SDBATS: A Novel Algorithm for Task Scheduling in Heterogeneous Computing Systems

Ehsan Ullah Munir[1], Sajjad Mohsin[2], Altaf Hussain[2], Muhammad Wasif Nisar[1], Shoukat Ali[3]

[1]*Department of Computer Science, COMSATS Institute of Information Technology, Wah Cantt. 47040, Pakistan.*
[2]*Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan.*

[3]*IBM Research Dublin, Ireland.*

shoukat.ali@ie.ibm.com
{*ehsanmunir, smohsin*, *altafhussain*,*wasifnisar*}*@comsats.edu.pk*

Abstract- A heterogeneous computing system (HCS) efficiently utilizes the heterogeneity of diverse computational resources interconnected with high speed networks to execute a group of compute intensive tasks. These are typically represented by means of a directed acyclic graph (DAG) with varied computational requirements and constraints. The optimal scheduling of the given set of precedence-constrained tasks to available resources is a core concern in HCS and is known to be NP-complete problem. Task prioritization has been a major criterion for achieving high performance in HCS. This paper presents a SD-Based Algorithm for Task Scheduling (SDBATS) which uses the standard deviation of the expected execution time of a given task on the available resources in the heterogeneous computing environment as a key attribute for assigning task priority. This new approach takes into account the task heterogeneity and achieves a significant reduction in the overall execution time of a given application. The performance of the proposed algorithm has been extensively studied under a variety of conditions on standard task graphs from Graph Partition Archive as well as on some real world application DAGs such as Gaussian Elimination and Fast Fourier Transformation application DAGs. Our results show that SDBATS outperforms well known existing DAG scheduling algorithms in terms of schedule length (makespan) and speedup.

Keywords: Heterogeneous computing, DAG scheduling, Task priority, Schedule length

1. Introduction.

A heterogeneous computing system (HCS) is a computing platform with diverse set of interconnected processors via high speed network to execute parallel applications. Due to diverse computational resources, task scheduling mechanism in HCS has become one of the key factors for achieving high performance computing. Maximum speedup gain through parallelization can be achieved by harnessing computational heterogeneity among the available processors with high quality schedules. The common objective of scheduling is to assign tasks onto available set of machines and order their execution so that task precedence requirements are satisfied with a minimum execution time (makespan).The problem of finding the optimal schedule is shown to be NP-complete [1, 2].

Precedence constrained task can be represented as a directed acyclic graph (DAG) in which the nodes represent tasks and the directed edges correspond to inter-task dependencies, such as a task's precedence. A plethora of heuristics such as clustering algorithms, list scheduling algorithms and duplication based algorithms, have been proposed in the literature [e.g. 7-9, 11-17, 23, 24] for the optimal solution of DAG scheduling problem. The heuristic we propose in this paper belongs to the class of list scheduling heuristics [e.g., 7, 11-14, 16]. The basic idea of list scheduling is to assign priorities to the tasks of the DAG and place the tasks in a list arranged in non-increasing order of priorities. A task with a higher priority is scheduled before a task with a lower priority and ties are broken using deterministic or arbitrary selection schemes.

Some well-known list scheduling algorithms such as *Heterogeneous Earliest Finish Time* (HEFT) [7], *Critical Path on a Processor* (CPOP) [7] and *Performance Effective Task Scheduling* (PETS) [17] prioritize the given set of tasks on the basis of task communication cost and average computation cost. The HEFT algorithm uses a recursive procedure to compute the rank of a task by traversing the graph upwards from

978-0-7695-4979-8/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPSW.2013.259

the exit task and vice-versa for CPOP. The PETS algorithm further divides the communication cost of a task into Data Transfer Cost (DTC) and Data Receiving Cost (DRC). D*ynamic level schedule* (DLS) [11] algorithm assigns the priorities to the tasks using leveled state of tasks and their precedence constraints. *Longest dynamic critical path* (LDCP) [13] assigns priorities by considering the critical path attribute of the given DAG. *Heterogeneous earliest finish time with duplication* (HEFD) [16] uses task variance as a computation capacity heterogeneity factor for setting weights to tasks and edges. Zhao and Sakellario [8] have investigated the rank function of the HEFT algorithm. They have shown that the task prioritization based on the mean computation cost does not always give optimal schedules. All of these above mentioned algorithms and schemes mostly rely on the mean or median values of the computation cost of a given task on the available set of processors and lack the basic notion of heterogeneity which is a key attribute of heterogeneous computing system.

Munir et al [20] used standard deviation of the expected execution time of a task as a criterion for prioritizing independent set of tasks. Their approach generated high-quality schedules. The motivation behind this work is to take into account the heterogeneity of the given task on the available set of processors as the key factor for assigning priorities for dependent set of tasks. The proposed algorithm shows promising results in terms of low schedule costs and system efficiency.

The rest of the paper is organized as follows. In Section 2, we define the task scheduling problem. Section 3 discusses some basic attributes of DAG scheduling. Section 4 introduces SDBATS algorithm and Section 5 provides results and discussions. Section 6 concludes the paper with some final remarks.

2. Task scheduling problem.

A scheduling system model is composed of an application, a target computing environment, and performance criteria for scheduling. An application is represented by $G = (V, E)$, where V is the set of v tasks and E is the set of e edges between the tasks. Each edge $e_{(i,j)} \in E$ represents the precedence constraint such that task n_i should complete its execution before task n_j starts. Communication data required to be transmitted from task n_i to task n_j is represented by a (v x v) matrix. In a given task graph, a task without any predecessor is called an entry task and a task without a successor is called an exit task.

The target system consists of a set of resources $R = [r_j: j=0, m-1]$ of m independent resources fully connected by a high-speed arbitrary network. The data transmission rate (bandwidth) of the links between different resources in HCS may be different due to diverse network environment. A computation cost matrix is represented as W (n x m), in which each $w_{i,j}$ gives the Estimated Computation Time (ECT) to complete the task v_i on resource r_j where $0 <= i < n$ and $0 <= j <= m-1$. The ECT value of a task may be different on different resources depending on a processor's computation capability. The communication cost between two resources r_x and r_y, depends on the channel initialization at both sender resource r_x and receiver resource r_y in addition to the communication time on the channel. We assumed that there are no contentions for inter processor communications among the tasks. We further assume that any two connecting tasks scheduled on the same processor have zero communication costs. A task graph with 10 tasks and its corresponding computation costs with respect to each resource (i.e. $\{r_1, r_2, r_3\}$) which is given in [7] is shown in figure (1).

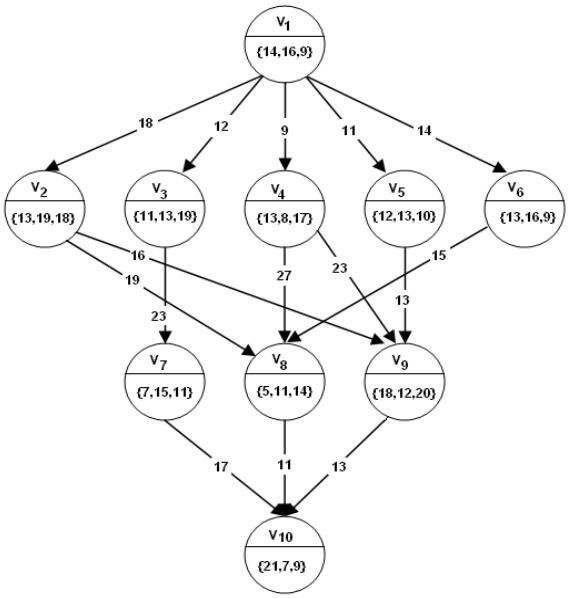

Figure 1. A simple task graph. Indices inside the nodes represent node numbers with computation cost on three processors while the weights on edges represent the communication cost.

3. DAG scheduling attributes.

Before proceeding to the next section, it is necessary to discuss some basic scheduling attributes

such as *rank upward*, *rank downward, earliest startup time* and *earliest finish time*, which will be used in the proposed scheduling algorithm.

The upward rank, $rank_u$, of the task ν_n is calculated using the following equation

$$rank_u(\nu_n) = \overline{w_n} + \max_{\nu_m \varepsilon\, succ(\nu_n)} \{\overline{d_{nm}} + rank_u(\nu_m)\} \quad (1)$$

where $\overline{w_n}$ is the average computation cost of the task ν_n, $\overline{d_{nm}}$ is the average communication cost of the edge from task ν_n to task ν_m, $succ(\nu_n)$ is the set of all immediate successors of task ν_n. For the exit task, the upward rank is

$$rank_u(\nu_n) = \overline{w_{exit}} \quad (2)$$

Similarly, the downward rank of task ν_n is defined by the following equation

$$rank_d(\nu_n) = \max_{\nu_m \varepsilon\, pred(\nu_n)} \{\overline{w_n} + \overline{d_{mn}} + rank_d(\nu_m)\} \quad (3)$$

where $pred(\nu_n)$ is the set of all immediate predecessors of the task ν_n. $EST(\nu_n, p_k)$ and $EFT(\nu_n, p_k)$ are the earliest startup time and earliest finish time of the task ν_n on the processor p_k, respectively. For the entry task ν_{entry},

$$EST(\nu_{entry}, p_k) = 0, \quad (4)$$

For the other tasks in the task graph, the EST and EFT values are computed recursively, starting from the entry task. EST and EFT are defined by

$$EST(\nu_n, p_k) = \max \{avail[k], \max_{\nu_m \varepsilon\, pred(\nu_n)} (AFT(\nu_m) + d_{m,n})\} \quad (5)$$

$$EFT(\nu_n, p_k) = w_{n,k} + EST(\nu_n, p_k) \quad (6)$$

where $avail[k]$ is the earliest time at which processor p_k is ready to execute the next task, $AFT(\nu_m)$ is actual finish time of task ν_m and $d_{m,n}$ is communication cost of the edge from task ν_n to task ν_m. In order to compute *EFT* of a task ν_n, all the immediate predecessor tasks of ν_n must have been scheduled. After a task n_m is scheduled on processor p_j, the earliest start time and the earliest finish time of task n_m on processor p_j is equal to the actual start time, $AST(n_m)$, and the actual finish time, $AFT(n_m)$, respectively. The schedule length (makespan) of the task graph is defined as follows

$$makespan = AFT(\nu_{exit}) \quad (7)$$

where ν_{exit} represents exit task.

4. Proposed algorithm.

The proposed algorithm consists of two phases, a task prioritizing phase, which arranges the tasks to schedule on the basis of some predefined criterion and a processor selection phase, which uses the same mechanism as adopted by HEFT insertion based policy.

Task Prioritization: The proposed algorithm considers, the computation cost heterogeneity of the available set of processors and communication cost heterogeneity of the links connecting any two processors, as a key attribute while calculating the priorities of the given DAG. The computational capabilities of different processors for a specific task may be different and it depends on the task size. HEFT takes the mean value of the expected computation and communication costs to calculate the rank of tasks. The proposed algorithm takes the standard deviation of the expected computation and communication costs as a significant attribute while calculating the upward rank of the task. The upward rank, $rank_u$, of each task has been calculated using the following equation

$$rank_u(\nu_n) = \sigma_n + \max_{\nu_m \varepsilon\, succ(\nu_n)} \{\sigma c_{ij} + rank_u(\nu_m)\} \quad (8)$$

where σ_n is the standard deviation of computation cost of any given task ν_i on the available pool of processors and σc_{ij} is the standard deviation of communication cost of any link between two connected processors. All the other parameters in equation are the same as described in the [7]. The upward rank for the DAG in figure 1 using equation 8 is given in the Table 1.The scheduling order of the tasks as calculated by SDBATS algorithm is $\{\nu_1, \nu_3, \nu_4, \nu_2, \nu_6, \nu_5, \nu_7, \nu_9, \nu_8, \nu_{10}\}$

Processor Selection: The processor selection phase of the proposed algorithm is a combination of task duplication and list scheduling techniques. Here the partial or selective duplication policy is employed and only the entry task is replicated on all of the given processors. The algorithm starts by arranging the tasks in a priority queue in decreasing order of their upward ranks. At any given instance, the task with highest priority is picked from the queue for scheduling. The schedule length of the DAG given in figure 1 so obtained for the SDBATS, HEFT and CPOP algorithms is 76, 80 and 86 respectively.

TABLE 1: Values of attributes using Standard Deviation as a priority metric for the task graph given in Figure 1.

Task	$rank_u(v_i)$	$rank_d(v_i)$	$rank_u(v_i) + rank_d(v_i)$
v_1	70.863	0.000	70.863
v_2	44.891	21.605	66.496
v_3	55.257	15.606	70.863
v_4	54.186	12.605	66.791
v_5	38.785	14.605	53.390
v_6	41.189	17.605	58.794
v_7	28.095	42.768	70.863
v_8	22.677	44.114	66.791
v_9	24.257	40.820	65.077
v_{10}	7.095	63.768	70.863

Algorithm 1: The SDBATS Algorithm

Input : DAG, set of tasks V, set of Processors P

Output : Schedule result, makespan

1. Set the computation costs of tasks and communication cost of links with standard deviation values.
2. Starting from the exit node, compute rank for all tasks by traversing given DAG upward using equation (8).
3. Sort the tasks in scheduling list by decreasing order of rank value.
4. **While** there are unscheduled tasks in the list **do**
5. Select the first task v_i from the list for scheduling
6. **for** each processor $p_k$in the processor set ($p_k \varepsilon$ P) **do**
7. Compute the earliest finish time by equation (6)
8. **end**
9. find the minimum earliest finish time processor p_k
10. **If** the task is the entry task
11. Assign the task to all the processor with minimum EFT (v_{entry} ,p_k)
12. **Else**
13. Assign the task v_i to processor p_k which minimize EFT (v_i ,p_k)
14. **End if**
15. **end while**

It is clear from the schedule length values so obtained that the use of standard deviation as a priority metric for rank calculations give considerable improvement in schedule length.

SDBATS algorithm has the same time complexity as the HEFT algorithm. For a given DAG with *e* edges and for *q* available set of processors, the SDBATS algorithm has a time complexity of $O(e \times q)$. In terms of number of tasks (v), the algorithm has a time complexity of the order of $O(v^2 \times q)$.

The pseudo code for the algorithm is shown above.

5. Experimental results and discussion.

This section presents a performance comparison of the proposed algorithm with four well known list scheduling algorithms such as HEFT, PETS, CPOP and DLS. For this purpose, standard task graphs from Chris Walshaw Graph Partitioning archive [26] were taken. Three other task graphs of real world application problem such as Gaussian Elimination, Fast Fourier Transform and Molecular Dynamic Code were also chosen for the simulation study. Computation and communication costs of the real world DAGs were assigned using the coefficient of variation based ETC generation method [4]. The scheduling algorithms were implemented in a simulation environment to generate output schedules. Finally the algorithms were analyzed using the following comparison metrics;

5.1 Comparison Metrics

The comparisons of the algorithms are based on the following metrics.

Makespan

The makespan is defined as the overall Completion time of the application tasks.

Schedule Length Ratio (SLR)

The main performance measure is the schedule length (makespan) of its output schedule. Since a large set of task graphs with different properties is used, it is necessary to normalize the schedule length to the lower bound, which is called the Schedule Length Ratio (SLR). The SLR value of an is defined as

$$SLR = \frac{makespan}{\sum CP_{min}} \qquad (9)$$

Speedup

The speedup value is computed by dividing the sequential execution time (i.e., the cumulative computation costs of the tasks) by the parallel execution time (i.e., the makespan).

5.2 Performance results on Standard Task Graphs

Standard task graphs at Chris Walshaw Graph Partitioning archive are usually undirected task graphs with vertices *V* and edges *E*. These graphs were made directed acyclic graphs by eliminating cyclic edges. Vertex and edge weights were assigned using Gamma Distribution with four input parameters V_{task}, V_{mach}, V_{link} and μ_{task} which determine the shape and scale parameters of the Gamma Distribution. Following are the parameters which were used to build weighted DAGs:

- Task heterogeneity is determined by $V_{task.}$ Five values so chosen are {0.1, 0.2, 0.3, 0.4, 0.5} where 0.1 represents low task heterogeneity and 0.5 represents high task heterogeneity.
- Machine heterogeneity represents the degree of processor execution times for a particular task and is determined by V_{mach}. Five values for V_{mach} are {0.1, 0.2, 0.3, 0.4, 0.5} where 0.1 represents low machine heterogeneity and 0.5 represents high machine heterogeneity.
- Link heterogeneity of a particular link between any two processors is set by taking five values of V_{link}. i.e. {0.1, 0.2, 0.3, 0.4, 0.5} where 0.1 represents low edge heterogeneity and 0.5 represents high edge heterogeneity.

For each value of V_{task}, V_{mach} and V_{link}, 100 different DAGs were generated. These diverse set of DAGs with different characteristics avoid bias towards a specific scheduling algorithm.

The first parameter taken for the comparison of different algorithms with the proposed algorithm is task heterogeneity. Average SLRs and average speedups produced by the SDBATS, PETS, HEFT, CPOP and DLS for various V_{task} values are shown in figure 3(a) & 3(b).

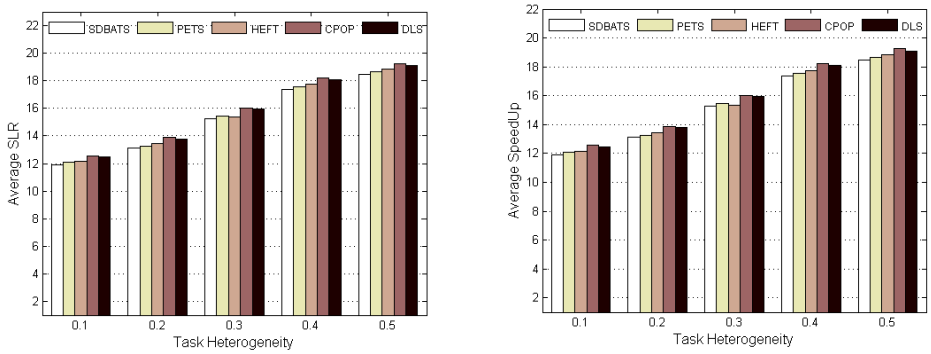

Figure 3. Standard Task Graphs (a) Average SLR and (b) Average SpeedUp for various task heterogeneity values

The average SLR value produced by SDBATS is shorter than PETS, HEFT, CPOP and DLS. The improvement achieved by the SDBATS over the well-known list scheduling algorithms for various V_{task} values is a clear manifestation that the proposed algorithm is good enough for solving problems of various complexities. Similarly the average Speedup values for the SDBATS outperform the existing algorithms for the above mentioned task heterogeneity values.

Machine heterogeneity is another parameter to judge the performance of a scheduling algorithm. Computation cost heterogeneity is represented by V_{mach}. Average SLR and average speedup values produced by the SDBATS, PETS, HEFT, CPOP and DLS for the V_{mach} values such as {0.1, 0.2, 0.3, 0.4, 0.5} are shown in figure 4(a) and 4(b). Figure 4(a) shows that the average SLR value produced by SDBATS is shorter than PETS, HEFT, CPOP and DLS for different computation cost heterogeneity values as mentioned in section 5.2. Similarly figure 4(b) shows that the average speedup value achieved by SDBATS algorithm is higher than PETS, HEFT, CPOP and DLS algorithms. All these results indicate that the SDBATS algorithm out performs PETS, HEFT, CPOP and DLS algorithms in terms of average SLR and average speedup values.

This is only possible because the SDBATS algorithm considers the computational heterogeneity of the expected execution time of each task on different processors and assigns that task first which has highest standard deviation to the fastest available processor.

Link heterogeneity is also an important parameter which is greatly affects the performance of a scheduling algorithm in a heterogeneous computing environment. This parameter is represented by link heterogeneity. The SDBATS algorithm has the efficiency to measure the communication heterogeneity through statistical function, such as standard deviation, and assign the job to the most appropriate processor. Figure 5(a) and 5(b) shows the average SLR and average speedup values for the SDBATS, PETS, HEFT, CPOP and DLS for various values of V_{link}. The average SLR value produced by SDBATS is shorter than PETS, HEFT, CPOP and DLS. Similarly the average speedup gain achieved by SDBATS over the other four algorithms is shows the supremacy of the proposed algorithm for distributed heterogeneous computing environments.

It is clear from figures 5(a) and 5(b) that the SDBATS algorithm outperforms PETS, HEFT, CPOP and DLS algorithms in terms of average SLR and average speed up for various link heterogeneities.

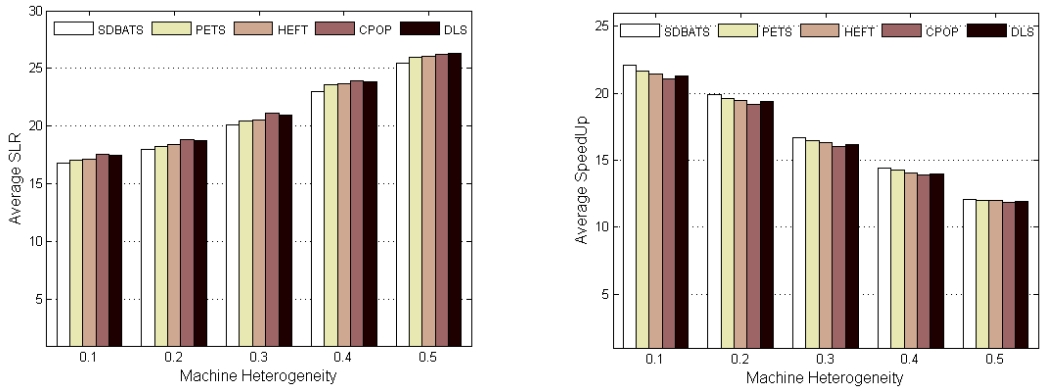

Figure 4. Standard Task Graph (a) Average SLR and (b) Average SpeedUp for various machine heterogeneity values

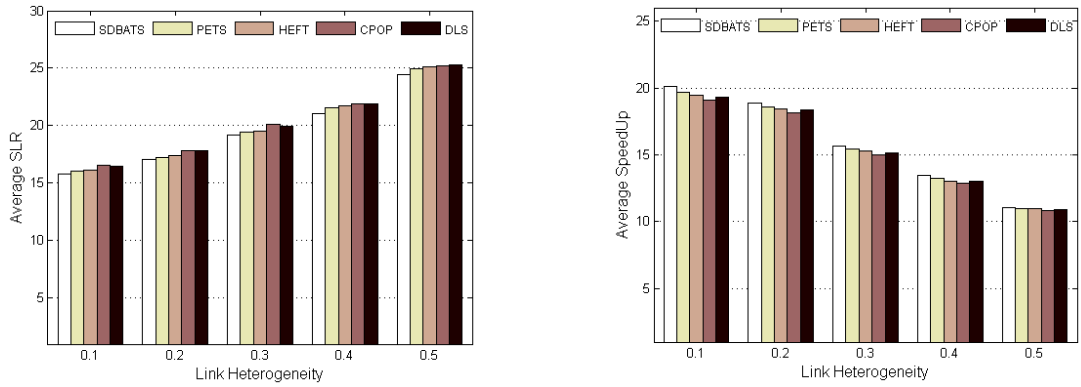

Figure 5. Standard Task Graphs (a) Average SLR and (b) Average SpeedUp for various link heterogeneity values

5.3 Performance results on Gaussian Elimination Task graphs

In this section the performance of the SDBATS algorithm is studied on the parallel application graphs of Gaussian elimination algorithm [e.g. 7, 16]. Since the structure of the graph is known in prior, so there is no need for the parameters such as number of nodes and parallelism parameter (α). The computation and communication costs were generated by the above mentioned coefficient of variation based ETC generation method using the Gamma distribution. The computation and communication costs heterogeneity is represented by various V_{mach} and V_{link} values as discussed in section 5.2. A new parameter called matrix size (M) of the Gaussian graph is introduced to represent the total number of nodes in the graph. The total number of nodes in the Gaussian graph is calculated using the equation [5]

$$\text{Number of Nodes} = \frac{M^2+M-2}{2} \qquad (10)$$

We evaluated the performance of the SDBATS by taking different matrix sizes from 6 to 20 with an increment of 2. The smallest graph size in this experiment has 14 nodes and the largest one has 209 nodes. The number of processors is set to 5 for all the experiments. The average SLR produced by SDBATS, PETS, HEFT, CPOP and DLS algorithms is shown in figure 7(a). The average SLR value produced by SDBATS is shorter than PETS, HEFT, CPOP and DLS for machine heterogeneity values of: 0.1, 0.2, 0.3, 0.4 and 0.5 respectively. Similarly the average speedup values of the scheduling algorithms with respect to above mentioned values are shown in figure 7(b). The SDBATS algorithm has a higher speedup value than the PETS, HEFT, CPOP and DLS algorithms for different input matrix size and different machine heterogeneity values.

The average speedup gain of SDBATS over the other four algorithms is a clear indication that for real world Gaussian application graphs, SDBATS outperforms the HEFT, PETS, CPOP and DLS algorithms in terms of SLR and speedup.

The average SLR and average speedup gain of SDBATS over the other four algorithms is a clear indication that for real world Gaussian application graphs, SDBATS outperforms the HEFT, PETS, CPOP and DLS algorithms in terms of SLR and speedup.

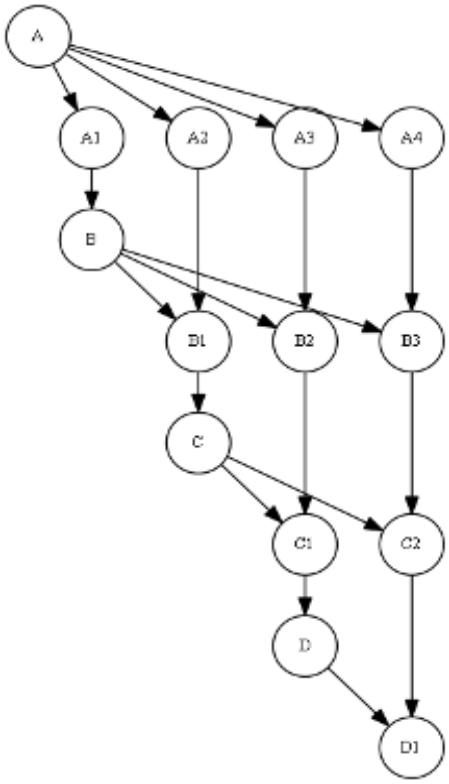

Figure 6.Gaussian Elimination task graph with matrix size of 5

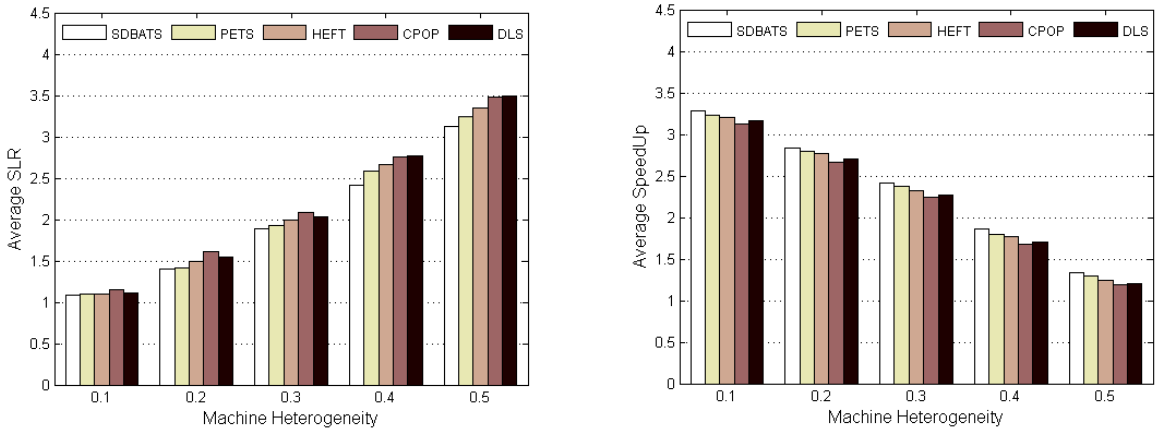

Figure 7. Gaussian Task Graphs (a) Average SLR and (b) Average SpeedUp for various machine Heterogeneity values

5.4 Performance results on Fast Fourier Transformation graphs

Fast Fourier transformation is an algorithm optimization of Discrete Fourier Transform Cooley-Tukey FFT algorithm is used for the task decomposition and generation of FFT task graph [e.g. 7, 16]. The required task graph is a combination of recursive call tasks and butterfly operation tasks. For a tasks graph of N nodes, there are 2xN-1 nodes representing the recursive call tasks and N x $\log_2$ (N) nodes are allocated for butterfly operation tasks.

The structure of FFT task graph is represented by parameters such as number of data points, computation and communication cost heterogeneities. For the evaluation of our algorithm, the values for V_{mach} {0.1, 0.2, 0.3, 0.4, 0.5} were chosen as described in section 5.2. The FFT task graphs are then generated with different number of data points starting from 2 to 32 with an incrementing power of 2. The number of processors is fixed to 5 and average values of SLR are calculated for different V_{mach} values. Similarly the average speedup is calculated by varying the same above mentioned V_{mach} values. Figure 9(a) and 9(b) shows the average SLR and average speedup values for different V_{mach} values.

The average SLR value produced by SDBATS is shorter than PETS, HEFT, CPOP and DLS for the V_{mach} values {0.1, 0.2, 0.3, 0.4 0.5} respectively. It is due to the fact that the standard deviation is the true parameter to cater for the computation cost heterogeneity of the FFT DAG which increases with the increase in the machine heterogeneity.

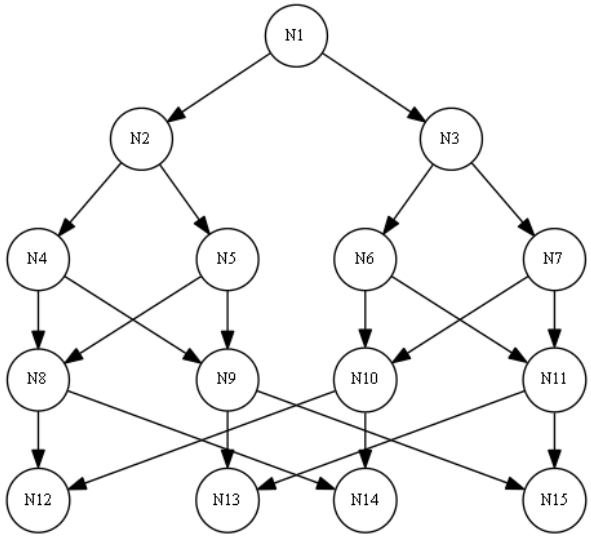

Figure 8.FFT task graph with four Data points

Similarly the average speedup gain achieved by SDBATS over the other four algorithms exhibits superiority of SDBATS over the other list scheduling algorithms.

It is clear that for real world Fast Fourier Transform application graphs, SDBATS outperforms the PETS, HEFT, CPOP and DLS algorithms in terms of SLR and speedup.

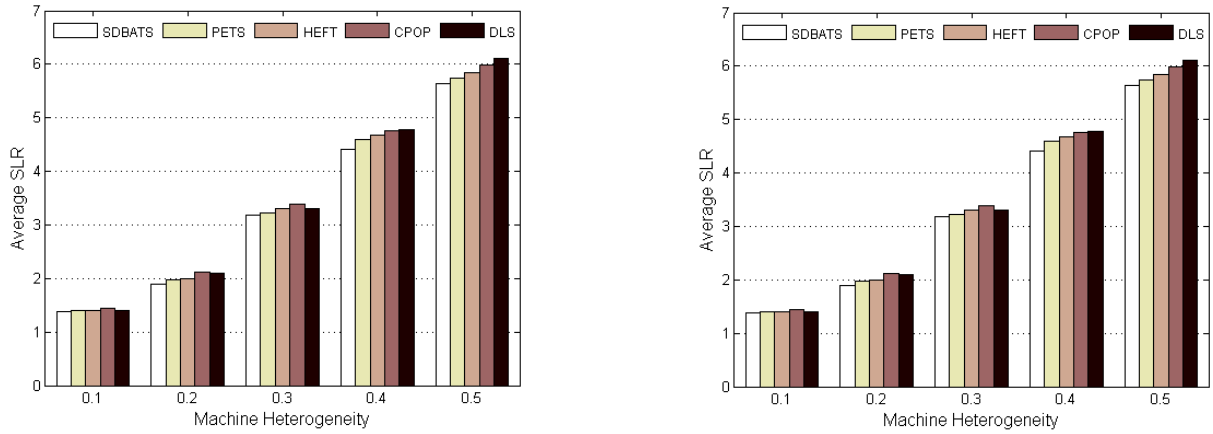

Figure 9. FFT Task Graphs (a) Average SLR and (b) Average SpeedUp for various Machine heterogeneity values.

5.5 Performance results on Modified Molecular Dynamics Code Task graph

Modified Molecular Dynamics Code is another interesting application with an irregular shape task graph [e.g. 7, 16]. This task graph has a static structure owing to its fixed value of nodes i.e 41, as shown in the figure 10. The varying parameters that are taken for the evaluation of the proposed algorithm are computation and communication cost heterogeneities. For the evaluation of our algorithm, the values for V_{mach} {0.1, 0.2, 0.3, 0.4, 0.5} were chosen as described in section 5.2. In the current scenario the average SLR of the SDBATS, HEFT, PETS, DLS and CPOP algorithms was evaluated by fixing the number of processors to seven and varying the V_{mach} values. Similarly the average speedup is calculated by varying the same above mentioned V_{mach} values.

Figure 11(a) shows the performance results for average SLR for different V_{mach} values. As in the case of previous experiments the average SLR value produced by SDBATS is shorter than PETS, HEFT, CPOP and DLS for the V_{mach} values {0.1, 0.2, 0.3, 0.4, 0.5} respectively.

Similarly the average speedup gain achieved by SDBATS over the other four algorithms is shown in figure 11(b) for the above mentioned V_{mach} values.

It is clear that for real world Molecular Dynamics Code application graphs, SDBATS outperforms the PETS, HEFT, CPOP and DLS algorithms in terms of average SLR and average speedup.

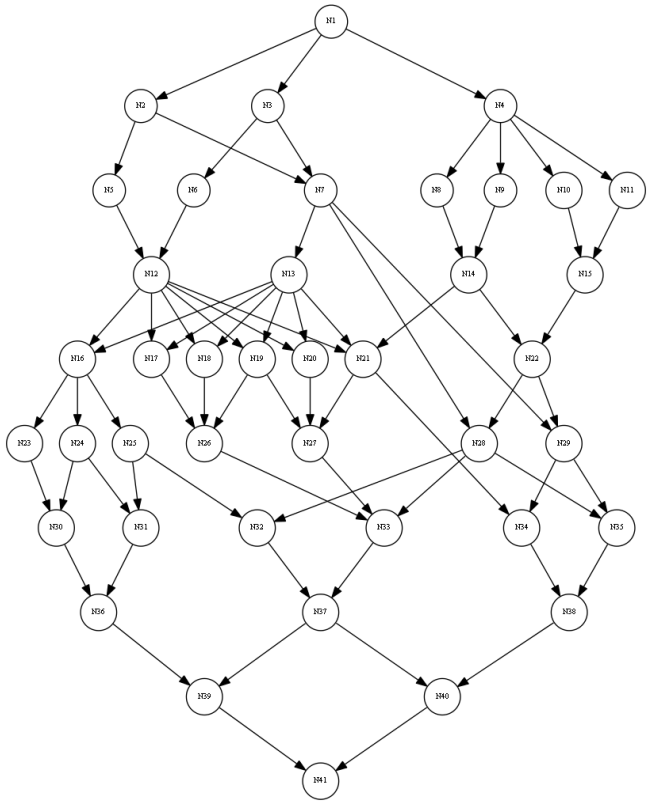

Figure 10. Molecular Dynamics Code task graph

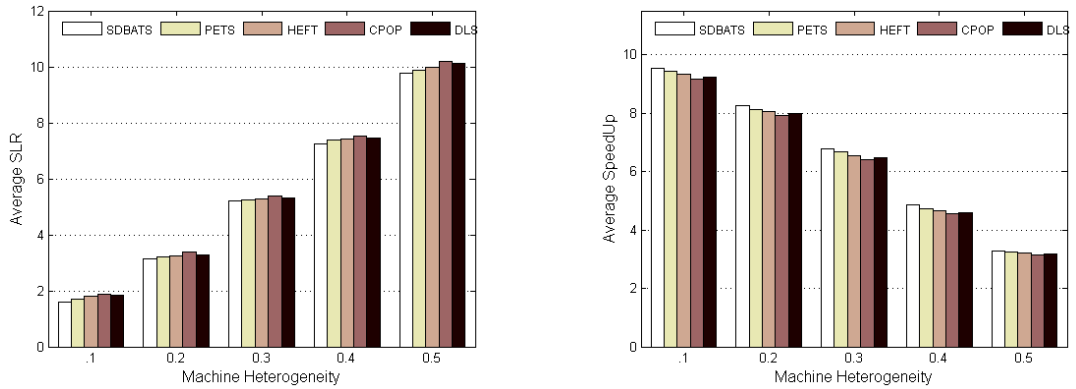

Figure 11. Molecular Dynamics Task Graphs (a) Average SLR and (b) Average SpeedUp for various machine Heterogeneity values

6 Conclusion.

We present a SD-Based Algorithm for Task Scheduling (SDBATS) for scheduling precedence constrained task graphs in heterogeneous computing systems. The proposed algorithm takes into account the heterogeneity in the execution time of the given tasks on the diverse set of resources. This intuitive concept accommodated the heterogeneous nature of the distributed computing systems. The standard deviation attribute of the SDBATS algorithm has proved to be a critical parameter for achieving high quality schedules with minimum makespan.

The performance of the SDBATS algorithm has been compared with four well known DAG scheduling algorithms, PETS, HEFT, CPOP and DLS. The comparative study is based on the random generated directed acyclic graphs as well as the DAGs real world application graphs. The results exhibit performance improvement of SDBATS over the PETS, HEFT, CPOP and DLS algorithms in terms of schedule length and speedup.

7 References.

[1] M. R. Gary and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman and Company, 1979.

[2] O. H. Ibarra, C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors", Journal of ACM, pp. 280–289, 1977.

[3] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, ``Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," Journal of Parallel and Distributed Computing, Vol. 59, No. 2, pp: 107-131, Nov. 1999.

[4] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, "Task execution time modeling for heterogeneous computing systems", 9th IEEE Heterogeneous Computing Workshop (HCW 2000), pp: 185-199, May 2000.

[5] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", 8th IEEE Heterogeneous Computing Workshop (HCW '99), pp: 30-44, Apr. 1999.

[6] Y. K. Kwok, I. Ahmad, "Benchmarking and comparison of the task graph scheduling algorithms", Journal of Parallel and Distributed Computing, pp: 381-422, 1999.

[7] H. Topcuoglu, S. Hariri and M. Y. Wu, "Performance Effective and Low Complexity Task Scheduling Algorithm scheduling for heterogeneous computing", IEEE Transaction on Parallel and Distributed System, Vol.13, No.3, 2002.

[8] H. Zhao, R. Sakellariou, "An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm", in Proceedings of 9th International Euro-Par Conference, Springer-Verlag, pp. 189-194, 2003.

[9] M. Y. Wu and D. D. Gajski, "Hypertool: A Programming Aid for Message-Passing Systems," IEEE Trans. Parallel and Distributed Systems, vol. 1, no. 3, pp. 330-343, July 1990.

[10] J. J. Hwang, Y. C. Chow, F. D. Anger, and C. Y. Lee, "Scheduling Precedence Graphs in Systems with Inter processor Communication Times", SIAM J. Computing, vol. 18, no. 2, pp. 244-257, April 1989.

[11] 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. 75-87, Feb. 1993.

[12] H. Zhao, R. Sakellariou, "A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems", Proc. 18th Parallel and Distributed Processing Symposium, pp: 26-30, April 2004.

[13] I. Daoud, N. Kharma, "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems", J. Parallel Distrib. Comput., pp. 399-409, 2008.

[14] G. Q. Liu, K. L. Poh, M. Xie, "Iterative list scheduling for heterogeneous computing", J. Parallel Distrib.Comput.pp. 654-665, 2005.

[15] X. Tang, L. Kenli, D. PADUA, "Communication contention in APN list scheduling algorithm", Science in China Series F: Information Sciences, pp. 59-69, 2009.

[16] X. Tang, L. Kenli, L. Guiping, L. Renfa, "List scheduling with duplication for heterogeneous computing systems", Journal of Parallel and Distributed Computing, pp. 323-329, 2010.

[17] E. Ilavarasan and P. Thambidurai, "Low complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments", Journal of Computer Sciences 3, pp. 94-103, 2007.

[18] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, S Ali, "Represent Task and machine Heterogeneities for Heterogeneous Computing Systems" Tamkang Journal of Science and Engineering , Special 50th Anniversary Issue , Vol. 3 , pp: 195-207, 2000.

[19] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, S Ali, "Task execution time modeling for

Heterogeneous Computing Systems" Tamkang Journal of Science and Engineering , Special 50th Anniversary Issue , Vol. 3 , pp: 195-207, 2000.
[20] E. U. Munir, J. Li, S. Shi, Z. Zou, Q. Rasool, "A new heuristic for task scheduling in heterogeneous computing environment", Journal of Zhejiang University-Science, pp:1715-1723, 2008.
[21] A. Al-Qawasmeh, A. Maciejewski, H. J. Siegel, "Characterizing heterogeneous computing environments using singular value decomposition". In the proceedings of 19th heterogeneity in computing workshop, April 2010.
[22] A. Al-Qawasmeh, A. Maciejewski, H. Wang, H. J. Smith, H. J. Siegel and J. Potter "Statistical measures for quantifying task and machine heterogeneities". Journal of Supercomputing, special Issue on Advances in Parallel and Distributed Computing. Vol. 57, No. 1, pp: 34-50, 2011.
[23] A. V. Chandak, B. Sahoo and A. K. Turuk. "Heuristic Task Allocation Strategies for Computational Grid". International Journal of Advanced Networking and Applications Vol 02, Issue 05, pp: 804-810, 2011.
[24] S. Zhang, S. Liu, C. Xie, L. Wang, Y. Gao and X. Han. "A High Dependability Grid Resource Scheduling Algorithm" Journal of Computational Information Systems, pp: 607-614, 2011.
[25] Y. K. Kwok, I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors". ACM Computing Surveys, pp: 406–471, 1999.
[26] Chris Walshaw Graph Partitioning Archive, http://staffweb.cms.gre.ac.uk/~wc06/partition/