2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

List-Based Task Scheduling for Cloud Computing

Muhammad Fasih Akbar[†], Ehsan Ullah Munir[‡], M. Mustafa Rafique[⋆], Zaki Malik[Ψ],
Samee U. Khan[§], Laurence T. Yang[ξ]
[†]Department of Computer Science, Allama Iqbal Open University, Islamabad - Pakistan
[‡]Department of Computer Science, COMSATS Institute of Information Technology, Wah Cantt - Pakistan
[⋆]IBM Research - Ireland
[Ψ]Department of Computer Science, Wayne State University - USA
[§]Department of Electrical and Computer Engineering, North Dakota State University - USA
[ξ]Department of Computer Science, St. Francis Xavier University - Canada
muhammad.fasihakbar@powerobjects.com; ehsanmunir@comsats.edu.pk; mustafa.rafique@ie.ibm.com;
zaki@wayne.edu; samee.khan@ndsu.edu; ltyang@stfx.ca

***Abstract*—Cloud computing model provides global and on-demand access to resources in a seamless manner with minimal interaction with the service provider. A typical cloud data center consists of several computational resources interconnected with each other through high-speed networks. In cloud the program execution can be visualized as a collection of multiple tasks represented by Directed Acyclic Graph (DAG) that execute in their logical sequence. Prioritization of these tasks plays an important role to achieve high performance and improved efficiency in a cloud environment. In this paper, we propose a novel task scheduling algorithm named Median Deviation based Task Scheduling (MDTS), which uses Median Absolute Deviation (MAD) of the Expected Time to Compute (ETC) of a task as a major attribute to calculate ranks of the given tasks. We use *coefficient-of-variation* (COV) based technique that considers task and machine heterogeneity to estimate the ETC of a particular DAG. The proposed algorithm is evaluated under various conditions using synthetic DAGs and real world applications. Our evaluation shows that the proposed MDTS algorithm produces high quality schedules and significantly reduces the makespan of an application by up to 25.01%.**

***Keywords*—Cloud computing; heterogeneous computing; DAG scheduling; makespan; task scheduling**

I. Introduction

The cloud computing paradigm follows a pay per use model where users use services from the cloud without knowing the hosting details and delivery mechanism [1], [2]. This provides suitable, global, and on-demand access to a shared pool of resources (i.e., servers, storage, networks, web services, etc.) for reduced time to market for enterprises and reduced time to discovers for scientific discoveries. These resources can be provided to users in a consistent way with minimum effort and interaction of the service provider [3], [4]. The aim of a cloud environment is to provide easy-to-use platform for dynamic and flexible applications. Such platform becomes available in the form of different hardware and software services. These services enable users to deploy their applications over the Internet by specifying their availability, performance, and Quality of Service (QoS) requirements [5]. Due to diverse configuration, composition, and deployment requirements of these applications, the engaged task scheduling and resource management strategies [6] become critical in improving the overall efficiency and the performance of cloud infrastructure.

In a cloud system, any program can be envisioned as the execution of different tasks contained in it. These tasks are typically of two types, i.e., independent and dependent tasks. Independent tasks can be executed concurrently by multiple Virtual Machines (VMs), whereas dependent tasks must be scheduled by satisfying their precedence constraints, which can be expressed as a Directed Acyclic Graph (DAG) where nodes represent tasks and edges represent inter-task dependencies [7]. It is extremely desirable to execute the constrained tasks in a scheduling order that minimizes the overall schedule length. However, finding the optimal solution for a task scheduling problem is known to be NP-Complete [7].

Task scheduling can be divided into two main categories, namely compile-time (static) and run-time (dynamic) scheduling [8]. Static scheduling is further divided into two categories, i.e., Guided Random Search Based (GRSB) and heuristics based [8], [9]. The GRSB algorithms are more expensive than heuristics based ones because GRSB algorithms need more iterations for producing a better schedule. Conversely, the heuristics based algorithms provision approximate solutions in polynomial time. The heuristics based technique has subcategories, i.e., duplication based [10], [11], cluster based [12], and list based [8]. The duplication based heuristics have higher time complexity whereas cluster based heuristics are more suitable for homogeneous systems. We focus on list based algorithms [7], [8], [13]–[17] in this paper, due to their lesser time complexity and higher efficiency to produce shorter makespan. The list based algorithms have two phases to schedule list of tasks. In the first phase, priority or rank is assigned for each task before sorting them in a descending order. In the second phase, the task with highest priority is scheduled on the available processor.

In this paper, we present a novel list-based task scheduling algorithm, named Median Deviation based Task Scheduling (MDTS), for restricted number of fully connected VMs in cloud environment. The proposed algorithm outperforms the state of art algorithms such as SDBATS [7], HEFT [8], CPOP [8], and PEFT [13], in terms of makespan, speedup, and efficiency. The time complexity of the proposed MDTS is $O(t^2 \times m)$ for t number of tasks and m number of VMs, which is the same as that of HEFT, PEFT, and SDBATS. We leverage *coefficient-of-variation* (COV) [18] to consider task heterogeneity and machine heterogeneity factors in the cloud environment while designing the proposed algorithm. COV based approach asserts the standard deviation as a percentage of the average values. In order to improve the quality and the reliability of the produced schedules, the proposed MDTS algorithm use Median Absolute Deviation (MAD) for calculating the ranks of the given tasks.

978-1-5090-5880-8/16 $31.00 © 2016 IEEE
DOI 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.143

Specifically, this paper makes the following contributions:

- We define the task scheduling problem and the associated attributes in the cloud systems for efficiently scheduling the given tasks on the available VMs by incorporating the inter-task dependencies, task heterogeneity, and the available cloud resources heterogeneity in making the scheduling decisions.
- We design and develop a list-based task scheduling algorithm, called Median Deviation based Task Scheduling (MDTS), that considers ranks of each task and selects the appropriate processor for each task such that the overall execution time of the scheduled tasks is minimized.
- We thoroughly evaluate and compare the proposed MDTS algorithm with four well-known list-based task scheduling algorithms, by using random DAGs and real world applications and studying more than 4,000 different graphs, to show that MDTS outperforms the other algorithms and can reduce the overall execution time of the executing tasks by up to 25%.

II. Related Work

In literature, a number of list scheduling algorithms have been proposed. Heterogeneous Earliest Finish Time (HEFT) [8] recursively calculates upward ranks of tasks with average computational and communication costs. Standard Deviation Based Task Scheduling (SDBATS) [7] uses standard deviation of computational and communication costs to calculate upward ranks. Critical Path On a Processor (CPOP) [8] sums up upward and downward ranks to create a priority queue and critical path. At each level of a DAG, Performance Effective Task Scheduling (PETS) [14] sums up average computational, data transfer, and data receiving costs to determine ranks of tasks. Heterogeneous Earliest Finish time with Duplication (HEFD) [11] uses task variance as heterogeneity factor to calculate computational and communication costs between tasks. PEFT [13] is based on look ahead approach, and forecasts the child tasks by computing an Optimistic Cost Table (OCT). The OCT is a matrix in which the number of tasks and processors are represented by the rows and columns, respectively, and each element $OCT(t_i, p_k)$ indicates the maximum of the shortest paths of t_i children tasks to the exit node considering that processor p_k is selected for task t_i. All these algorithms are based on standard deviation or average of task weights on available processors and do not incorporate the heterogeneity of the system.

A recent work [7] considers standard deviation to incorporate task and machine heterogeneity on available processors. We use *coefficient-of-variation* (COV) [18] based approach to consider these heterogeneity factors. The COV of a set of values is a better measure for the diffusion in the values than the standard deviation because it asserts the standard deviation as a percentage of the mean of values [18]. Let σ be the standard deviation, and μ be the mean, of a set of execution times, then $COV = \sigma/\mu$. We use this approach to compute Expected Time to Compute (ETC) of tasks and communication costs between them. The COV-based ETC generation method provides higher control over values in a given column or row of the ETC matrix than range-based method [18]. Moreover, we use Median Absolute Deviation (MAD) for calculating ranks of tasks. MAD is a robust statistical measure of dispersion as compared to standard deviation and is more pliable to outliers in the given data set. Outliers can also affect the interquartile range but the advantage of MAD can be used to approximate the standard deviation [19], [20]. Moreover, rank calculations is simpler with MAD as compare to standard deviation and takes less execution time. Our extensive evaluation shows that the proposed algorithm produces high quality schedules in terms of Schedule Length Ratio (SLR), speedup, efficiency, and reduced schedule length.

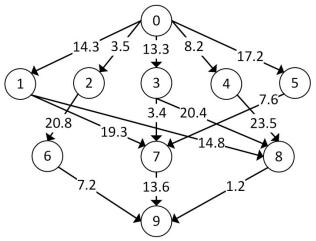


Fig. 1: A task graph with communication costs on the edges.

Task	vm_0	vm_1	vm_2
0	14.4	15.5	9.3
1	9.0	13.2	9.3
2	20.2	23.2	4.1
3	4.0	27.6	17.6
4	3.8	18.8	3.4
5	13.5	5.4	11.0
6	14.3	13.5	10.5
7	3.0	4.4	9.4
8	5.1	8.2	10.1
9	19.0	8.5	2.3

TABLE I: Computational cost matrix for task graph in Fig. 1.

III. Task Scheduling Problem

An application, a target computational system, and scheduling criteria build up a foundation for the model of scheduling system. An application can be represented as a DAG, $G(T, E)$, where $T = \{t_i, i = 0, \ldots, n-1\}$ is a set of n tasks and E represents set of edges between tasks $E = \{e_{i,j}, i < j\}$, and $e_{i,j}$ shows the precedence constraints between two tasks. Tasks $t_i, t_j \in T$, which are connected to each other by $e_{i,j}$ that represents the precedence constraint of task t_j being dependent on task t_i for its execution. It also shows that the output of the task t_i will be used as input for task t_j, and t_j cannot be scheduled before t_i. The task t_j is the successor of t_i and t_i is the predecessor of t_j. A matrix $CmC_{(t\times t)}$ shows the communication cost between any two tasks t_i and t_j. In the given DAG, a task without any predecessors is called an entry task and a task without any successors is called an exit task.

A cloud computing system comprises of a set $VM = \{vm_i : i = 0, m-1\}$ of m independent VMs which are fully connected through high speed network. The data transfer frequency (DTF) may be different due to a diverse network bandwidth of cloud environment. We can represent DTF as $m \times m$ matrix, and between any two VMs as $DTF_{m\times m}$. The Expected Computational Cost (ECC) can be shown by another matrix $ECC_{n\times m}$ to execute a task t_i on a VM vm_j, where $0 <= i <= n-1$ and $0 <= j <= m-1$. The ECC depends on the computational capacity of a VM and can be different for each VM. The communication cost between vm_x and vm_y depends on two factors. First is the initialized channel at the processors on both ends of communication, and second is the communication cost of the channel. We assume that every processor of a VM can communicate to other processor of a different VM without any contention. Similarly we assume that tasks scheduled on the same VM have zero communication cost between them. An example DAG with 10 tasks, and their computational and communication costs are shown in Fig. 1 and Table I, respectively. These costs are calculated by COV based-approach [18] described earlier.

t_i	$Rank_u(t_i)$	$Rank_d(t_i)$	$Rank_u(t_i)$ + $Rank_d(t_i)$
0	58.4	0	58.4
1	42.0	16.4	58.4
2	41.3	5.6	46.9
3	36.8	15.4	52.2
4	37.1	10.3	47.4
5	31.6	19.3	50.9
6	14.1	32.8	46.9
7	21.3	37.1	58.4
8	8.5	43.7	52.2
9	5.6	52.8	58.4

TABLE II: Upward ranks calculated by MDTS algorithm for DAG in Fig. 1.

IV. Task Scheduling Attributes

In this section we discuss task scheduling attributes, i.e., upward rank, downward rank, earliest startup time, and earliest finish time, that we use in the proposed algorithm.

The upward rank can be defined by the following equation:

$$Rank_u(t_n) = \overline{CompCost_n} + max_{t_m \in succ(t_n)} \left\{ \overline{CommCost_{n,m}} + Rank_u(t_m) \right\}, \quad (1)$$

where $\overline{CompCost_n}$ is the mean computational cost of the task t_n, $\overline{CommCost_{n,m}}$ is the mean communication cost of the link from task t_n to t_m, and $succ(t_n)$ is the set of immediate successors of task t_n. We define the upward rank of an exit task as follows:

$$Rank_u(t_{exit}) = \overline{CompCost_{exit}} \quad (2)$$

The downward rank for a task t_n can be defined as:

$$Rank_d(t_n) = max_{t_m \in pred(t_n)} \{ \overline{CompCost_n} + \overline{CommCost_{m,n}} + Rank_d(t_m) \} \quad (3)$$

where $pred(t_n)$ represents the set of immediate predecessors of task t_n. We define the earliest startup time $EST(t_n, vm_r)$ and the earliest finish time $EFT(t_n, vm_r)$ of task t_n on virtual machine vm_r, respectively. For the entry task, EST will be:

$$EST(t_n, vm_k) = 0 \quad (4)$$

Starting from the entry task, the EST and EFT for other tasks in the task graph can be calculated as:

$$EST(t_n, vm_r) = max\{avail[r], max_{t_m \in pred(t_n)} (AFT(t_m) + CommCost_{m,n})\} \quad (5)$$

$$EFT(t_n, vm_r) = CompCost_{n,r} + EST(t_n, vm_r) \quad (6)$$

In Eq. (5), $avail[r]$ is the available time of VM vm_r to execute the next task. A VM becomes available and ready to execute the next task when it has finished the execution of already assigned tasks. $EFT(t_m)$ is the earliest finish time for the task t_m, and $CommCost_{m,n}$ is the communication cost between tasks t_m and t_n. When a task t_m is scheduled on a VM vm_k, its EST and EFT are called Actual Start Time $AST(t_m)$ and Actual Finish Time $AFT(t_m)$, respectively. When an algorithm schedules all tasks of a DAG, the AFT of the exit task is the makespan of the schedule. If t_{exit} represents the exit task, then makespan can be represented as:

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

Algorithm 1: The MDTS Algorithm.

Input: DAG, Tasks Set T, Virtual Machines Set VM

1 Calculate the computational costs of tasks and communication costs of links with COV based approach as given in [18];
2 Calculate ranks of all tasks by traversing DAG upward from exit task to entry task by using Eq. (8);
3 Sort the tasks in descending order by rank values.;
4 **while** *unscheduled tasks exists in set* T **do**
5 Get the first task t_i with highest priority from the list T;
6 **forall** *virtual machine* vm_j *in the virtual machine set* VM **do**
7 Calculate EFT by Eq. (6);
8 **end forall**
9 Get the minimum EFT virtual machine vm_j;
10 Assign the task t_i to virtual machine vm_j that minimize $EFT(t_i, vm_k)$;
11 **end while**

Ranked Tasks	Virtual Machines						Selected Virtual Machines
	vm_0		vm_1		vm_2		
	EST	EFT	EST	EFT	EST	EFT	
t_0	0	14.4	0	15.5	0	9.3	2
t_1	23.6	32.6	23.6	36.8	9.3	18.6	2
t_2	12.8	33.0	12.8	36.0	18.6	22.7	2
t_4	17.5	21.3	17.5	36.3	22.7	26.1	0
t_3	22.6	26.6	22.6	50.2	22.7	40.3	0
t_5	26.6	40.1	26.5	31.9	22.7	33.7	1
t_7	39.5	42.5	37.9	42.3	39.5	48.9	1
t_6	43.5	57.8	43.5	57.0	22.7	33.2	2
t_8	33.4	38.5	47.0	55.2	47.0	57.1	0
t_9	55.9	74.9	42.3	50.8	55.9	58.2	1

TABLE III: The calculated EST, EFT values for virtual machines vm_0, vm_1, and vm_2 for tasks in Fig. 1.

V. Median Deviation Based Task Scheduling (MDTS) Algorithm

The proposed algorithm, shown in Algorithm 1, consists of two parts. The first part ranks the tasks and the second part selects the processor using the insertion based policy.

A. Ranking Tasks

In MDTS algorithm, we consider the task and machine heterogeneity by using COV based approach [18] for calculating the computational and communication costs. We include communication cost heterogeneity between two VMs during ranks computation for each task of the DAG. In the proposed algorithm, we use median deviation for expected computational and communication costs to calculate the upward ranks of a given set of tasks. The formula for calculating the upward rank $Rank_u$ is given as:

$$Rank_u(t_n) = MAD(CompCost_n) + max_{t_m \in succ(t_n)} \{MAD(CommCost_{n,m}) + Rank_u(t_m)\}, \quad (8)$$

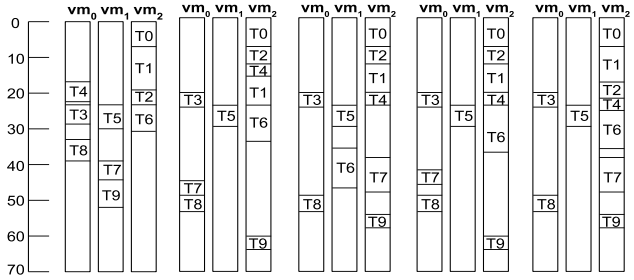
where $MAD(CompCost_n)$ is MAD of the computational cost of task t_n on the available set of VMs and $MAD(CommCost_{n,m})$ is MAD of communication cost of the link between tasks t_n and t_m. Table II shows the calculated upward ranks for the DAG shown in Fig. 1 using Eq. (8).

B. Processor Selection

In the processor selection part of MDTS algorithm, we schedule the task at the available VM that completes the task

	MDTS		SDBATS		HEFT		CPOP		PEFT	
	NBS	NSEB	NBS	NSEB	NBS	NSEB	NBS	NSEB	NBS	NSEB
Random DAGs	1092	1572	168	1276	408	1251	6	3	308	564
Gaussian Elimination	858	1012	391	1179	417	1339	124	0	398	1246
Montage Graphs	1258	1158	433	1072	519	1153	483	3	476	766
LIGO Inspiral Analysis	1104	678	425	608	563	716	571	0	495	484

TABLE IV: The number of times an algorithm gives best makespan and the number of solutions equal to best.



(a) MDTS **(b)** SDBATS **(c)** PEFT **(d)** HEFT **(e)** CPOP

Fig. 2: Schedules of the task graph in Fig. 1 with makespans.

earliest. In each iteration, the task with the highest rank is selected from the ordered list for scheduling, and then EST and EFT of that task on each VM are calculated as shown in Table III. The machine with shortest EFT is selected to schedule the current task. We use insertion based policy in every iteration in order to minimize the overall makespan. For a possible insertion of a task t_i, the idle timeslot on a VM vm_j is found when all the predecessors of the task t_i has provided the input data to vm_j. The searching for an idle timeslot continues until a suitable one is found to adjust the computational cost of the task t_i. This shows that MAD is a better approach for rank calculations to reduce the overall makespan. MDTS runs with an equal complexity of HEFT, PEFT, and SDBATS. For a given DAG, MDTS has the time complexity of $O(e \times m)$, where e is the number of edges and m is the number of VMs. For t number of tasks the time complexity is $O(t^2 \times m)$.

VI. Performance Evaluation

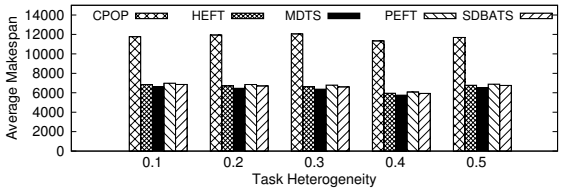
In this section, we compare the performance of MDTS with four well-known list scheduling algorithms, namely SDBATS, HEFT, CPOP, and PEFT, using random DAGs and real world applications including, LIAGs [21], GE [7], [8], and MG [22]. We observe consistent performance results using MDTS across different applications. We have implemented the proposed MDTS algorithms in CloudSim [5], which is an extendable framework for modeling and simulating cloud computing infrastructures and services. The computation and communication costs of random DAGs and real world applications are calculated using COV based method [18].

A. Comparison Metrics

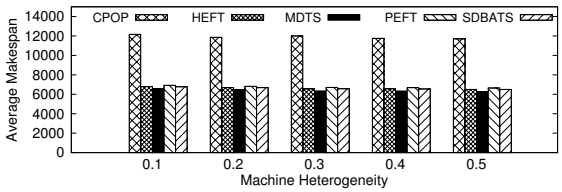
We evaluate and analyze the algorithms used in our evaluation using the following metrics.

1) Makespan: Makespan is the completion time of an exit task in a DAG that is scheduled by a scheduling algorithm.

2) Schedule Length Ratio (SLR): SLR is a major comparison metric for the output schedule of a DAG. It is defined as the ratio of the makespan to the sum of minimum computational costs of critical path tasks. A critical path in a DAG is the

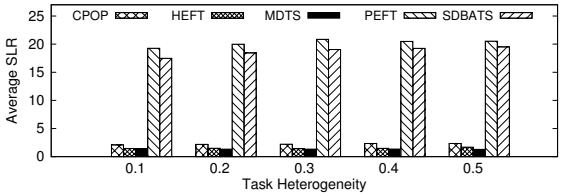


(a) Average makespan for different task heterogeneity values.

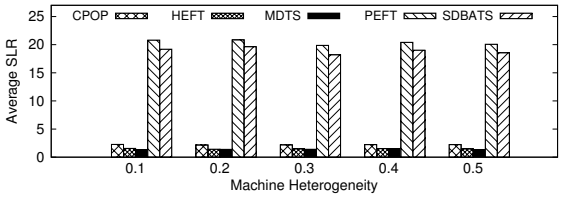


(b) Average makespan for different machine heterogeneity values.

Fig. 3: Makespan on random DAGs.



(a) Average SLR for different task heterogeneity values.



(b) Average SLR for different machine heterogeneity values.

Fig. 4: SLR on random DAGs.

longest path from the start node to the exit node. SLR can be calculated as:

$$SLR = \frac{Makespan}{\sum_{t_j \in CP_{MIN}} min(CompCost_{i,j})} \quad (9)$$

3) Speedup: The speedup is obtained by dividing the sum of the computation costs of all the tasks in the given task graph with the schedule length of the output schedule. The sum of the computation costs (i.e., sequential execution time) can be computed if all tasks are mapped to the same VM that minimizes the collective computation costs of the tasks.

$$Speedup = min\Big[\sum_{i=0}^{n} CompCost_{i,j}\Big]/Makespan, \quad (10)$$

where $t_i \in T$ and $vm_j \in VM$.

4) Efficiency: Efficiency is the ratio of the speedup to the number of VMs used in each execution of the given DAG. If

Parameter	Values
Set of tasks	T = {10, 20, 30, . . . , 100}
Communication to computation ratio	CCR = {0.5, 1, 1.5, 2, . . . , 10.0}
Set of shape parameter	S = {0.5, 1.0, 2.0}
Out degree for a node	OD = {2, 3, 4, 5, 6}
Set of task heterogeneity	V_{task} = {0.1, 0.2, 0.3, 0.4, 0.5}
Set of machine heterogeneity	V_{mach} = {0.1, 0.2, 0.3, 0.4, 0.5}

TABLE V: Parameters values for generating weighted graphs.

m is the number of VMs, the efficiency can be defined as:

$$Efficiency = Speedup/m \tag{11}$$

5) Number of Better Quality Solutions: Number of Better Quality Solutions metric shows the performance of a particular algorithm compared to the other algorithms. In our evaluation, the Number of Best Solutions (NBS), and the Number of Solutions Equal to Best (NSEB) are determined. The NBS shows that how many times a particular algorithm produces minimum makespan compared to the other algorithms. The NSEB refers to the best solution of a specific algorithm but other algorithms can also produce the same result during an experiment. The count of better quality solutions for random DAGs, GE, MG, and LIA are shown in Table IV.

B. Random Task Graphs

We generate random DAGs that takes the following parameters to generate weighted graphs:

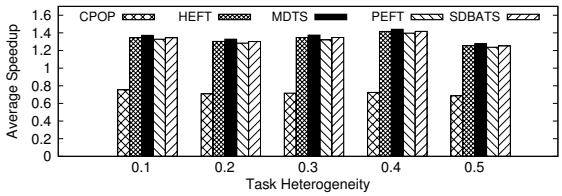
- Total count of tasks in the DAG, (n).
- We used a shape parameter (S). The height of a DAG is taken from the uniform distribution with an average value of $\sqrt{n}/S$. The height is an integer value not less than randomly generated real value. Similarly, the width of each level is randomly selected from a uniform distribution with an average of $S \times \sqrt{n}$. When $S \gg 1$, a shorter graph is generated with high degree of parallelism. When $S \ll 1$, a longer graph with low parallelism is generated.
- Out degree (OD) of a node in a DAG.
- Communication to computation cost ratio, (CCR). A DAG with low CCR is considered as a computational intensive application, and with high value indicates a communication intensive application.

We assign all required values once for a parameter used in single experiment. In our evaluation, we have generated more that 4,000 different graphs with different characteristics by using the given set of values. To generate random DAGs, we use the values shown in Table V from the related set [7].

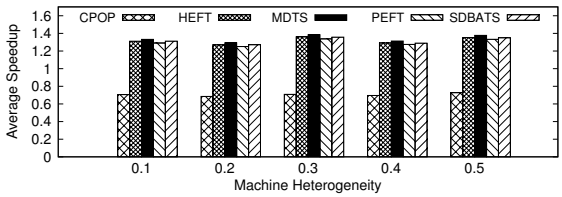
C. Performance Evaluation on Random Task Graphs

We use average makespan, average SLR, and average efficiency as performance metrics for comparing algorithms on the given DAGs. These metrics are measured for different task heterogeneity and machine heterogeneity values.

In Fig. 3(a) and Fig. 3(b), average makespan on random DAGs for different values of V_{task} and V_{mach} is presented, which shows that the proposed algorithm produces shorter schedule length. Average makespans produced by SDBATS, HEFT, CPOP, and PEFT for different V_{task} and V_{mach} values

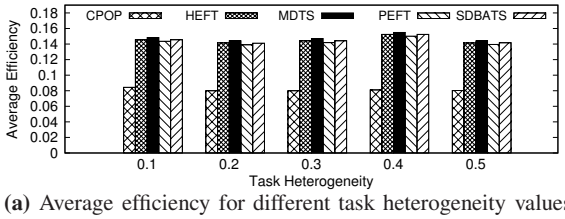


(a) Average speedup for different task heterogeneity values.

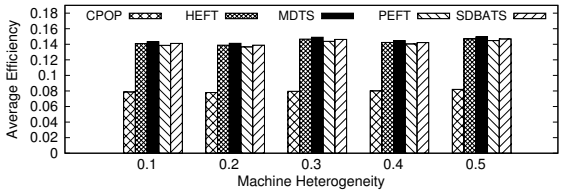


(b) Average speedup for different machine heterogeneity values.

Fig. 5: Speedup on random DAGs.



(a) Average efficiency for different task heterogeneity values.



(b) Average efficiency for different machine heterogeneity values.

Fig. 6: Efficiency on random DAGs.

indicate that the average makespan generated by MDTS is less than these algorithms for different task and machine heterogeneity values.

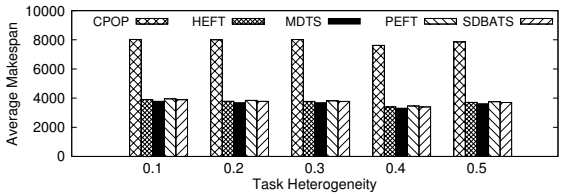
The average SLR is presented in Fig. 4(a) and Fig. 4(b), and the average speedup is shown in Fig. 5(a) and Fig. 5(b). Fig. 6(a) and Fig. 6(b) show the average efficiency of different algorithms for different task and machine heterogeneity values, respectively. The results of the average SLR and efficiency show that MDTS outperforms the other algorithms used in our evaluation because it takes into account the computational cost heterogeneity of different VMs and assigns the current task with highest median deviation to the best VM.

D. Performance Evaluation on Gaussian Elimination (GE)

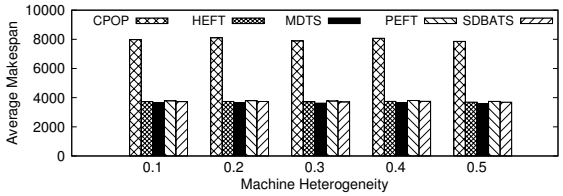
In this section, the performance of MDTS is evaluated with Gaussian Elimination (GE) [7], [8]. Here, the earlier used parameters are not required because the structures of the graphs are known in advance. In GE, the total number of nodes is represented by the matrix size (ms) and the total nodes are calculated as:

$$TotalNodes = (ms^2 + ms - 2)/2 \tag{12}$$

We take different matrix sizes ranging from 4 to 10 with an increment of 1 to generate different GE. The generated graphs

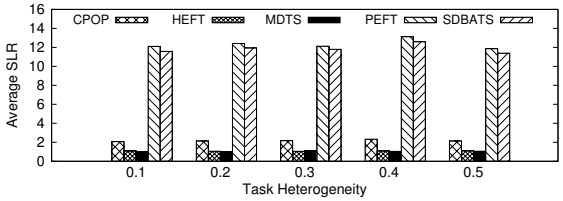


(a) Average makespan for different task heterogeneity values.

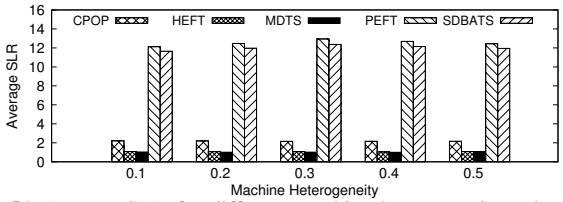


(b) Average makespan for different machine heterogeneity values.

Fig. 7: Makespan on Gaussian Elimination graphs.

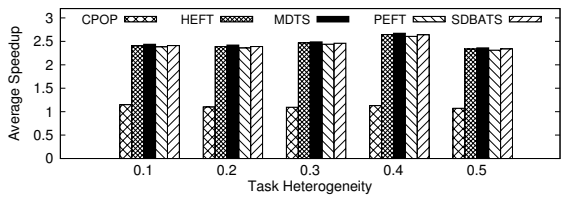


(a) Average SLR for different task heterogeneity values.

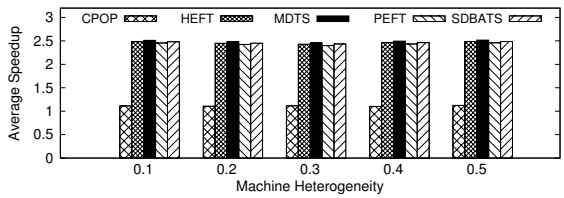


(b) Average SLR for different machine heterogeneity values.

Fig. 8: SLR on Gaussian Elimination graphs.

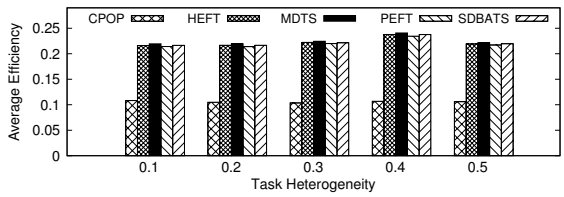


(a) Average speedup for different task heterogeneity values.

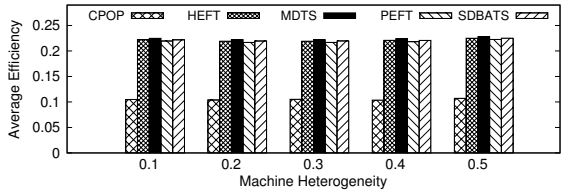


(b) Average speedup for different machine heterogeneity values.

Fig. 9: Speedup on Gaussian Elimination graphs.



(a) Average efficiency for different task heterogeneity values.



(b) Average efficiency for different machine heterogeneity values.

Fig. 10: Efficiency on Gaussian Elimination graphs.

with the given matrix parameter range have a total number of nodes from 5 to 44. The number of VMs is taken randomly from the set 3, 6, 9, 12, 15 for each GE.

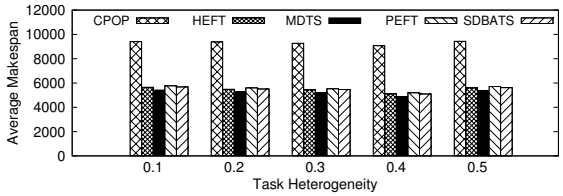
The results of the average makespan of MDTS using GE, presented in Fig. 7(a) and Fig. 7(b), show that MDTS produces shorter makespan than SDBATS, CPOP, HEFT, and PEFT for different task and machine heterogeneity values of: 0.1, 0.2, 0.3, 0.4, and 0.5. The average SLR produced by the proposed algorithm and other four algorithms is calculated for different values of task and machine heterogeneity and is shown in Fig. 8(a) and Fig. 8(b), respectively. Using results of SLR for GE, it is obvious that MDTS outperformed HEFT, CPOP, PEFT, and SDBATS. Similarly, average speedup and efficiency for all the studied algorithms is also calculated by different task and machine heterogeneity values with GE and are shown in Fig. 9(a), Fig. 9(b), Fig. 10(a), and Fig. 10(b), respectively. The results of efficiency and speedup show that MDTS is more efficient as compared to the other four algorithms used in our evaluation on GE graphs.

E. Performance Evaluation on Montage Graphs (MG)

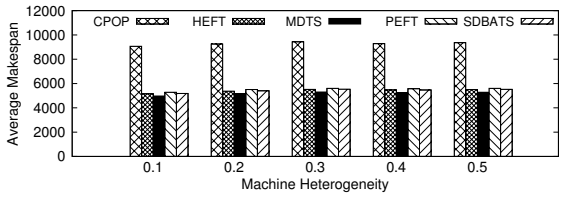
Montage Graphs (MG) [22] is another real-world application that is used for comparing the performance of the proposed algorithm with SDBATS, CPOP, HEFT, and PEFT. We created different MG of 26, 51, and 101 nodes. The performance comparison of the studied algorithms using MG for average values of makespan, SLR, speedup, and efficiency with different task and machine heterogeneity values are shown in Fig. 11(a), Fig. 11(b), Fig. 12(a), Fig. 12(b), Fig. 13(a), Fig. 13(b), Fig. 14(a), and Fig. 14(b), respectively. The results of these comparison metrics using MG show better performance of the proposed algorithm over the other four algorithms.

F. Performance Evaluation on LIGO Inspiral Analysis Graphs (LIAG)

In this section, the performance of MDTS is evaluated on another real-world application graph, LIGO Inspiral Analysis (LIAG). The LIAG is used for analysis and generation of gravitational waveforms on data that is gathered by conjoining the compact binary systems [22]. As the structure of this graph is known in advance, considering the number of task or shape parameter (S) is not required for performance evaluation using LIAG. We consider heterogeneity factors V_{task} and V_{mach} [18] for our analysis on all algorithms used in our evaluation. We used graphs of sizes 31, 51, and 101 for LIAG in our experiments. The average makespan produced by MDTS, SDBATS, HEFT, PEFT, and CPOP is shown in Fig. 15(a) and Fig. 15(b) with different values of task and machine

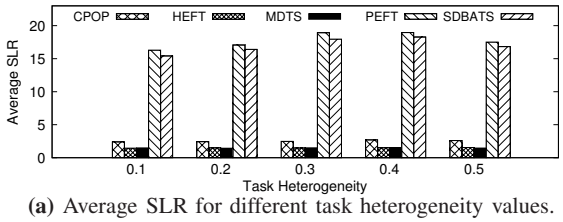


(a) Average makespan for different task heterogeneity values.

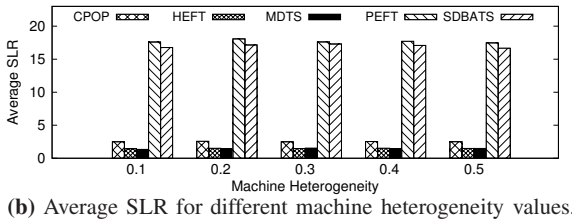


(b) Average makespan for different machine heterogeneity values.

Fig. 11: Makespan on Montage Graphs.



(a) Average SLR for different task heterogeneity values.



(b) Average SLR for different machine heterogeneity values.

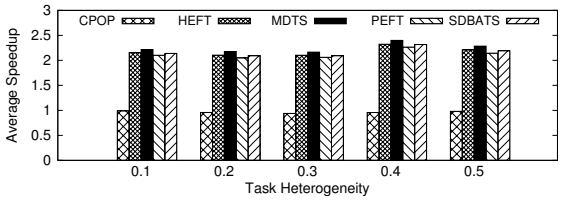
Fig. 12: SLR on Montage Graphs.

heterogeneity, respectively. The makespan results show that, on the average, MDTS outperforms the other four algorithms.

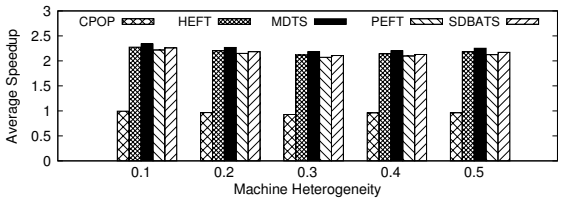
Fig. 16(a) and Fig. 16(b) show the results for average SLR for the proposed and other four algorithms with different task and machine heterogeneity values, respectively. The results show the that average SLR is less than the other four algorithms when using LIAG. The proposed algorithm outperforms HEFT, PEFT, CPOP, and SDBATS. The average speedup results shown in Fig. 17(a) and Fig. 17(b) indicate that the proposed algorithm outperforms the other four algorithms used in our evaluation for comparison. The results of average efficiency is shown in Fig. 18(a) and Fig. 18(b). Our evaluation using LIAG shows that MDTS outperforms PEFT, SDBATS, CPOP, and HEFT in terms of average efficiency with varying values of task and machine heterogeneity.

VII. Conclusions

In this paper, we present a new list-based task scheduling algorithm, Median Deviation based Task Scheduling (MDTS), with quadratic time complexity. The proposed algorithm uses median deviation for rank calculation of the given tasks. We use coefficient of variation based expected time to compute to achieve higher level of task and machine heterogeneity. The proposed algorithm also leverages median deviation that

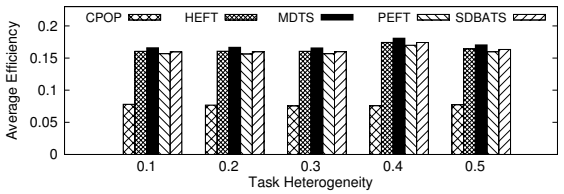


(a) Average speedup for different task heterogeneity values.

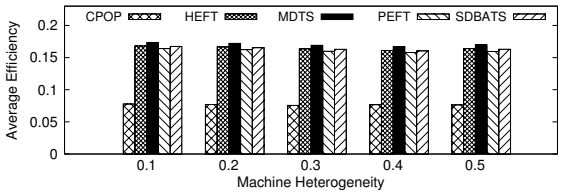


(b) Average speedup for different machine heterogeneity values.

Fig. 13: Speedup on Montage Graphs.

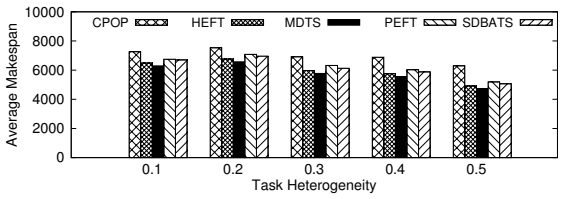


(a) Average efficiency for different task heterogeneity values.

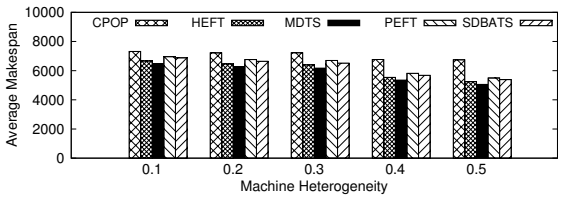


(b) Average efficiency for different machine heterogeneity values.

Fig. 14: Efficiency on Montage Graphs.



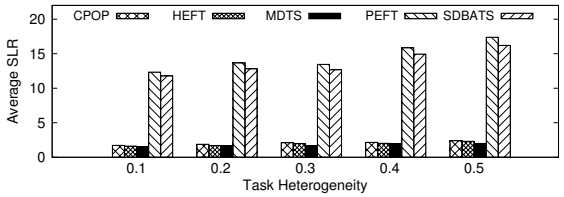
(a) Average makespan for different task heterogeneity values.



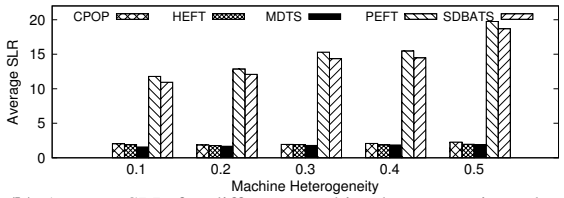
(b) Average makespan for different machine heterogeneity values.

Fig. 15: Makespan on LIGO Inspiral graphs.

has shown to be a significant parameter to acquire schedules of high quality with minimum schedule length, less schedule length ratio (SLR), higher speedup, and maximum efficiency. We compare the performance of the proposed MDTS algorithm with widely used list-based scheduling algorithms, namely Heterogeneous Earliest Finish Time (HEFT), Standard Devia-

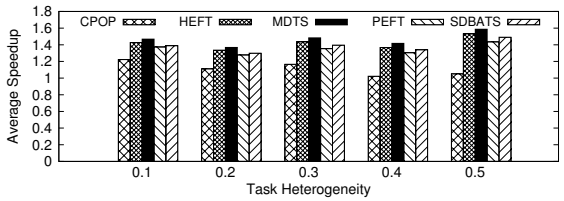


(a) Average SLR for different task heterogeneity values.

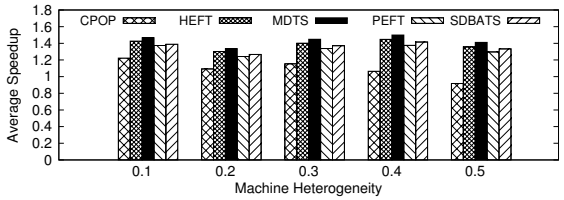


(b) Average SLR for different machine heterogeneity values.

Fig. 16: SLR on LIGO Inspiral Graphs.

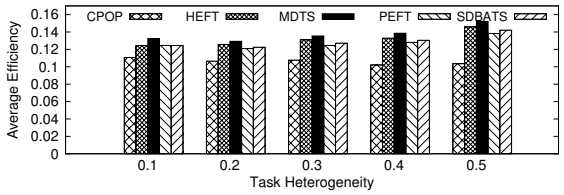


(a) Average speedup for different task heterogeneity values.

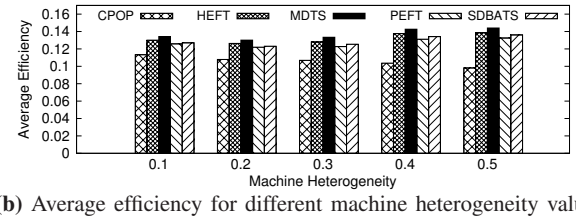


(b) Average speedup for different machine heterogeneity values.

Fig. 17: Speedup on LIGO Inspiral Graphs.



(a) Average efficiency for different task heterogeneity values.



(b) Average efficiency for different machine heterogeneity values.

Fig. 18: Efficiency on LIGO Inspiral Graphs.

tion Based Task Scheduling (SDBATS), Predict Earliest Finish Time PEFT, and Critical Path On a Processor (CPOP), using synthetic graphs and real-world applications and show that the proposed algorithm outperforms the existing algorithms used in our evaluation.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility." *Future Generation Comp. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] N. Tziritas, S. U. Khan, C.-Z. Xu, and J. Hong, "An optimal fully distributed algorithm to minimize the resource consumption of cloud applications," *CoRR*, vol. abs/1206.6207, 2012.

[3] J. Li, Q. Li, S. Khan, and N. Ghani, "Community-based cloud for emergency management," in *Proc. SoSE*, 2011.

[4] S. M. Hashemi and A. K. Bardsiri, "Cloud computing vs. grid computing," *ARPN Journal of Systems and Software*, vol. 2, pp. 188–194, 2012.

[5] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan 2011.

[6] K. S. Shin, M.-J. Park, and J.-Y. Jung, "Dynamic task assignment and resource management in cloud services by using bargaining solution," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 7, pp. 1432–1452, 2014.

[7] E. U. Munir, S. Mohsin, A. Hussain, M. W. Nisar, and S. Ali, "SDBATS: A novel algorithm for task scheduling in heterogeneous computing systems." in *Proc. IEEE IPDPS Workshops (IPDPSW)*, 2013.

[8] H. Topcuouglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[9] S. Gotoda, M. Ito, and N. Shibata, "Task scheduling algorithm for multicore processor system for minimizing recovery time in case of single node fault." in *Proc. IEEE CCGRID*, 2012, pp. 260–267.

[10] K. L. Jing Mei and K. Li, "A resource-aware scheduling algorithm with reduced task duplication on heterogeneous computing systems," *J Supercomput*, vol. 68, no. 3, pp. 1347–1377, 2014.

[11] X. Tang, K. Li, G. Liao, and R. Li, "List scheduling with duplication for heterogeneous computing systems." *J. Parallel Distrib. Comput.*, vol. 70, no. 4, pp. 323–329, 2010.

[12] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "A cluster-based data-centric model for network-aware task scheduling in distributed systems," *Int. J. Parallel Program.*, vol. 42, no. 5, pp. 755–775, Oct 2014.

[13] H. Arabnejad and J. Barbosa, "List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 682–694, 2013.

[14] E. Ilavarasan, P. Thambidurai, and R. Mahilmannan, "Performance effective task scheduling algorithm for heterogeneous computing system." in *Proc. ISPDC*. IEEE Computer Society, 2005, pp. 28–38.

[15] H. M. Fard, R. Prodan, and T. Fahringer, "Multi-objective list scheduling of workflow applications in distributed computing infrastructures." *J. Parallel Distrib. Comput.*, vol. 74, no. 3, pp. 2152–2165, 2014.

[16] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A multi-objective approach for workflow scheduling in heterogeneous environments," in *Proc. IEEE/ACM CCGRID*, 2012.

[17] Y. Dai and X. Zhang, "A Synthesized Heuristic Task Scheduling Algorithm," *The Scientific World Journal*, vol. 2014, p. 9, 2014.

[18] S. Ali, H. J. Siegel, M. Maheswaran, D. A. Hensgen, and S. Ali, "Task execution time modeling for heterogeneous computing systems." in *Heterogeneous Computing Workshop*, 2000, pp. 185–199.

[19] T. Pham-Gia and T. Hung, "The mean and median absolute deviations," *Mathematical and Computer Modelling*, vol. 34, no. 7-8, pp. 921–936, 2001.

[20] "Wikipedia Median absolute deviation," http://en.wikipedia.org/wiki/Median_absolute_deviation, accessed: 2014-07-26.

[21] "Workflow generator," https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, accessed: 2014-10-10.

[22] "Pegasus Workflow Generator," https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, accessed: 2014-07-10.