Scheduling Tasks in DAG to Heterogeneous Processor System

Wai-Yip Chan and Chi-Kwong Li
Department of Electronic Engineering,
The Hong Kong Polytechnic University, Hong Kong.

Abstract

Heterogeneous processors configuration in parallel and distributed becomes a practical solution in modern parallel and distributed system. In order to execute tasks in such system with better performance, scheduling algorithms which support the configuration are needed. This paper starts with studying a technique called Heterogeneous List Scheduling Heuristic (HLS) for designing scheduling algorithm to schedule tasks into heterogeneous systems. With this, an experience of designing scheduling algorithm for scheduling task into heterogeneous system is described. This is done by modifying an algorithm called Relative Mobility, which is proposed by Chan and Li [2][3] for scheduling task into homogeneous system, to propose an algorithm called Heterogeneous Relative Mobility Scheduling algorithm (HRMS). Finally, an experiment is conducted to show some important properties as scheduling tasks into different configurations of processors.

1 Introduction

A better schedule of tasks running in parallel and distributed systems can improve performance of using these systems undoubtedly. The objectives of scheduling algorithms are to allocate tasks into processors and to arrange their execution orderly so that data dependencies are satisfied with the length of schedule produced (parallel time) is minimized. It is proved that scheduling problem is NP-complete [4] and obtaining optimal scheduling solution in low time complexity is not easy. In order to reduce complexity, heuristic approaches are being used in scheduling algorithms design that can produce near optimum scheduling solutions in lower time complexity [4].

The List Scheduling heuristic (LS) is a fundamental class of heuristic scheduling algorithm for scheduling DAG (Direct Acyclic Graph). Many algorithm designs are based on this for scheduling tasks into homogeneous system. They show great success in scheduling tasks from DAG to homogeneous case of parallel and distributed systems. However, owing to the growth of VLSI technology and the nature of practical processors configurations in resent years, parallel and distributed systems are configured in different rank of processor's performance. There is no longer guarantee that the system must be homogeneous. In this case, the processor speed and data link rate may not be identical. This is called heterogeneous system. Unfortunately, the existing scheduling algorithms are no longer valid in such system.

To design and implement algorithm for scheduling task into heterogeneous system, a technique called "Heterogeneous List Scheduling (HLS)" was proposed by Chan and Li [1] as a framework. The technique suggests how to determine and use heuristic in the classical List Scheduling heuristic. Consequently, scheduling algorithms for heterogeneous system can be easily designed by plotting those from classical List Scheduling heuristic approaches. This paper begins with a brief study of the technique. Based on the technique, an algorithm called Heterogeneous Relative Mobility algorithm (HRMS) is proposed to schedule tasks from DAG into heterogeneous processors system. Lastly, important properties as well as performance of the algorithm for scheduling tasks into heterogeneous processors configuration are studied.

2 The Heterogeneous List Scheduling heuristic (HLS)

The "List Scheduling Heuristic" has been adopted by most designs of heuristic scheduling algorithms. The basic principle is that all tasks in DAG are assigned priority which are used to generate a priority list. Based on the priority list, the tasks with the highest priority are scheduled into a processor in accordance with different rules of processor assignment. Although the approach is capable of designing algorithms for scheduling task into homogeneous processors system, two problems exist while scheduling task into heterogeneous counterpart. Firstly, the priority value and dependence constrain cannot be determined prior to the actual task assignment. As a result, priority list cannot be generated and the highest priority task cannot be selected as usual. Secondly, since

0-8186-8332-5/98 $10.00 © 1998 IEEE

the dependence constrain cannot be directly obtained from the entry DAG. The usual heuristic of task to processors assignment in List Scheduling heuristic cannot be applied straightly.

In these occasions, the approach of list scheduling heuristic cannot be directly used to scheduling task into heterogeneous system. Chan and Li analyzed the situation and proposed a solution called "Heterogeneous List scheduling" technique [1]. The algorithm 1 shown below is an algorithmic description of the technique. The technique solves the first problem by using a normalized reference value of processor speed and communication rate in determination of priority values. Thus, a priority list can be generated for task selection. For the second problem, the technique suggests those heuristics of tasks to processors assignment should consider the actual speed of assignment processors and rate of communication links when assigning task to processors. Hence, data dependence constrain is satisfied and tasks can be executed in a proper order.

1. Each node in the task graph is assigned a priority. A priority queue is initialized for ready tasks by inserting every task that has no immediate predecessors. Tasks are sorted in descending order of task priorities. (To determine priority for those tasks which have been scheduled to processors, the priority values and dependence constrains are determined by the actual scheduled processor speed and communication rate. On the other hand, for those task which have not been scheduled, the values of processor speed as well as communication rate are calculated by using a normalized uniform speed and rate as a reference value of measurement respectively.)
2. As long as the priority queue is not empty to do the following:
   2.1 A task is obtained from the front of the queue.
   2.2 The "best" idle processor is selected to run the task. (The decision is made by considering the actual speed of assignment processors and rate of communication links.)
   2.3 Recalculate the priority values following the step 1. When all the immediate predecessors of a particular task are executed, that successor is not read and can be inserted into the priority queue.

Algorithm 1, The Heterogeneous List Scheduling technique.

3 The Heterogeneous Relative Mobility Scheduling algorithm (HRMS)

This section proposes an algorithm called Heterogeneous Relative Mobility Scheduling algorithm (HRMS) for scheduling task from DAG into heterogeneous processor system. The design is based on scheudling heuristics using in Relative Mobility Scheduling algorithm (RMS) [2][3]. They include Relative Mobility ($M_r(n_i)$), which is defined as $M_r(n_i) = M(n_i)/W(n_i)$, as heuristic of priority list generation and Condition 1 shown in [2] as heuristic of task to processors assignment. Hence, with applying the HLS technique, a flexible and low complexity scheduling algorithm is designed.

An algorithmic description of the HRMS algorithm is shown in algorithm 2a-b. The algorithm first sorts the set of given processors in a decreasing order of speed shown in step 1 of algorithm 2b. This step identifies the slowest speed processor to obtain a normalized factor ε for calculating task priority. Then the *Extended Relative Mobility (EM_r)*(for which will be defined in the following sections) of each un-scheduled task in the task graph is determined in step 2. Then a priority list of free task L' is generated in ascending order of EM_r. Hence the highest priority free task can be selected from the list for scheduling. To schedule a task into the "best" suitable processor, condition H shown in algorithm 2b are used. They are used to find a processor that the task can be executed with the shortest finish time. Finally, the steps 2 to 4 are repeated until all tasks in the task graph are scheduled.

Assume a task n_p is examined to be scheduled to P_m, to which l tasks, n_{m_1}, n_{m_2}, ..., n_{m_l}, have been scheduled. If the moving intervals of these tasks do not intersect with the moving interval of n_p, then n_p can be scheduled on P_m. Otherwise, assume the moving intervals of tasks n_{m_i}, $n_{m_{i+1}}$, ..., n_{m_j} ($1 \le i \le j \le l$) intersect the moving interval of n_p. n_p can be scheduled to P_m, if there exists k ($i \le k \le j+1$),

$$W(n_p)_{P_m} \le min(T_F(n_p), T_L(n_{m_k})) - max(T_S(n_p), T_S(n_{m_{k-1}}) + W(n_{m_{k-1}})_{P_m}.$$

Where $W(n_i)_{P_m}$ is the cost function of a node running in P_m and, The T_F, T_S and T_L computation should assume that task n_p is executed on P_m; and When n_p is scheduled to P_m, n_p is inserted before the first task in the task sequence of P_m that satisfies the inequality listed above; and

if $n_{m_{i-1}}$ does not exist, $T_S(n_{m_{i-1}}) = 0$ and $W(n_{m_{i-1}}) = 0$; and if $n_{m_{j+1}}$ does not exist, $T_L(n_{mj+1}) = \infty$. Otherwise, the task cannot be scheduled to P_m.

Algorithm 2a, Condition H: Necessary and sufficient condition for scheduling a task into a processor P_m.

1: Sort the speed of processors in descending order from left to right.

2: Calculate Extended Relative Mobility (EM_r) for all tasks. Let L' be the group of tasks in L with minimum relative mobility. Let n_i be a task in L' that does not have any predecessors in L'.

Hu's 1961

3: Using the Condition H, find a processor P_m in which n_i would finish in the earliest time. When n_i is scheduled on P_m, all edges connecting n_i and other tasks already scheduled to P_m are changed to zero.

For two tasks n_i and n_j are independence, if n_i is scheduled before task n_j on same processor P_m. An edge with weight zero is added from n_i to n_j in the graph. If n_i is scheduled after task n_j, add an edge with weight zero from n_j to n_i in the graph. To ensure there is no deadlock, checking if adding the edges result in forming a loop. If so, schedule n_i to the next available space.

4: Recalculate relative mobility for the modified graph. Remove n_i from L and L' and repeat steps 2, 3 and 4 until L is empty.

Algorithm 2b. The HRMS algorithm

3.1 Heuristics used in the HRMS algorithm

There are two types of heuristics used in the HRMS algorithm; the Extended Relative Mobility for priority list generation and the insertion schedule with earliest finish time for task to processors assignment. To generate a priority list of free task, step 2 of the HRMS algorithm subject to the step 1 of the List scheduling heuristic wherein the Extend Relative Mobility (EM_r) is used. The EM_r heuristic in is used to identify an un-scheduled free task that its moving range of starting the task is the shortest. For instance, a task with zero EM_r implies that the task should start its execution without suffering any form of delay. Hence, it is termed as critical tasks in critical path of the scheduling task graph. On the other hands, for those tasks which EM_r greater than zero are called non-critical tasks. These tasks can suffer delay in starting their execution, so they can be scheduled latter. Owing to this argument, a priority list generated by sorting the EM_r in ascending order for prioritize those tasks to be schedule in each scheduling step.

For the heuristic of task to processors assignment, the heuristic of insertion schedule with earliest finish time is used in which is formulated in step 3 of algorithm 2a and condition H in algorithm 2b. That is, the algorithm uses condition H to find processors in the given processors set that can execute the task in any empty time slot of the processors. There will be more than one processor satisfy the condition. To decide which processor is "the best" processor to execute the task. The step 3 of algorithm is used to make the decision that assigning the task to a processor in which can execute the task with earliest finish time. It is found that the assignment mechanism is similar to that of the RMS algorithm. The major difference between them is the ways of moving interval determination in condition H. As mention in HLS technique, the decision of assigning task into heterogeneous system should be made by considering the actual speed of assignment processors and rate of communication links. Thus, to determine the task insertion to the idle interval, the condition H of the HRMS algorithm considering the actual speed of processors and the actual rate of communication.

3.2 An extended concept of relative mobility (EM_r)

It was shown in [2][3] that Relative Mobility (M_r) is used as a heuristic to identify a free task to be scheduled in the RMS algorithm. The M_r is defined in homogeneous processors systems as $M_r(n_i) = M(n_i)/W(n_i)$, where $M(n_i)$ is the mobility of a task n_i and $W(n_i)$ is the computation time of a task n_i. Since the computation speed of processors in homogeneous processors systems are equal, the $M(n_i)$, the $W(n_i)$ and the $M_r(n_i)$ can be simply found in homogeneous processors systems. However, the speed of processors are different in heterogeneous processors systems. The computation time of a task n_i running in different processors are not the same. The $M(n_i)$, the $W(n_i)$ and the $M_r(n_i)$ cannot be identified while a task is not scheduled. Owing to those reasons, the step 1 of the HLS technique suggests using a reference speed to determine the heuristic. Thus, the concept of relative mobility is extended and an extended heuristic called *Extended Relative Mobility (EM_r)* is introduced in the HRMS algorithm.

The EM_r heuristic extends the concept of M_r using as heuristic in heterogeneous processors systems. In order to determine relative mobility in heterogeneous situation, a normalized factor ε, which is described in definition 1 and criterion 1, is used as a reference speed of priority values calculation. This in fact is subject to step 1 of the HLS technique that for those tasks which have not been scheduled into any processors, the calculation is based on assuming the task is running on a processors with computation speed ε. Otherwise, the calculation depends on the actual scheduled proces-

sor. Therefore, the Extended Mobility ($EM(n_i)$), the Cost of a task running in processor P_m ($W(n_i)_{P_m}$) as well as the Extended Relative Mobility($EM_r(n_i)$) are determined through definition 2 to 4 respectively.

Definition 1:
The ε is defined as the least speed of computation among processors inside a given processor set P.

$$\varepsilon = min\{s_i\},$$

where s_1 = speed of processor i, i=$p_0..p_n$, and $p_i \in P$.

Criterion 1:
If a task in a task graph has not been scheduled into any processors, the task is assumed to run on the least performance processors with computation speed ε. Otherwise, the task is running on a processor with the speed of that processor.

Definition 2:
The cost function of a task running in a processor P_m is defined as:

$$W(n_i)_{P_m} = \frac{i(n_i)}{s_{p_m}},$$

where $i(n_i)$=total number of instruction of task n_i.

Definition 3:
The Extended Mobility(EM) is defined as:

$$EM(n_i) = T_L(n_i) - T_S(n_i),$$

where $T_S(n_i)$= earliest start time of n_i and $T_L(n_i)$=latest start time of n_i whom are calculated under criterion 1 and definition 2.

Definition 4:
The Extended Relative Mobility(EM_r) is defined as:

$$EM_r(n_i) = EM(n_i)/Wi(n_i)_{P_{c1}},$$

where P_{c1} is a processor under criterion 1.

3.3 Properties of the HRMS algorithm

Property 1:
In homogeneous processors systems, the Extended Relative Mobility (EM_r) is the Relative Mobility.
Proof: In homogeneous processors systems, all processors are the same in computation speed $s_{p_0} = s_{p_2} = ... = s_{p_m} = \varepsilon$ $(m = p - 1)$. Therefore, $EM_r(n_i) \boxminus M_r(n_i)$.

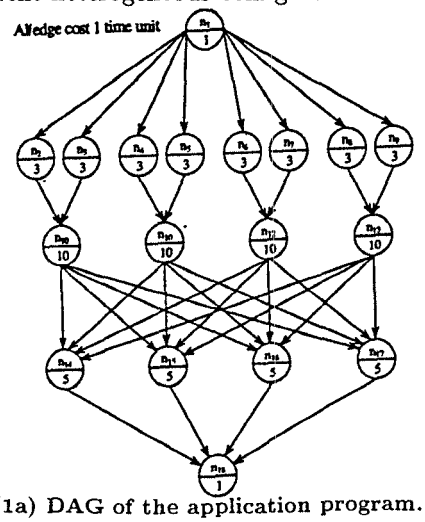
Property 2:
In homogeneous processors systems, the Heterogeneous Relative Mobility Scheduling Algorithm (HRMS) is reduced to the Relative Mobility Scheduling algorithm.
Proof: It is subsequent to property 1.

Property 3:
The task assignment strategy of HRMS algorithm is favorable to schedule tasks into processor ranked from fast processors to slow processors with the goal of minimizing the finishing time.
Proof: It is shown in experimental studies.

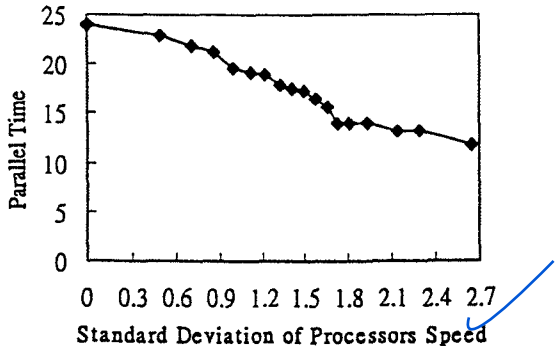
Property 4:
The HRMS algorithm produces a shorter parallel time schedule as the standard deviation of the speed in the processors configuration is larger. In other words, the HRMS algorithm favor in scheduling jobs onto a single high-performance processors.
Proof: It is shown in experimental studies.

4 Experimental Studies : Scheduling in different processor configurations

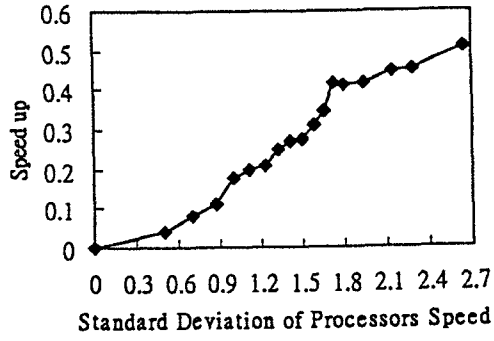
This experiment is carried to study the behavior of the HRMS algorithm as scheduling tasks into different configurations of processors speed. Consider an algorithm of finite element analysis technique applied to atmosphere science application which is represented as DAG shown [5] in figure 1a. The HRMS is used to schedule the graph into eight processors in which the aggregated computational power of these processors are eight instructions per time unit. There are more than 100 processor sets are generated in which the speed of the eight processors are randomly generated. According to the standard deviation of speed of the eight processors in each sample, we analyse the behaviour of the HRMS algorithms as scheduling tasks into different heterogeneous configurations.



(1a) DAG of the application program.



(1b) Parallel time vs. system configuration.



(1c) Speedup vs. system configuration.

Figure 1a-c, the behavior of the HRMS algorithm running in different configuration of processors speed.

Figures 1b and 1c show changes in parallel time and effect of percentage change in speed up as compared with homogeneous processors of the produced schedule running in different standard deviation of processor configurations. It is showned in the experiment that the HRMS algorithm can schedule tasks into different configurations of processors satisfying data dependence and completing in short parallel time. Besides, it is interesting to find out that parallel time of the produced schedule is decreasing and the percentage of increment in speed up is increasing as the standard deviation is increasing. These show that the HRMS algorithm is favorable in scheduling tasks into a processor set with high standard deviation. In other words, the best configuration of the processors set is of few number of high-performance processors. This shows the property 3 and property 4. The HRMS follows the rule of Max-Min that it tries to minimize the communication cost by minimizing parallelism when performance of individual processors are better.

5 Conclusion

A proposed technique of modifying a homogeneous scheduling algorithm to a heterogeneous scheduling algorithm called Heterogeneous List Scheduling Technique (HLST) is described. Based on this technique, a low-complexity heterogeneous scheduling algorithm with a significant performance called HRMS is proposed. It is found from the experiment that the parallel time is improved as standard deviation of the speed of processing element increases. This finding states the fact that maximization of standard deviation of processors speed will result in minimization of parallelism. A Max-min relation not only occur in the trade off between communication and parallelism, but also occur in trade off between configuration of processors and parallelism in heterogeneous processors systems. The proposed algorithm is can schedule parallel tasks into a mixture of high and low performance heterogeneous parallel computers with the trade off between degree of parallelism and processors utilization.

References

[1] Wai-Yip Chan and Chi-Kwong Li, "A New Technique of List Scheduling Algorithm for Heterogeneous Processors Systems," in Proc. ISCA International Conference on Computer Applications in Industry and Engineering, 1997.

[2] Wai-Yip Chan and Chi-Kwong Li, "The Relative Mobility Scheduling Algorithm (RMS)," in Proc. ISCA International Conference on Computer Applications in Industry and Engineering, 1997.

[3] Wai Yip Chan and Chi Kwong Li, "An aggressive Parallel Tasks Scheduling Algorithm: Relative Mobility Scheduling Algorithm (RMS)", will be appeared in IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, August 1997.

[4] M.Y. Wu and D. Gajski, "Hypertool: A programming aid for message-passing systems", *IEEE Trans. Parallel Distributed System*, vol. 1, pp. 330-343, 1990.

[5] Lewis, T.G., "Parallel programming support environment research," Technical Report, Oregon Advanced Computing Institute, 1989.