scheduling is applied to a wide variety of problems for which there are multiple tasks to be performed on a number of processors. The tasks considered here are non-preemptive. The goal is to schedule these tasks on the processors in (the minimum execution time is referred to as makespan) while meeting the required constraints. Task Duplication Based Scheduling is the best technique to achieve minimum makespan. Since the tasks to be scheduled have characteristics that are known a-priori , then their relationship can be represented via a DAG, G=(V,E) where G is a graph with V nodes representing tasks and E as edges representing prerequisite constraints and communication links. The constraints and the values the problem is attempting to consider consist of following:
1. The system contains a series of limited, fully connected homogeneous processors.
2. Uniform communication cost exists between processors.
3. Each task has some prerequisite constraints that need to be satisfied. It means that given task cannot be executed until its prerequisite tasks are first performed.
4. Task duplication is allowed. This can help in minimizing communication costs. That is if task and its predecessor are on the same processor, then there is no communication cost to compute for their execution.

4.2 Problem Representation

A parallel program can be represented by a directed acyclic graph (DAG)
G = (V, E). The weight of a node ni is called the computation cost and is denoted by w (ni) [2].The edges in the DAG, each of which is denoted by (ni, nj), correspond to the communication messages and precedence constraints among the nodes. The weight of an edge is called the communication cost of the edge and is denoted by c (ni, nj). The source node of an edge is called the parent node while the sink node is called the child node.
A problem is represented as a dag below with 9 nodes.

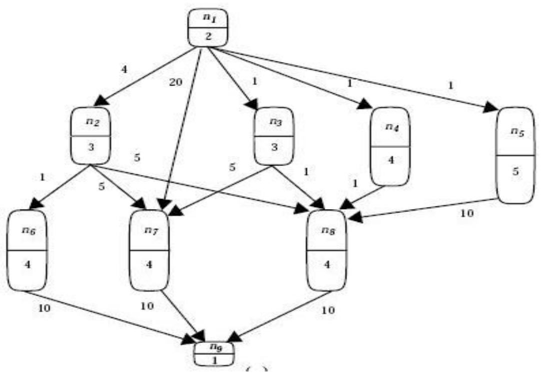


Figure3. Dag model (adapted from [21])

4.2.1 Accurate model- In an accurate model, 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 upon the node allocation and the network topology

4.2.2 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.

4.2.3 Approximate model 2: In this model, the message receiving time and sending time are ignored in addition to approximating the edge weight by a constant.

4.3 Basic techniques in DAG scheduling

Most scheduling algorithms are based on the so-called list scheduling technique [15], [2], [5], [8], [9], [12]. The basic idea of list scheduling is to make a scheduling list (a sequence of nodes for scheduling) by assigning them some priorities and then repeatedly execute the following two steps until all the nodes in the graph are scheduled:

1. Remove the first node from the scheduling list.
2. Allocate the node to a processor which allows the earliest start-time.

There are various ways to determine the priorities of nodes, such as HLF (Highest Level First) [15], LP (Longest Path) [2], and LPT (Longest Processing Time) [6], [11] and CP (Critical Path) [12].

Recently a number of scheduling algorithms based on a dynamic list scheduling approach have been suggested [6], [8], [13]. In a traditional scheduling algorithm, the scheduling list is statically constructed before node allocation begins, and most importantly, the sequencing in the list is not modified. In contrast, after each allocation, these recent algorithms recomputed the priorities of all unscheduled nodes, which are then used to rearrange the sequencing of the nodes in the list.

Two frequently used attributes for assigning priority are the t-level (top level) and b-level (bottom level) [9], [17]. The t-level of a node ni is the length of a longest path (there can be more than one longest path) from an entry node to ni (excluding ni). Here, the length of a path is the sum of all the node and edge weights along the path. The b-level of a node ni is the length of a longest path from ni to an exit node. A critical path (CP) of a DAG, which is an important structure in the DAG, is a longest path in the DAG.

After the scheduling list is constructed by using the node priorities, the nodes are then scheduled to suitable processors. Usually a processor is considered suitable if it allows the earliest start-time for the node. However, in some sophisticated scheduling heuristics, a suitable processor may not be the one that allows the earliest start-time.

5. TDB SCHEDULING ALGORITHMS

A few existing task duplication-based scheduling algorithms developed for the homogeneous processors and heterogeneous processors are briefly described below:

5.1. Task Duplication-based Algorithms for Homogeneous Processors

Few task duplication scheduling algorithms for homogeneous processors are as:

5.1.1 Duplication Scheduling Heuristic (DSH) Algorithm. *Kruatuachue* [20] proposed the DSH algorithm as an extension to the ISH algorithm, which was the first algorithm to utilize task duplication [2],[10],[14]. The DSH algorithm uses the idea of list-scheduling combined with task duplication to reduce the makespan [9]. Tasks are given priorities using static b-level to indicate the urgency of being scheduled.

5.1.2. Bottom-up Top-down Duplication Heuristic (BTDH) Algorithm. *Chung et al.* Proposed the BTDH algorithm an extension of the DSH algorithm described above [2],[10]. The major improvement brought by the BTDH algorithm over the DSH algorithm is that the former keeps on duplicating ancestors of a task even when the duplication time slot is filled up and the start time of the task under consideration temporarily increases [6]. The BTDH algorithm also uses static level for priority assignment.

5.1.3. Linear Clustering with Task Duplication (LCTD) Algorithm. *Shirazi at al.***,** proposed the LCTD algorithm. The LCTD algorithm first iteratively clusters tasks into larger tasks [2], [7], [10], [14]. At each step, tasks on the longest path are clustered and removed from the task graph.[6] This operation is repeated until all tasks in the graph are removed. After performing the clustering step, the LCTD algorithm identifies those edges among clusters that determine the overall completion time. The algorithm then attempts to duplicate the parents corresponding to these edges to reduce the start times of some tasks in the clusters. Linear clustering prematurely also constraints the number of processors used.

5.1.4. Critical Path Fast Duplication (CPFD) Algorithm. *Ahmad and Kwok* proposed a duplication-based algorithm called CPFD algorithm [2], [7], [10], [14]. Initially the tasks nodes in the task graph were classified into three categories namely: Critical Path Nodes (CPN). In-Branch Nodes (IBN) and Out-Branch Nodes (OBN). CPN nodes are the nodes that lies on the critical path and these nodes are most important nodes because their finish times effectively determine the final makespan.

Table 1. Partial taxonomy of TDB scheduling algorithms for homogeneous processors

Algorithm	Processor	Observation/ limitation
DSH	Unbounded	Duplicates immediate predecessor tasks when duplication reduces
BTDH	Unbounded	Duplicates predecessor tasks even if the duplication increases the EST of a task, but at later stages the duplication decreases some other dependent tasks. Higher complexity and proved better than HLFET and ETF.
LCTD	Unbounded	In the context of TDB scheduling, linear clustering prematurely constraints the numbers of processors are used. The start time of some critical nodes can be decreased by using a processor on which all its ancestors are duplicated.
CPFD	unbounded	Task in the CP, are duplicated as early as possible by considering precedence constraints.

5.2 Task Duplication-based Algorithms for Heterogeneous Processors
A fair amount of research work has been carried out for the development of task duplication-based scheduling algorithms for heterogeneous processors in the past and consequently few algorithms were developed. Some of them are:
5.2.1 Heterogeneous Critical Parents with Fast Duplicator (HCPFD) Algorithm. Hagras et al. proposed the HCPFD algorithm. The HCPFD algorithm aims to achieve high performance and low complexity. The algorithm consists of two phases, a listing-phase, which is a simplified version of the CNPT heuristic for heterogeneous environments and suggested low complexity duplication mechanism as a machine assignment phase**.** The machine assignment phase simply selects the machine p that minimizes the Task Finish Time (TFT) of v, and duplicates its critical parent at the idle time between v, and the previous task on p. if this time slot is enough thus duplication will reduce the Task Start Time (TST) of v, on p.

5.2.2. Dynamic Critical Path Duplication (DCPD) Algorithm. Liu et al. proposed the DCPD algorithm. The DCPD initially computes the average execution rates and the average communication rates for all heterogeneous processors. The algorithm initially assumes that each task is assigned to one virtual processor and that the communication overhead between tasks is the average communication rate time the communication volume between tasks.
Then the DCPD scheduling algorithm estimates b-levels for all tasks by using the average execution rates for all heterogeneous processor. In the scheduling process, a task is called examined after it is scheduled to some processor and unexamined before it is scheduled to some processor. The tasks without predecessor are in the ready set.

Table 2: Partial Taxonomy of TDB Scheduling Algorithms For Heterogeneous Processors

Algorithms	Processors	Observations/ limitations
HCPFD	bounded	Compared with non –duplication based algorithms HEFT, CPOP and FLB and proved better than other algorithms.
DCPD	bounded	Outperforms the CPNT, TANH and HCPFD algorithms but redundant task duplications.

6. PERFORMANCE METRICS

6.1. Makespan

The scheduling length is called makespan. The shorter the makespan the better is the algorithm. Makespan is calculated by measuring the finishing time of the exit task by the algorithm.

6.2. Scheduling length ratio (SLR)

The time taken to execute tasks on a critical path is the lower bound of the schedule length. To normalize the schedule length to the lower bound, the SLR is defined as:

SLR = (makespan/critical path)

6.3. Speed up

Speedup 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.

$$S_P = (\sum^{n}_{i=1} T_i) / T_p$$

Where $\sum^{n}_{i=1=} T_i$: sum of computational time of tasks (Ti) in sequential order = 1, 2, 3…n.
T_p : total parallel execution time or scheduling length of a DAG.

6.3. Efficiency

The efficiency of a parallel program is a measure of processor utilization.

$$EFF = S_P \, / \, N_P$$

S_P : Speed up and N_P: Number of processors

6.5 Load Balancing

The load balancing is calculated by the ratio of scheduling length to average execution time over all processors.

L Bal = SL/ Avg

Where **SL**: Scheduling length.
Avg: It is ratio of sum of processing time of each processor and numbers of processors are used.

7. CONCLUSION and FUTURE WORK

Multiprocessor task scheduling is a rich area demanding the application of efficient methods to minimize the task computation time in real-world applications. The multiprocessor scheduling can be explained with DAG model. There are many task duplication based scheduling algorithms as studied in this paper. The task duplication based scheduling provides greater efficiency and minimum make span time as compared to other scheduling techniques. To further improve its performance many heuristics have been used by various researchers. Its future work can be extended to dynamic heterogeneous systems more suitable for real-time applications.

REFERENCES

[1] Y.C.Chung and S.Ranka, "Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithms on Distributed–Memory Multiprocessors", Proceedings of Supercomputing'92, Nov.1992, pp.512-521.(conference style)

[2] I. Ahmad, Y.-K. Kwok, "Analysis, Evaluation and Comparison of Algorithms for Scheduling Task graphs on parallel processors" IEEE Proc. on parallel Architectures, Algorithms, and Networks", pp.207-213, 1996 (conference style).

[3] Koichi.Askura, B.Shao, and T.Watanabe "A Task Duplication Based Scheduling Algorithm for Avoiding Useless Duplication", Nagoya University, pp.45-15(2003).(journal style)

[4] I. Ahmad, Y.-K. Kwok "Benchmarking the Task Graph Scheduling Algorithms".IPPS/SPDP, 1999(conference).

[5] G.N Srinivasa, Bruce R. Musicus. "Generalized Multiprocessor Scheduling for Directed Acyclic Graphs" In

Third Annual ACM Symposium on Parallel Algorithms and Architectures 1994, pp237-246(symposium).
[6] G.L.Park, Behrooz Shirazi, Jeff Marquis and H.Choo. "Decisive Path Scheduling: Anew List Scheduling Method" Communication of the ACM, vol.17, no.12, Dec1974, pp 472-480.(journal style)
[7] S.Ranweere and D.P.Agarwal. "A Task Duplication Based Scheduling Algorithm for heterogeneous systems". In proc. International Parallel and Distributed Processing Symposium,pp445-450,May 2000(symposium)
[8] Min You Wu "On Parallelization of static scheduling Algorithms", IEEE Transactions on Parallel and Distributed
Systems, Vol.10, No.4, April 1999, pp 517-528.(conference style)
[9] Ishfaq Ahmad, Y.-K. Kwok. "static scheduling algorithms for allocating DAGs to Multiprocessors" Hong Kong Research Grants Council July1998,pp6-39
[10] Literature survey(book chapter style)
[11] Ranjit Rajak "A Novel Approach for Task Scheduling in Multiprocessor System", International Journal Of Computer Applications(0975-8887)Volume 44 No.11, April 2012. (journal style)
[12] M.A. Mouhamed, "Lower Bound on the Number of Processors and Time for Scheduling Precedence Graphs with Communication Costs", IEEE Transactions on Software Engineering, Vol.16, No.12,Dec 1990, pp 1390-1401(conference style)
[13] F.D.Anger, J.J .Hwang and Y.C.Chow, "Scheduling with sufficiently loosely Coupled Processors", Journal on Parallel and Distributed Computing, Vol.9, 1990, pp 87-92. (journal style)
[14] I. Ahmad, Y.-K. Kwok. "On exploiting task duplication in parallel program scheduling", IEEE Proc. Vol.9, 1998. (conference style)
[15] T. Hagrs, J.Janecek "Static vs. Dynamic List–Scheduling Performance Comparison" Acta Polytechnica Vol.43 No. 6/2003, pp 1415-1420. (journal style)
[16] I.De Falco, R.Del Balio, E.Tarantino, "An Analysis of Parallel Heuristics for Task Allocation in Multicomputers", Computing: Archiv fur Informatik und Numerik, Vol.59, no.3, 1997, pp.259-75. (journal style)
[17] J.Du andJ.Y.T.Leung. "Complexity of scheduling Parallel Task Systems", SIAM Journal on Discrete Mathematics,
vol. 2, no.4, 1989, pp.473-487. (journal style)
[18] H.El.Rewini, H.H.Ali andT.G.Lewis. "Task Scheduling in Multiprocessing Systems", IEEE Computer, Dec.1995,
Pp-27-37(journal style)
[19] T.L. Adam, K.M. chardy and J.R.Dickson, "Comparison of list schedules for parallel processing systems" , communications of the ACM, vol.17,no.12, dec 1974,pp.685-690(journal style)
[20] B.Kruatachue and T.G.Lewis, "Duplication scheduling Heuristics (DSH): A New precedence Task Scheduler for Parallel Processor Systems," Technical Report, Oregon State Univ. Corvallis, OR 97331, 1987.(technical report style)
[21] Ishfaq Ahmad and Yu-Kwong Kwok, "A Comparison of Task-Duplication-Based Algorithms for Scheduling Parallel Programs to Message-Passing Systems", Hong Kong Research Grants Council.