A Static Scheduling Heuristic for Heterogeneous Processors

Hyunok Oh and Soonhoi Ha

The Department of Computer Engineering, Seoul National University, Seoul, 151-742, Korea: e-mail: {oho,sha}@comp.snu.ac.kr

This paper presents a static scheduling heuristic called best-imaginary-level (BIL) scheduling for heterogeneous processors. The input graph is an acyclic precedence graph, where a node has different execution times on different processors. The static level of a node, or BIL, incorporates the effect of interprocessor communication (IPC) overhead and processor heterogeneity. The proposed scheduling technique is proven to produce the optimal scheduling result if the topology of the input task graph is linear. The performance of the BIL scheduling is compared with an existing technique called the general dynamic level (GDL) scheduling with various classes of randomly generated input graphs, resulting in about 20% performance improvement.

1 Introduction

Thanks to the rapid development of networks, the distributed systems become a promising workhorse to increase the computing power. These systems are usually heterogeneous systems with significant overhead of interprocessor communication (IPC). The schedulers for such heterogeneous systems need to account for IPC overhead and processor heterogeneity: a task takes different execution times on different processors.

The proposed scheduling takes an acyclic precedence graph (APG) as an input task graph in which a node represents a task and a directed arc is associated with a number that specifies the amount of IPC overhead. We assume that the execution time of a node N_i on a processor P_j, $E(N_i, P_j)$, is known at compile-time for each node-processor pair. If node N_i cannot be executed on processor P_j, $E(N_i, P_j)$ is infinite. An example APG and its node execution-time table is shown in Fig. 1 (a). We also assume that interprocessor communication time can be overlapped with computation time in a schedule.

The scheduling objective in this paper is to minimize the scheduling length or makespan of the input APG. Since this scheduling problem is NP-hard in the strong sense [2], we will rely on heuristics. As an existing scheduling heuristic for heterogeneous scheduling problem, we will consider the General Dynamic Level (GDL) scheduling technique developed by G.C.Sih and E.A.Lee [1]. We will show that the proposed technique is more algorithmically appealing to generate

* This research is supported by the S.N.U. Research Fund

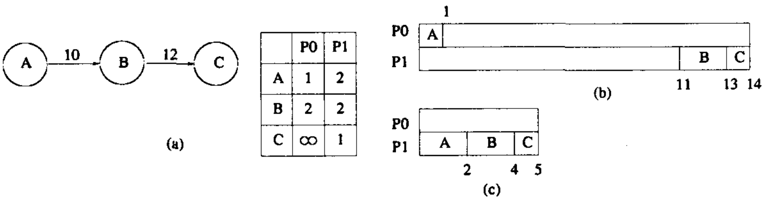
an optimal schedule for a special class of APG. Moreover, we obtain about 20% performance improvement over GDL with randomly generated APGs.

2 GDL Scheduling Technique

Both GDL and the proposed BIL scheduler are based on the list scheduling idea[3]. Each node is assigned a priority, or the static level of the node. The list scheduling schedules the runnable nodes in the decreasing order of priority. The variants of list scheduling techniques differ in terms of how to assign priorities to the nodes and on which processor a selected node is to be scheduled [4].

A popular choice of the static level is the critical path length of a node to a terminal node. To compute the critical path in a heterogeneous system, the GDL scheduler defines the assumed execution time of node N_i, denoted $E^*(N_i)$, as the median execution time of the node over all processors. While the IPC overhead is not considered in the static level computation, the GDL scheduler considers it by adjusting the level, or the dynamic level, when the node becomes runnable. Also, it includes the effect on the child node and the resource scarcity cost in the dynamic level computation to overcome a major drawback of the list scheduling algorithms: fail to consider the global effect of the current scheduling decision.

Consider an example shown in Fig. 1 (a). The scheduling result from the GDL heuristic is represented in Fig. 1 (b) with a Gantt chart that represents on which processor and at which time each node is scheduled. When node A is scheduled, the GDL still fails to consider the effect of the processor selection on node C. Therefore, node A is scheduled on processor P0. As a result, it suffers from the huge IPC overhead between A and B since node B should be scheduled to processor P1 considering node C.



	P0	P1
A	1	2
B	2	2
C	∞	1

Fig. 1 An example to show the effect of descendant consideration

3 The Proposed BIL Scheduler

We define the static level, the best imaginary level (BIL), of a node as follows.

Definition 1

$$BIL(N_i, P_j) = E(N_i, P_j) + \max_{d_l \in D(N_i)} [\min(BIL(d_l, P_j), \min_{k \neq j}(BIL(d_l, P_k) + d(N_i, d_l))]$$

where $d(N_i, d_l)$ means the amount of IPC overhead between N_i and d_l which is in the descendant $D(N_i)$.

The BIL of node N_i indicates the critical path length including the IPC overhead assuming that the node is scheduled on processor P_j, based on the critical assumption that all descendant nodes can be scheduled at the best times. Since it is not always possible, we use the term *best imaginary*. Since it considers the IPC overhead in its computation, the BIL of a node can be thought as the global information of all descendant nodes under the optimistic assumption. The BIL scheduler shows a drastic improvement with the example of Fig. 1 (a) as shown in Fig. 1 (c). Node A is now scheduled on processor P1 since the scheduler considers the effect of the far descendant C reflected in the BIL computation of node A. In fact, the BIL scheduler produces the optimal scheduling result if the APG is linear.

Theorem 1. *The BIL scheduler produces the optimal scheduling result if the APG is linear*

We can easily prove by induction that the $BIL(N_i, P_j)$ indicates the shortest finish time starting from node N_i to the terminal node assuming that node N_i is scheduled on processor P_j. The optimal schedule length is nothing but the minimum value of the BIL values of the starting node, N_1. In other words, $\min_j BIL(N_1, P_j)$ is the optimal scheduling length. Since the $BIL(N_1, P_j)$ contains the information on which processor the child node is to be scheduled, the BIL scheduler completes the schedule once it determines the minimum value of the BIL value of node N_1.

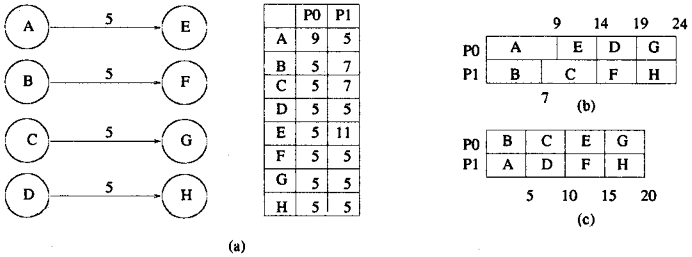
3.1 Node Selection

As the scheduling proceeds, we adjust the level of a node N_i on processor P_j to measure the best imaginary makespan, BIM. BIM is defined as follows: $BIM(N_i, P_j) = T(N_i, P_j) + BIL(N_i, P_j)$, where the node N_i cannot be scheduled on processor P_j before $T(N_i, P_j)$. Note that a runnable node has N different BIM values, one for each processor, if the total number of processors is N.

Among the runnable nodes, we select a node to be scheduled aiming to minimize the performance penalty based on a pessimistic assumption for each node. Suppose that the number of runnable nodes at a scheduling stage is k. We define the priority of a node as the k-th smallest BIM value or the largest finite BIM value if the k-th smallest BIM value is undefined. In case more than one node have the same priority, we adopt a tie breaking policy in a recursive form: we compare the $(k-1)$-th BIMs of nodes that have the same k-th BIM.

3.2 Processor Selection

The next step is to determine the optimal processor for the selected node. Even though we are apt to select the processor associated with the smallest BIM value, this selection scheme may not be optimal since the BIL of a node is pessimistic assuming the IPC overhead is visible. In case the parallelism of the APG is high, the execution time becomes the more important factor than the IPC overhead since the IPC overhead is likely to be hidden. Consider the example of Fig. 2 (a). When node A is selected, the number of runnable nodes is 4 and the number of processors is 2. Therefore, we may expect that on the average two nodes will be scheduled on each processor before its child node is scheduled. It means that the IPC overhead between nodes A and E is likely to be hidden while the BIL fails to account for this fact as shown in Fig. 2 (b). Therefore, we define the revised BIM as follows.



	P0	P1
A	9	5
B	5	7
C	5	7
D	5	5
E	5	11
F	5	5
G	5	5
H	5	5

Fig. 2 An example APG to demonstrate the effect of graph parallelism

$$BIM^*(N_i, P_j) = T(N_i, P_j) + BIL(N_i, P_j) + E(N_i, P_j) * max[\frac{k}{N} - 1, 0]$$

Using this revised BIM value, we obtain the better result as shown in Fig. 2 (c). If more than one processor have the same revised BIM value, we select the processor that makes the sum of the revised BIM values of other nodes on the processor maximum.

4 Experiments

To estimate the performance of the BIL scheduler, we used a random APG generator with various numbers of nodes and processors as well as graph parallelism. We performed 100 experiments with 30, 50, 100, 150, 200, 250, 300, 350, 400 and 500 nodes on 2, 3, 5, 7, 10, 12, 15, 20, 25 and 30 processors. It is repeated 10 times with different seeds for random number generation for each pair of nodes and processors and averaged the results. We shows the performance improvement over the GDL in Fig. 3 by varying the ratio of the IPC overhead and the average execution times of nodes by 0.5, 1 and 2. Let $\bar{C}$ be the mean IPC overhead and $\bar{E}$ be the average node execution time. For instance, $\bar{C} = 0.5 * \bar{E}$ means that the IPC overhead is a half of the average execution time. We observed that that the performance improvement of the BIL scheduler becomes significant when the number of processors is greater than 2. We conjecture the reason of performance difference two-folds. First, as the number of processors grows, the effect of graph parallelism would be decreased, so that the BIL scheduler considers

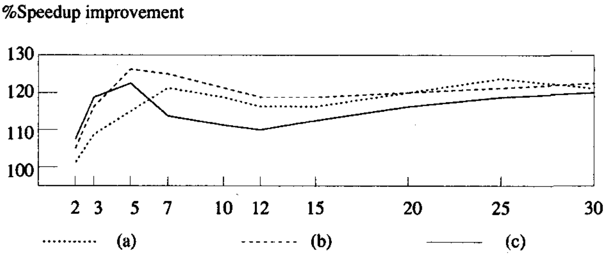


Fig. 3 Percentage improvement of the BIL scheduler over the GDL scheduler. (a) $\bar{C} = 0.5 * \bar{E}$ (b) $\bar{C} = \bar{E}$ (c) $\bar{C} = 2 * \bar{E}$.

the IPC overhead more effectively. Second, the GDL examines only two processors for each node to consider the global effect (resource scarcity) of the current scheduling decision.

The time complexity of this technique is $O(n^2 p \log p)$ in total, where n is the number of nodes and p is the number of processors. While it is smaller in order than the complexity of GDL, our experiments revealed that it takes about the same time with the GDL scheduler because the coefficients of the BIL is larger.

5 Conclusion

We have proposed a non-preemptive static scheduling heuristic called Best-Imaginary-Level (BIL) scheduling for heterogeneous processors. The proposed scheduling technique is proven to produce the optimal scheduling result if the topology of the input task graph is linear. The performance of the BIL scheduling was compared with an existing technique called the general dynamic level (GDL) scheduling with various classes of randomly generated input graphs, resulting in about 20% performance improvement.

The proposed scheduling is expected to be applicable to the large span of target architectures from the network computing to the hardware/software codesign. Currently, we are building an environment to execute an input program graph on such heterogeneous systems. For more general applicability, the heuristic needs to be extended to consider the effect of resource limitation.

References

1. G.C.Sih and E.A.Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures", *IEEE Trans. parallel and distributed systems*, vol. 4, no.2, pp. 175-187, Feb. 1993
2. J.K.Lenstra & A.H.G Rinnooy Kan, "Complexity of Scheduling under Precedence Constraints", *Operations Research 26, 1* (Jan-Feb 1978)
3. E.Coffman, *Computer and Job-Shop Scheduling Theory*, Wiley, New York, 1976
4. T.L.Adam, K.M.Chandy and J.R.Dickson, "A Comparison of List Schedules for Parallel Processing Systems", *Commun. ACM*, vol. 17, no. 12, pp. 685-690, Dec. 1974