Egyptian Informatics Journal 18 (2017) 205–214

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

Green cloud environment by using robust planning algorithm

Jyoti Thaman [a,*], Manpreet Singh [b]

[a] M.M. University, Ambala, Haryana, India
[b] M.M. University, Sadopur, Ambala, India

ARTICLE INFO

Article history:
Received 19 January 2016
Revised 15 December 2016
Accepted 5 February 2017
Available online 20 February 2017

Keywords:
Planning algorithms
Scheduling algorithms
Ready wueue
Robust
Cloud computing

ABSTRACT

Cloud computing provided a framework for seamless access to resources through network. Access to resources is quantified through SLA between service providers and users. Service provider tries to best exploit their resources and reduce idle times of the resources. Growing energy concerns further makes the life of service providers miserable. User's requests are served by allocating users tasks to resources in Clouds and Grid environment through scheduling algorithms and planning algorithms. With only few Planning algorithms in existence rarely planning and scheduling algorithms are differentiated. This paper proposes a robust hybrid planning algorithm, Robust Heterogeneous-Earliest-Finish-Time (RHEFT)[1] for binding tasks to VMs. The allocation of tasks to VMs is based on a novel task matching algorithm called Interior Scheduling. The consistent performance of proposed RHEFT algorithm is compared with Heterogeneous-Earliest-Finish-Time (HEFT)[2] and Distributed HEFT (DHEFT)[3] for various parameters like utilization ratio, makespan, Speed-up and Energy Consumption. RHEFT's consistent performance against HEFT and DHEFT has established the robustness of the hybrid planning algorithm through rigorous simulations.

1. Introduction

Diverse resources with varied capabilities and connected through high speed interconnecting network provides new platform for distributed processing. Cloud and Grid computing evolved from such aggregation of resources. These are primarily maintained by service providers (Amazon, IBM, Microsoft, etc). Users subscribe for services from these platforms and submit their tasks for processing. Users are served by allocating their tasks to various resources and executing them. When tasks executions times, inter-task dependencies and inter-task data transfer size is known then such task model is called static model. User's submissions are processed in clouds by subjecting tasks to resources. Resource usage in clouds depends upon the types and sequence of tasks and resources. Work flow technologies are used to deal with increasing complex data, data-intensive application, simulations and analysis. These technologies are also used to schedule computational tasks on distributed resources, to manage dependencies among tasks and to stage data sets into and out of execution sites [8]. These workflows are used to model computations in many scientific disciplines [9].

A number of task scheduling algorithm are proposed in literature which are broadly classified into list-scheduling algorithms, level-by-level scheduling, batch scheduling, duplication based scheduling, dependency scheduling, batch dependency scheduling algorithm, Genetic Algorithm (GA) based scheduling algorithms and hybrid algorithm. List scheduling algorithm creates a list of task while respecting task dependency. Tasks in list are processed in order of their appearance in the task list. The performance of such algorithm is comparatively better than other categories of algorithms. Level-by-level scheduling algorithms consider tasks of one level in task-graph such that task considered are independent of each other. This set of tasks may not include all the tasks in ready queue. In Genetic algorithm based solution schedules are reasonably acceptable but the computational complexity of

* Corresponding author.
E-mail addresses: jyoti.thaman77@gmail.com (J. Thaman), dr.manpreet.singh.in@gmail.com (M. Singh).

[1] Robust Heterogeneous-Earliest-Finish-Time (RHEFT).
[2] Heterogeneous-Earliest-Finish-Time (HEFT).
[3] Distributed HEFT (DHEFT).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.eij.2017.02.001
1110-8665/

algorithm is relatively high. Hybrid algorithm explores various combinations of existing classes of scheduling algorithms.

Task scheduling in heterogeneous systems is considered in Het erogeneous-Earliest-Finish-Time (HEFT) [7], Duplication based HEFT [21] and Deadline–Budget Constrained Scheduling (DBCS) [4]. In HEFT [7], authors proposed a ranking of tasks on the basis of bandwidth, task's length, and parent-child relationships. Tasks are considered for execution in order of their rank in decreasing order. Duplication based HEFT used the concept of task duplication and utilized the free cycles of VMs for execution of duplicate tasks. Distributed HEFT (DHEFT) exploits the concept of distributed approach and better exploits the concept of VM level availability for better task-VM mappings [20]. This paper proposes a variant of HEFT called Robust HEFT (RHEFT) by using a hybrid approach and a novel scheduling algorithm for set of independent tasks. Tasks are ranked as per ranking method of HEFT, which is followed by grouping of free tasks into same group. Groups are processed in order of their creation. Tasks in a group are processed such that scheduling reduces the variance in difference of task's execution time and VM's mean execution time. Section 2 presents the related works, Section 3 presents the preliminary. Section 4 presents Interior Scheduling (IS) and RHEFT algorithm. Section 5 presents simulation set-up and performance discussion. Finally, concluded in Section 6 with a future direction.

2. Related works

This section presents a brief review of several research works done in the field of scheduling. Research work in [10–15,8,14] proposed scheduling solutions for workflows. Work in [15,18,17,19,5,1] refers to solution for independent tasks. [4,7] presents scheduling algorithms for heterogeneous systems. [3] presents a taxonomy of scheduling of tasks in clouds and grids.

Research work in [8], provided multiple scientific applications including astronomy, bioinformatics, earthquake science, and gravitational-wave physics is based on novel workflow profiling tools that provide detailed information (includes I/O, memory and computational characteristics) about various computational tasks that are present in the workflow. In [10], authors described an extension to Pegasus whereby resource allocation decisions are revised and described how adaptive processing has been retrofitted to an existing workflow management system; a scheduling algorithm that allocates resources based on runtime performance. The results were evaluated using grid middleware over clusters. In [11], authors proposed a dynamic critical-path-based adaptive workflow scheduling algorithm for grids, which determines efficient mapping of workflow tasks to grid resources dynamically by calculating the critical path in the workflow task graph at every step. In [12], authors designed and analyzed a two-phase scheduling algorithm for utility Grids, called Partial Critical Paths (PCP), that was used to minimize the cost of workflow execution while meeting a user defined deadline and also proposed two workflow scheduling algorithms one was one-phase algorithm which is called IaaS Cloud Partial Critical Paths (IC-PCP), and a two-phase algorithm which is called IaaS Cloud Partial Critical Paths with Deadline Distribution (IC-PCPD2) that have a polynomial time complexity which make them suitable options for scheduling large workflows. Work in [13], proposed a new dynamic task scheduling algorithm for Heterogeneous environments called Clustering Based HEFT with Duplication (CBHD). The CBHD algorithm is considered an amalgamation between the most two important task scheduling in Heterogeneous machine, The Heterogeneous Earliest Finish Time (HEFT) and the Triplet Clustering algorithms. CBHD outperforms the HEFT and Triplet algorithm by decreasing the makespan by 2.5%. It also achieves better load balancing than the HEFT algorithm by 70%, and it increases processors utilization by 10% with respect to the HEFT and Triplet algorithms. In [14], the authors presented a Hybrid Cloud Optimized Cost scheduling algorithm that decides which resources should be leased from the public cloud and aggregated to the private cloud to reduce costs while achieving the established desired execution time. HCOC tried to optimize the monetary execution costs while maintaining the execution time lower than Deadline.

In [15], authors proposed a novel heuristic for scheduling of set of independent tasks, called Balanced Minimum Completion Time (BMCT). First phase performs initial allocation using FCFS. In next phase BMCT tries to minimize the complete execution time by swapping tasks between machines. This results in balancing of load among the machines. BMCT has shown promising results when compared with Dynamic Level Scheduling (DLS) [16], Heterogeneous Earliest Finish Time (HEFT) [7]; Critical Path On a Processor (CPOP) [7] etc. under consistent heterogeneous, partially consistent heterogeneous and inconsistent heterogeneous environments. In [5], presented multi-objective PSO based optimization algorithm for dynamic environment of clouds and optimize energy and processing time. Proposed algorithm provides an optimal balance results for multiple objectives. The experimental results illustrated that the proposed methods out-performed the Best Resource Scheduling (BRS) and Random Selection Algorithm (RSA). In [17], authors proposed, two task scheduling algorithm namely user-Priority Awarded Load Balance Improved Min-Min Scheduling Algorithm (PA-LBIMM) and Load Balance Improved Min-Min (LBIMM) scheduling algorithm were proposed with objectives to decrease job's completion time, improve the load balance and satisfy users' priority demands in the cloud. LBIMM performs in two phases namely first phase is min-min and second phase is preemption of smaller tasks from heavenly loaded resources and migrate them to resources with fastest completion time for preempted job. In PA-LBIMM tasks are divided into two groups based high or low priority. Initially, allocation is done to tasks with higher priority and then tasks of lower priority are allocated to resources. Initial allocation is realized through Min-Min scheduling algorithm. In Next phase load balancing based on preemption of tasks is performed. Result reported in paper proves that PALBIMM and LBIMM outperform the Min-Min algorithm in all aspects. In [1], authors proposed an energy efficient scheduling algorithm, (EEVS) considering the deadline constraint. EEVS can support DVFS well. From the computation of total energy of a PM, authors conclude that there is an optimal frequency for it to process certain VMs. Based on the optimal frequency; authors define the optimal performance–power ratio to weight the heterogeneities of the PMs. The PM with highest optimal performance–power ratio will be used to process the VMs first unless it does not have enough computation resources. Finally the cloud should be reconfigured to consolidate the computation resources of the PMs to further reduce the energy consumption. EEVS consumes less energy and processes more VMs successfully than the existing methods. In [4], presented a heuristic scheduling algorithm with quadratic time complexity that considers two important constraints for QoS-based workflow scheduling, time and cost, named Deadline–Budget Constrained Scheduling (DBCS) for heterogeneous systems. DBCS has the lowest time complexity (quadratic time complexity), while other algorithms mostly have cubic or polynomial time complexities. In terms of the quality of results, DBCS achieves rates of successful schedules similar to higher-time complexity algorithms for both random and real application workflows on diverse platforms.

In [18], authors presented two novel dynamic scheduling algorithms for heterogeneous and federated cloud system. The objective was to achieve resource optimization mechanism for preempt-able applications in autonomous heterogeneous cloud environment. Authors also proposed a dynamic procedure with

updated information. The procedure helped to achieve considerable improvement in resource utilization and energy efficiency in any given resource contentious environment. In [19], authors had presented a thorough review of workflow scheduling algorithms under different classes. Authors proposed a paradigm to classify the existing workflow scheduling algorithms and presented a useful concluding remark. In [6], authors presents a workflow schedule optimization algorithm (MER) that can be used with any existing workflow scheduling algorithm as a post-processing technique. It consists of three major phases to first find the trade-off points between the minimum makespan increase and the maximum resource usage reduction, and to consolidate tasks and resources leading to significant improvement in resource efficiency. Based on results from extensive experiments with five real-world scientific workflows confirm the claims. Finally, this work study revealed that by allowing a small degree of makespan increase, such exploitation reduces resource usage far greater than any incurred makespan increase. Based on results obtained from our extensive simulations using scientific workflow traces, we demonstrate MER is capable of reducing the amount of actual resources used by 54% with an average makespan increase of less than 10%.

In [3], authors Identified & explained the aspects and classifications unique to workflow scheduling in the cloud environment in three categories, namely, scheduling process, task and resource. Lastly, review of several scheduling techniques are included and classified onto the proposed taxonomies. The proposed taxonomies serve as a stepping stone for those entering this research area and for further development of scheduling technique. The present taxonomies of cloud workflow scheduling problems and techniques based on analysis of existing research literature, which classifies techniques in grid workflow scheduling, by adding new aspects unique to cloud computing and refining some existing ones. It is noticeable that almost every technique proposed so far has the assumption that resources are virtual machine instances (i.e. infrastructure-as-a-service).

Most of the works reviewed in this section refers to scheduling with objective of reducing makespan, improving resource utilization and reducing financial liabilities. Most proposals lack basic consideration like hybrid of scheduling techniques. With limited scope of improvement in scheduling schemes and without considering out-of-box alteration, this work presents a mathematical viable solution for improving the performance of scheduling algorithms.

3. Preliminary

In this section HEFT [7] algorithm is discussed as preliminary to this research. Heterogeneous-Earliest-Finish-Time (HEFT) algorithm was proposed by Topcuoglu et al. The algorithm is based on the computation of task's rank. Algorithm computes average execution time for each task and average communication time between resources of two successive tasks on the basis of parent-child relationship between concerned tasks. Let $time(T_i, r)$ be the execution time of task T_i on resource r and let R_i be the set of all available resources for processing of T_i. The average execution time of a task T_i is defined as

$$\overline{t_i} = \frac{\sum_{r \in R_i} time(T_i, r)}{|R_i|} \tag{1}$$

Let time (e_{ij}, r_i, r_j) be the data transfer time between resources r_i and r_j which process the task T_i and task T_j respectively. Let R_i and R_j be the set of all available resources for processing T_i and T_j respectively. The average transmission time from T_i to T_j is defined by:

$$c_{ij} = \frac{\sum_{r_i \in R_i, r_j \in R_j} time(e_{ij}, r_i, r_j)}{|R_i||R_j|} \tag{2}$$

Then tasks in the workflow are ordered in HEFT based on a rank function. For an exit task the rank value is:

$$Rank(T_i) = \overline{t_i} \tag{3}$$

The rank values of other tasks are computed recursively based on Eqs. (1)–(3) as shown in Eq. (4).

$$Rank(T_i) = \overline{t_i} + \max_{T_j \in succ(T_i)} (\overline{c_{ij}} + Rank(T_j)) \tag{4}$$

HEFT is based on global approach on scheduling without taking into consideration the complete set of tasks in ready queue. This poor approximation of ready queue tasks affects the performance of HEFT in highly resource available environment. HEFT performs allocation of tasks to VMs on the basis of ranks. HEFT is accepted widely in various projects of significant importance like ASKALON project [22] to provide scheduling for a quantum chemistry application, WIEN2K [23], and a hydrological application, Invmod [24] on the Austrian Grid.

4. Robust HEFT: A hybrid Planning algorithm

This section presents a hybrid planning algorithm for cloud environment which addresses the limitations of HEFT. Section 3 discussed HEFT planning algorithm which is one of the most promising planning algorithm. HEFT works on the centralized approach and utilizes the ranks of the tasks as decision parameter while subjecting next tasks to some free VM. Ranked tasks are arranged and scheduled in non-increasing order by their ranks. Next-ranked task is assigned to next free VM. This assignment/mapping of ranked task to free/available VM is random. No suitability criteria were used for this mapping. As a result HEFT could poorly approximate the ready queue. The schedules obtained from HEFT were not able to utilize the available resources in best possible way. These many limitations of HEFT provide motivation for some improvements in the functioning of HEFT.

4.1. Robust HEFT (RHEFT)

The improvement in working of planning algorithms has been presented in this section through a hybrid of HEFT and Interior Scheduling. The working of RHEFT is divided into three phases. In phase 1, HEFT is used for generation of tasks which are sorted on the basis of ranks. The working principle of HEFT is explained in Section 3. The ranks are computed using {Eqs. (1)–(4)}. The benefits of HEFT includes that a non-linear Task graph is converted to a linear list of tasks. A visible limitation of HEFT is that the task scheduled next is only a member of set of tasks in ready queue. This limitation not only reduces the utilization of resources but also increases the length of schedules. Reduced utilization and longer schedule length not only affects energy consumption but also proves to be economically inefficient.

Keeping these limitations in view, an extension of HEFT is proposed by using hybrid of HEFT and IS in this work. New planning strategy is named as Robust HEFT (RHEFT).

In phase 2, RHEFT divides the resultant ranked tasks into several sets, where each set contains a set of independent ranked tasks. Phase 2 outputs are the set of tasks ready for scheduling. Each such set represents a bigger portion of set of ready tasks. Phase 2 begins at step 9 and finish at Step18, in algorithm presented in Fig. 1. The output of phase 2 is directed acyclic graph where each node represents the set of independent tasks identified in phase 2 of RHEFT {Fig. 2}.

In phase 3, IS scheduling is applied on set of independent tasks. Phase 3 begins at Step 19 in RHEFT algorithm presented in Fig. 1. IS scheduling approach is presented ahead in the section.

4.2. Interior scheduling (IS)

Interior scheduling approach is a novel idea for mapping set of independent tasks to available VMs. The mapping utilizes the statistical characteristics of the VMs and set of tasks at hand.

Let $Task = \{T_1, T_2 \ldots, T_n\}$ {Table 1} represents the set of tasks with their execution requirements. Similarly, assume that $VM = \{M_1 M_2, \ldots, M_l\}$ represents the set of available VMs with their capacity. An execution matrix may be computed using values in set *Task* and *VM*.

$$MAT = \begin{bmatrix} e_{1,1} & \cdots & e_{a,n} \\ \vdots & \ddots & e_{a,n} \\ e_{l,1} & \cdots & e_{a,n} \end{bmatrix} \tag{5}$$

where $e_{i,j}$ represents the execution time of task T_j while executing on M_i. Using *MAT* matrix we can compute average of each row as $VM.MeanExecutionTime_i = (e_{1,1} + e_{1,2} + \ldots e_{1,n})/n$. This value can be used in Eq. (5) for minimizing the variance.

$$(\sigma^2 = (VM.MeanExecutionTime - Task.ExecutionTime)^2) \tag{6}$$

Using this as objective we have considered following example.

Consider an example where, *Task*

$$= \left\{ \begin{matrix} 78,92,23,33,55,77,88,78,102, \\ 23,33,55,106,85,78,91, \\ 23,33,55,79,88,78,92, \\ 26,33,56,74,88,79,105 \end{matrix} \right\} \text{ and } VM = \{12,7,12,11\}.$$

Numerical values in *Task* represents the MIPS of tasks. In total 30 tasks were considered. Numerical values in *VM* represents the MIPS of VMs and total 4 VMs were considered. A comparison in performance of Min-Max, Max-Max and IS was performed on utilization and makespan parameters. The comparison has been shown in Table 2.

The results shown in Table 2 uses min–max, max-max and IS for task-machine mappings. In IS approach any tasks T_i is mapped to

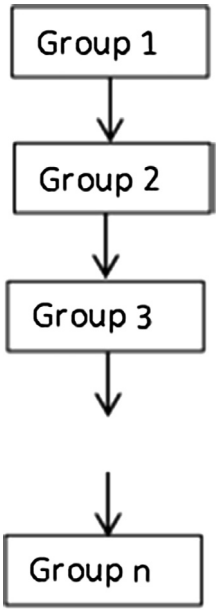


Fig. 2. Output directed acyclic graph of phase 2 in RHEFT.

Table 1
Notation table.

S. No.	Notation	Meaning	Values
1.	T_i	i^{th} Task	–
2.	$T(1 \times t)$	Task Length Matrix	–
3.	$M(m \times 1)$	VM Capacity Matrix	–
4.	Q	FIFO Queue	–
5.	$S(m \times t)$	Output Schedule Matrix	–
6.	S_i	i^{th} Set of independent Tasks	–
7.	M_i	i^{th} Virtual Machine	–
8.	$L(m \times t)$	Load Matrix	–
9.	m	Number of VMs	{5, 10, 20}
10.	t	Number of Tasks	{50,100}

Algorithm: RHEFT

1. Calculate mean execution time for each task T_i by using equation 1
2. Calculate mean data transfer delay between tasks and their successors in a task graph or workflow by using equation 2
3. Calculate Rank of each task T_i by using equations 3 and 4
4. Construct a queue Q by insertion of tasks in descending order by their Rank
5. Construct a set S_j for addition of tasks
6. Compute Load Matrix $L\ (m * t)$ from M and T.
7. Compute mean execution times $MEM(m * 1)$ for each machine using $L\ (m * t)$
8. Initialize $S\ =\ Zeros\ (m, t)$ // Zero Matrix
9. **While** Q not empty **do**
10. $\quad T_i\ =\ DeQueue\ (Q)$
11. $\quad$ **If** $Parent(T_i) \notin S_j$ && $Child(T_i) \notin S_j$
12. $\quad\quad$ Add task T_i to set S_j
13. $\quad$ **Else**
14. $\quad\quad j\ = j + 1;$
15. $\quad\quad$ Construct a set S_j for addition of tasks
16. $\quad\quad$ Add task T_i to set S_j
17. $\quad$ End **If**
18. End **While**
19. **For** $i\ =\ 1\ to\ t$ **do**
20. $\quad$ Identify the Machine id (M_i) which is free and can be subjected next task
21. $\quad$ Identify the tasks id (T_j) in S_j which if allotted to M_i results in minimum increase in variance of execution times of completed and newly submitted task T_j
22. $\quad$ Set $S\ (i, j)\ =\ L\ (i\ , j)$
23. End **For**
24. Print values of S matrix as output schedule.
25. End **RHEFT**

Fig. 1. Robust HEFT planning algorithm for workflows in clouds.

Table 2
Performance comparison based on Makespan and Utilization of Min-Max, Max-Max and IS

Scheme	Min-Max	Max-Max	IS
Makespan	62.92	56.83	49
Utilization	81.11	78.74	98.59

an M_j only if the execution time ($e_{j,i}$) has smallest difference as compared mean execution time on M_j. This principle is expressed in Eq. (5) where we strive to reduce the variance of tasks assigned to each VM for a given set of independent tasks. Improved performance of IS in example above validates the strength of task-VM mapping criteria discussed above.

Each time IS approach selects a task to be scheduled next on a particular VM such that Eq. (5) is satisfied. The selected task is most appropriate task to be schedules next on given VM. In fact, this approach identified a task whose execution time characteristics for given VM exhibits correlations with execution time characteristics of tasks already submitted or completed on given VM. This approach can be applied to overcome the non-aligned task allocation/binding issues in other task mapping or scheduling algorithms and heuristics.

5. Simulation and analysis

Simulation environment and various performance characteristics of different planning algorithms are presented in this section. Performance analysis is presented here is based on simulation in WorkflowSim.

5.1. Simulation setup

Simulation is carried out by using WorkflowSim [20] configured in Eclipse on an Intel Core 2 Duo, 2.0 GHz Linux based laptop. Simulation is considered for task sizes equal to 50 and 100. Numbers of VMs considered for simulation purpose were equal to 5, 10 and 20 respectively. Various VM characteristics as defined in WorkflowSim are retained as-is where-is.

Each VM considered in simulation possessed 1000 MIPS, 512 MB RAM, bandwidth 1000 MB/s, Processing Elements (PEs) 1 and Image Size 10,000. VM architecture is inherited from 'Xen'. Besides this Space Shared scheduling of tasks was considered for the simulation purpose. Maximum power consumption rate for

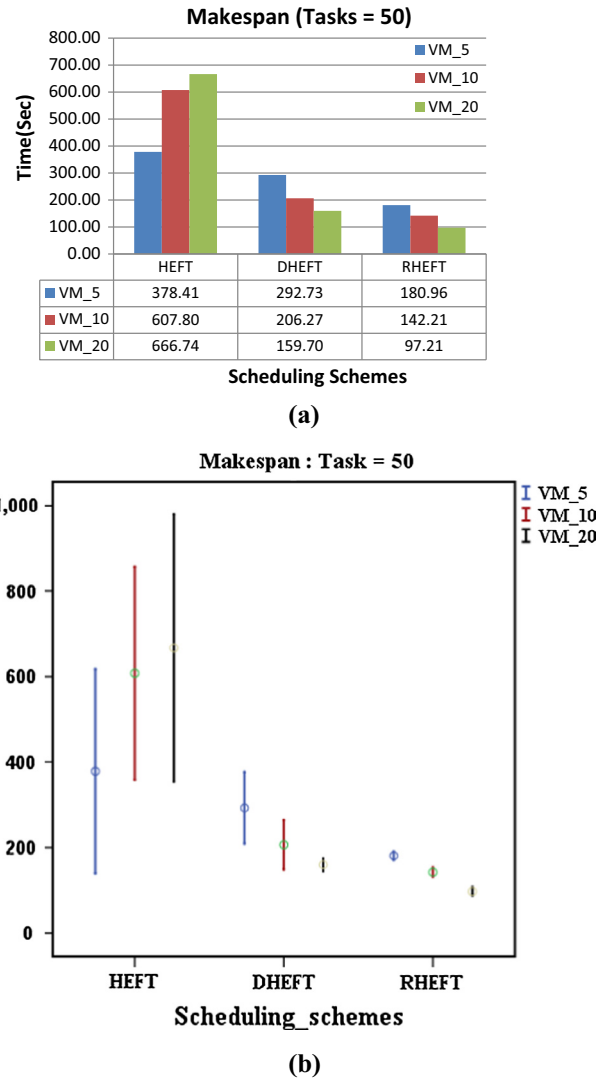


Fig. 3. (a): Makespan characteristics of various scheduling/planning algorithms using 50 tasks (b) makespan error graphs of various scheduling/planning algorithms using 50 tasks (with C.I. = 95%).

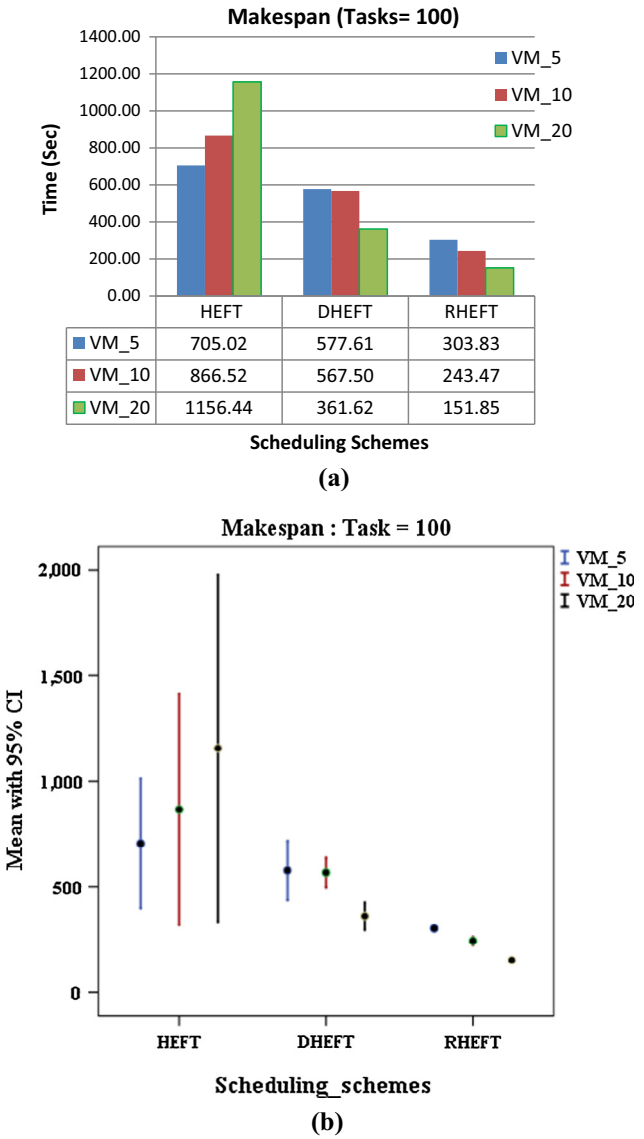


Fig. 4. (a): Makespan characteristics of various scheduling/planning algorithms using 100 tasks (b) Makespan error graphs of various scheduling/planning algorithms using 100 tasks (with C.I. = 95%).

VMs is considered fixed to 250 Watts/s. Extensive simulation of WorkflowSim supported planning algorithms like HEFT, DHEFT and new proposal RHEFT is considered. Simulation based output is drawn in Figs. 2–9 respectively. To represent static task model, Montage workflow for 50 and 100 tasks is considered for close imitation of CPU intensive tasks. Simulation is performed to study the impact on Makespan, Utilization, Energy consumption and Speed-up. Next subsection presents a detailed discussion on various performance parameters.

5.2. Performance discussion

Performance plots of RHEFT and other scheduling schemes like, HEFT and DHEFT are shown in Figs. 3–10. Tables 3–6 provides in-depth error analysis at Confidence Interval (CI = 95%). Various significant parameters which are relevant for context in cloud computing are as follows.

a. *Makespan:* Makespan is defined as the time span between the instant when first task is scheduled and the instant when last task completed the execution. Any parallel execution of tasks reduces the makespan characteristics.

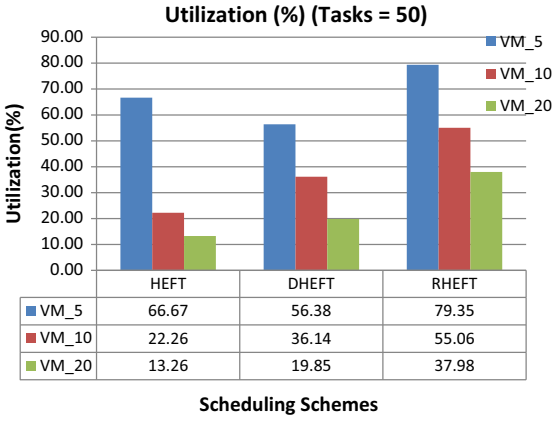
b. *Utilization:* Utilization is defined as the ratio of duration of actual usage and duration of actual availability. Improved utilization reduces the makespan characteristics.

c. *Energy Consumption:* Resources in clouds consumes energy from the moment when they are allocated for execution of tasks of users. Improved utilization reduces the span of usage. Energy consumption is defined in Watts. Let $power^{n}_{max}$ exploit maximum power consumed by n^{th} server. The idle server consumes nearly 70% of a fully utilized server [2]. Power consumption by the server n^{th} at any instant of time t is [2].

$$power^{n}(t) = power^{n}_{max} * 0.70 + power^{n}_{max} * 0.30 * U_n \quad (7)$$

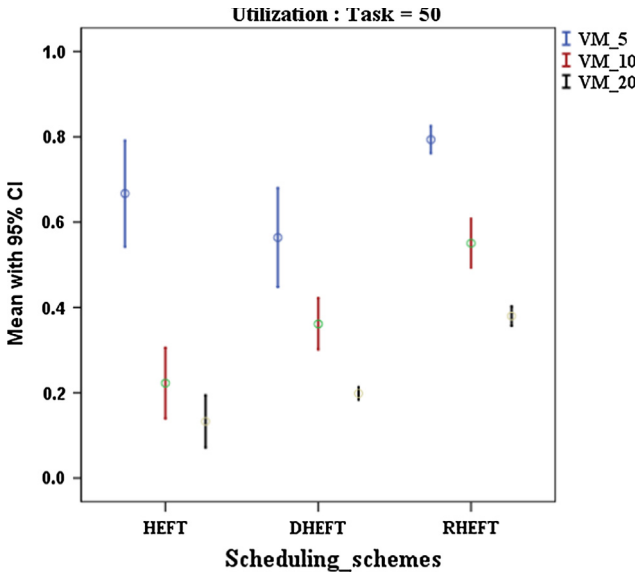
where $U_n(t)$ represent utilization at that instant of time. Servers consume a lot of power even if they are idle. It is better if idle or lightly loaded server nodes may be vacated and switched-off.

d. *Speed-up:* Speed up is defined as the ratio of makespan of parallel execution of set of tasks to makespan of sequential execution of tasks.



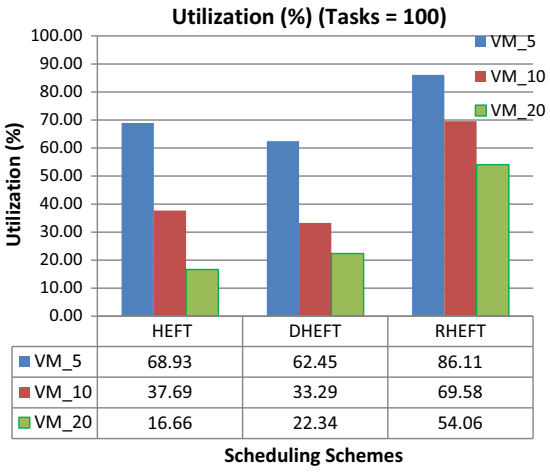
	HEFT	DHEFT	RHEFT
VM_5	66.67	56.38	79.35
VM_10	22.26	36.14	55.06
VM_20	13.26	19.85	37.98

(a)



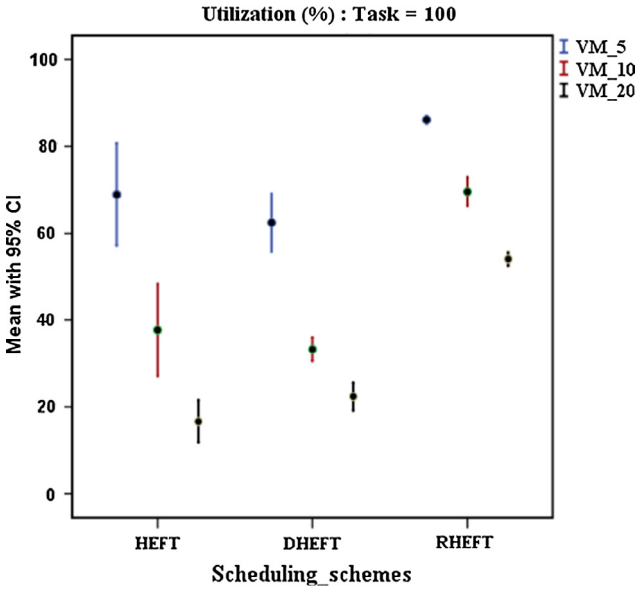
(b)

Fig. 5. (a) Utilization characteristics of various scheduling/planning algorithms using 50 tasks (b) Utilization error graphs of various scheduling/planning algorithms using 50 tasks (with C.I. = 95%).



	HEFT	DHEFT	RHEFT
VM_5	68.93	62.45	86.11
VM_10	37.69	33.29	69.58
VM_20	16.66	22.34	54.06

(a)



(b)

Fig. 6. (a) Utilization characteristics of various scheduling/planning algorithms using 100 tasks (b) Utilization error graphs of various scheduling/planning algorithms using 100 tasks (with C.I. = 95%).

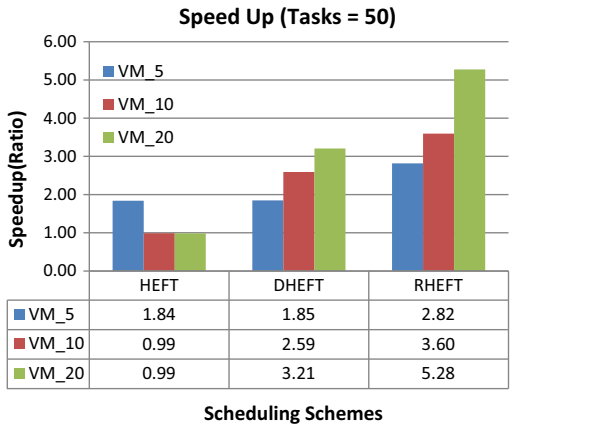
Figs. 3(a) and 4(a) presents makespan characteristics of HEFT, DHEFT and RHEFT for 50 tasks and 100 tasks respectively. The bars for HEFT with 50 and 100 tasks exhibit that when VM are increased from 5 to 10 and from 10 to 20 VMs respectively, makespan is on increasing spree. Rather with the increase of resources it should decrease. Another conclusion that can be drawn is that when tasks are increased from 50 to 100 tasks, slope of makespan characteristic for HEFT turns from positive to negative. Negative slope confirms that with more tasks HEFT better utilizes more resources than with less number of tasks. Although more VMs are available for the execution of same set of tasks, but execution time in HEFT is increasing and is highest as compared to other schemes plotted in Figs. 3(a) and 4(a). The reason for this kind of behavior is attributed to fact that HEFT considers smallest set of tasks for allocation from all the tasks in ready queue.

DHEFT used the concept of distributed approach and maps the tasks without computing ranks. Distributing the decision of task-VM mapping and considering Earliest Finish Time First approach, DHEFT improves makespan characteristics in comparison to HEFT.

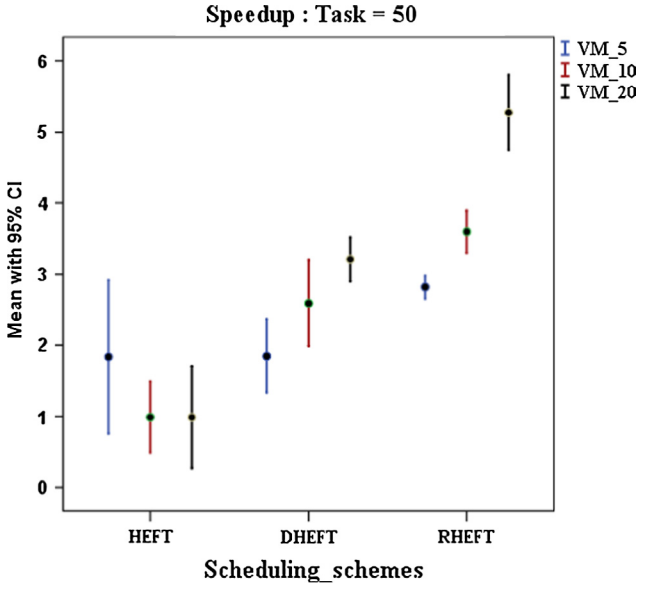
In RHEFT, phase 2 identifies a subset of independent or free tasks which better approximates set of tasks in ready queue. In phase 3 IS, is used to schedule set of independent tasks on available resources. IS improves the makespan characteristics by generating a schedule based on {Eq. (6)}. Phase 3 is hybrid phase and advances the scheduling from global to sub-local level. This characteristic of RHEFT results in improvement of makespan characteristics in comparison to HEFT and DHEFT. Figs. 3(a) and 4(a) presents makespan characteristics of HEFT, DHEFT and RHEFT respectively.

Standard Error Graphs shown in Figs. 3(b) and 4(b) respectively, drawn at Confidence Interval (CI = 95%) of 95, exhibits that HEFT and DHEFT has shown a lot of variations in results over repeated experimentation. RHEFT has resulted in minimum error at CI = 95%. The increases in number of tasks as well as increase in numbers of resources, both are better utilized in RHEFT. RHEFT has shown minimum error. Tables 3a and 3b presents lower and upper bounds of standard error w.r.t. mean makespan statistics, for 50 and 100 tasks respectively. Lower range of RHEFT dictates robust behavior of RHEFT.

Figs. 5 and 6 plots the utilization performance and error graphs at CI = 95%, for HEFT, DHEFT and RHEFT for 50 and 100 tasks

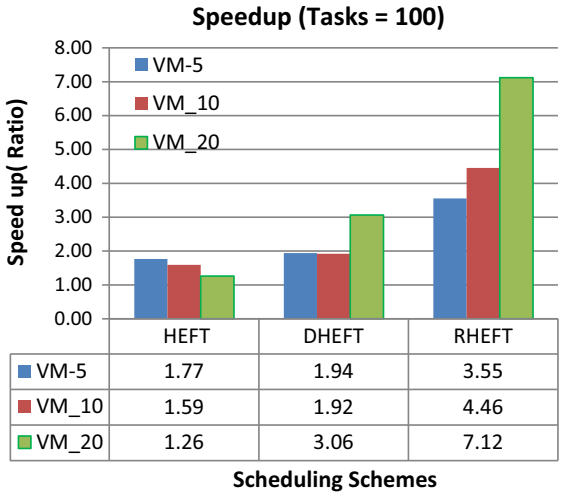


(a)

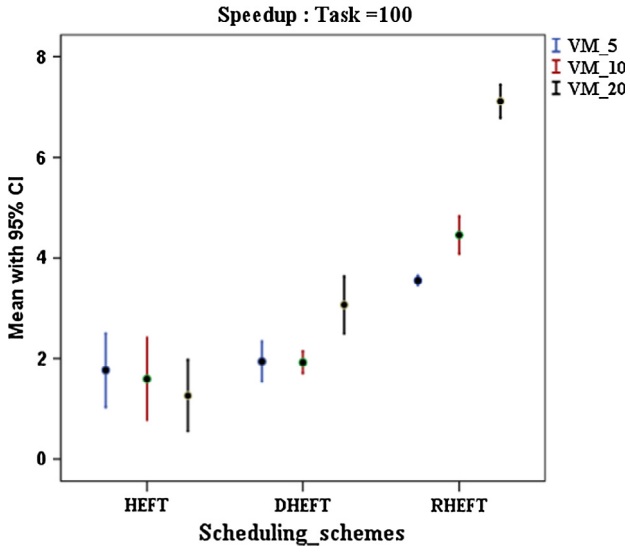


(b)

Fig. 7. (a) Speed up characteristics of various scheduling/planning algorithms using 50 tasks (b) Speed up error graphs of various scheduling/planning algorithms using 50 tasks (with C.I. = 95%)



(a)



(b)

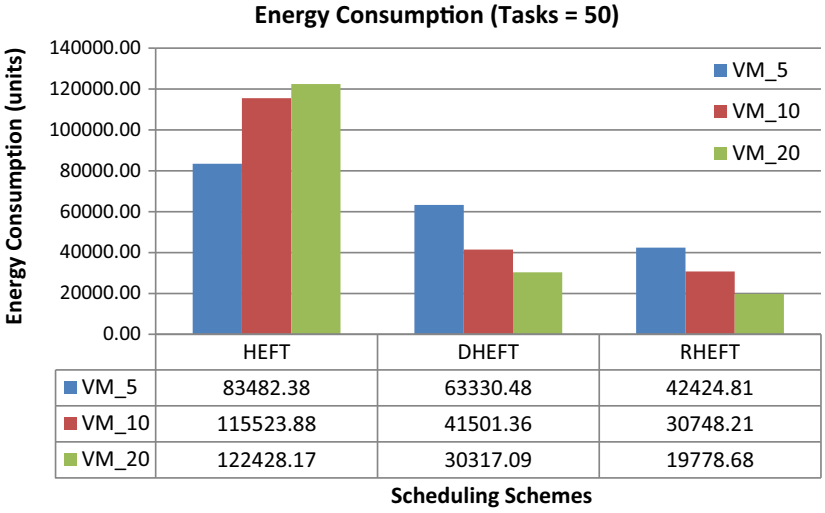
Fig. 8. (a) Speed up characteristics of various scheduling/planning algorithms using 100 tasks (b) Speed up error graphs of various scheduling/planning algorithms using 100 tasks (with C.I. = 95%).

respectively. HEFT categorically selects tasks only on the basis of ranks of tasks. This results in under-utilized resources. This is shown in Figs. 5(a) and 6(a). Even with the increase in resources, HEFT fails to use available extra resources. Increase of resources doesn't improve the performance of HEFT rather it is sheer waste of resources. HEFT has improved in terms of utilization with increase in number of tasks, i.e., when tasks are increased from 50 to 100 utilization improved marginally at VM = 10 and VM = 20. For a given set of tasks, selection of tasks for execution is independent of available resources. Extra resources are thus waste in HEFT. In case of DHEFT, no ranks were calculated. It was based on the principle of Earliest Finish Time. In DHEFT, a better Task-VM mapping was resulted. This improves the makespan characteristics in DHEFT as compared to that of HEFT.

When it comes to RHEFT, the performance is much better than other schemes. Utilization in RHEFT is more than both HEFT and DHEFT, but utilization is falling with higher resource availability. The falling trend in utilization is best compensated with reduced makespan characteristics of RHEFT. RHEFT exploits the resources to better utilization level than other schemes. That's why this work is named as Robust HEFT (RHEFT). Figs. 5(b) and 6(b) draws the standard error at CI = 95% for 50 and 100 tasks respectively. The Error data in Tables 4a and 4b gives better insight that utilization in RHEFT vary in smallest range among HEFT, DHEFT and RHEFT.

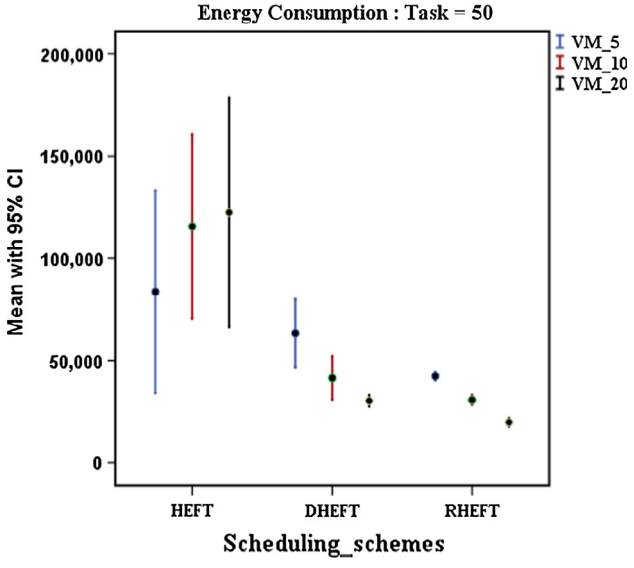
Figs. 7(a) and 8(a), plots the speed-up achieved as a result of parallel execution of tasks as compared to sequential execution of tasks. Better performance of RHEFT is due to consistent better utilization. RHEFT performed better than HEFT and DHEFT under both scenario i.e. VM = 5, VM = 10 and VM = 20. Figs. 7(b) and 8 (b) plots the standard error at CI = 95%. The range of variations in RHEFT is lowest when considered at 95% confidence Intervals. Also, considering higher values of average utilization in RHEFT, error range is acceptable. The error range of HEFT and DHEFT are poor at their low speed up levels. Error data is shown in Tables 5a and 5b for 50 and 100 tasks respectively.

Figs. 9(a) and 10(a) draws energy characteristics of this work. Energy consumption is based on {Eq. (6)}. The improved utilization and reduced makespan affects the energy consumptions. The nega-



	HEFT	DHEFT	RHEFT
VM_5	83482.38	63330.48	42424.81
VM_10	115523.88	41501.36	30748.21
VM_20	122428.17	30317.09	19778.68

(a)



(b)

Fig. 9. (a) Energy consumption characteristics of various scheduling/planning algorithms using 50 Tasks (b) Energy consumption error graphs of various scheduling/planning algorithms using 50 tasks (with C.I. = 95%)

tive slope of energy consumption in RHEFT is attributed to reduced makespan and consistently better utilization characteristics.

Error graphs in Figs. 9(b) and 10(b) plots the standard error at CI = 95%. Low variation in RHEFT as compared to HEFT and DHEFT justifies the robustness of RHEFT. This is why RHEFT is step forward towards Green cloud. The error tables in Tables 6a and 6b, justifies the claims.

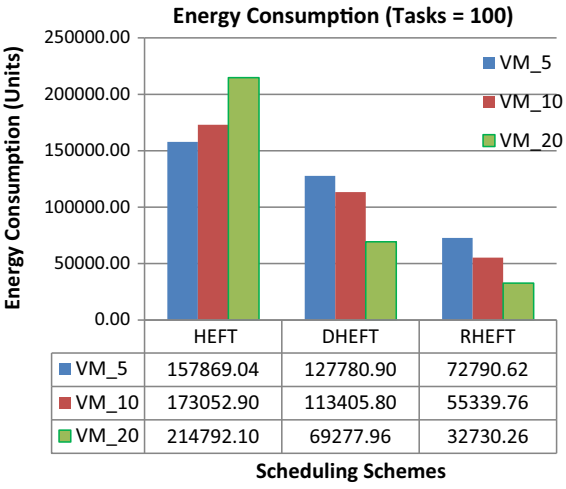


Fig. 10. (a) Energy consumption characteristics of various scheduling/planning algorithms using 100 tasks (b) Energy consumption error graphs of various scheduling/planning algorithms using 100 tasks (with C.I. = 95%)

Table 3a
Error Table for Makespan (95% CI) (Tasks = 50).

No. of VMs	Scheduling schemes	Mean	95% Confidence interval of mean	
			Lower bound	Upper bound
VM_5	HEFT	378.410	139.200	617.610
	DHEFT	292.730	208.240	377.220
	RHEFT	180.960	170.500	191.420
VM_10	HEFT	607.810	357.620	857.990
	DHEFT	206.270	147.910	264.640
	RHEFT	142.200	130.360	154.050
VM_20	HEFT	666.740	353.700	979.780
	DHEFT	159.700	144.580	174.810
	RHEFT	97.2108	720	107.700

Table 3b
Error Table for Makespan (95% CI) (Tasks = 100).

No. of VMs	Scheduling schemes	Mean	95% Confidence interval of mean	
			Lower bound	Upper bound
VM_5	HEFT	705.020	396.970	1013.070
	DHEFT	577.610	437.090	718.120
	RHEFT	303.830	295.320	312.330
VM_10	HEFT	866.520	319.750	1413.290
	DHEFT	567.500	495.490	639.500
	RHEFT	243.470	222.890	264.050
VM_20	HEFT	1156.440	334.370	1978.520
	DHEFT	361.620	295.830	427.4000
	RHEFT	151.850	144.660	159.040

Table 4a
Error table for utilization (95% CI) (Tasks = 50)

No. of VMs	Scheduling Schemes	Mean	95% Confidence interval of mean	
			Lower bound	Upper bound
VM_5	HEFT	66.6717	54.2043	79.1391
	DHEFT	56.3833	44.7462	68.0205
	RHEFT	79.3511	76.1298	82.5724
VM_10	HEFT	22.2612	13.9532	30.5692
	DHEFT	36.1435	30.0653	42.2217
	RHEFT	55.0611	49.2955	60.8268
VM_20	HEFT	13.2624	7.0856	19.4393
	DHEFT	19.8467	18.3853	21.3080
	RHEFT	37.9750	35.6656	40.2844

Table 4b
Error table for utilization (95% CI) (Tasks = 100).

No. of VMs	Scheduling schemes	Mean	95% Confidence Interval of Mean	
			Lower bound	Upper bound
VM_5	HEFT	68.9267	57.0728	80.7805
	DHEFT	62.4500	55.7452	69.1548
	RHEFT	86.1133	85.1471	87.0795
VM_10	HEFT	37.6900	27.1016	48.2784
	DHEFT	33.2900	30.5268	36.0532
	RHEFT	69.5767	66.2300	72.9233
VM_20	HEFT	16.6583	11.7654	21.5513
	DHEFT	22.3467	19.0820	25.6114
	RHEFT	54.0567	52.4988	55.6145

Table 5a
Error Table for Speedup (95% CI) (Tasks = 50).

No. of VMs	Scheduling schemes	Mean	95% Confidence interval of mean	
			Lower bound	Upper bound
VM_5	HEFT	1.8402	0.758	2.9223
	DHEFT	1.849	1.3284	2.3695
	RHEFT	2.8178	2.6552	2.9804
VM_10	HEFT	3.5957	3.2953	3.8962
	DHEFT	2.5922	1.9846	3.1997
	RHEFT	0.9882	0.4844	1.492
VM_20	HEFT	5.2757	4.7438	5.8075
	DHEFT	3.2074	2.8945	3.5204
	RHEFT	0.9869	0.2704	1.7033

Table 5b
Error table for speedup (95% CI) (Tasks = 100).

No. of VMs	Scheduling schemes	Mean	95% Confidence interval of mean	
			Lower bound	Upper bound
VM_5	HEFT	1.7667	1.0291	2.5042
	DHEFT	1.9417	1.5414	2.3420
	RHEFT	3.5550	3.4555	3.6545
VM_10	HEFT	1.5950	0.7791	2.4109
	DHEFT	1.9233	1.7055	2.1411
	RHEFT	4.4567	4.0773	4.8360
VM_20	HEFT	1.2633	0.5479	1.9787
	DHEFT	3.0633	2.4878	3.6388
	RHEFT	7.1200	6.7878	7.4522

Table 6a
Error table for energy consumption (CI = 95%) (Tasks = 50).

No. of VMs	Scheduling schemes	Mean	95% Confidence interval of mean	
			Lower bound	Upper bound
VM_5	HEFT	83482.3750	33925.5424	133039.2076
	DHEFT	63330.4750	46530.7385	80130.2115
	RHEFT	42424.8083	40260.6029	44589.0137
VM_10	HEFT	115523.8750	70293.2319	160754.5181
	DHEFT	41501.3583	30849.3341	52153.3825
	RHEFT	30748.2125	28268.3684	33228.0566
VM_20	HEFT	122428.1688	66237.3683	178618.9692
	DHEFT	30317.0854	27541.8024	33092.3684
	RHEFT	19778.6750	17671.8589	21885.4911

Table 6b
Error table for energy consumption (CI = 95%) (Tasks = 100).

No. of VMs	Scheduling schemes	Mean	95% Confidence interval of mean	
			Lower bound	Upper bound
VM_5	HEFT	157869.04170	94555.52180	221182.56150
	DHEFT	127780.85000	98814.82020	156746.87980
	RHEFT	72790.61670	70819.00140	74762.23200
VM_10	HEFT	173052.91830	71934.31160	274171.52510
	DHEFT	113405.76330	99901.86710	126909.65960
	RHEFT	55339.76000	50227.43340	60452.08660
VM_20	HEFT	214792.14000	66493.52670	363090.75330
	DHEFT	69277.96170	57129.58030	81426.34300
	RHEFT	32730.26170	31171.73010	34288.79320

6. Conclusion and future discussion

A hybrid planning algorithm, RHEFT is presented in this work. RHEFT is hybrid of HEFT and a novel scheduling algorithm for independent tasks called Interior Scheduling. In RHEFT, a conversion of sequential ranked tasks into set of independent ranked tasks is performed and better approximates the set of tasks in the ready queue. Using IS scheduling algorithm a HEFT planning algorithm from global allocation got evolved into RHEFT planning algorithm resulted with sub-local allocation. This evolution of HEFT to RHEFT exhibits consistent utilization across different availability levels of resources. The resultant algorithm is thus named Robust HEFT. The error analysis presented at CI = 95%, justifies the nomenclature of Robust HEFT (RHEFT). The extension of RHEFT which approximates ready queue even better than RHEFT is future scope of this work.

References

[1] Ding Y, Qin X, Liu L, Wang T. Energy efficient scheduling of virtual machines in cloud with deadline constraint. Future Gener Comput Syst 2015.
[2] Kliazovich D, Pecero JE, Tchernykh A, Bouvry P, Khan SU, Zomaya AY. CA-DAG: communication aware directed acyclic graphs for modeling cloud computing applications. In: IEEE 6th international conference on cloud computing. p. 277–84.
[3] Smanchat S, Viriyapant K. Taxonomies of workflow scheduling problem and techniques in the cloud. Future Gener Comput Syst 2015;52:1–12.
[4] Arabnejad H, Barbosa JG, Prodan R. Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources. Future Gener Comput Syst 2016;55:29–40.
[5] Jena RK. Multi objective task scheduling in cloud environment using nested PSO framework; 2015.
[6] Lee YC, Han H, Zomaya AY, Yousif M. Resource-efficient workflow scheduling in clouds. Knowledge-Based Syst 2015;80:153–62.
[7] Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 2002;13(3):260–74.
[8] Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K. Characterizing and profiling scientific workflows. Future Gener Comput Syst 2013;29 (3):682–92.
[9] Juve G, Deelman E. Scientific workflows in the cloud. In: Grids, clouds and virtualization. London: Springer; 2011. p. 71–91.
[10] Lee K, Paton NW, Sakellariou R, Deelman E, Fernandes AA, Mehta G. Adaptive workflow processing and execution in pegasus. Concurrency Comput: Pract Exp 2009;21(16):1965–81.
[11] Rahman M, Hassan R, Ranjan R, Buyya R. Adaptive workflow scheduling for dynamic grid and cloud computing environment. Concurr Comput: Pract Exp 2013;25(13):1816–42.
[12] Abrishami S, Naghibzadeh M, Epema DH. Deadline-constrained workflow scheduling algorithms for Infrastructure as a service clouds. Future Gener Comput Syst 2013;29(1):158–69.
[13] Abdelkader DM, Omara F. Dynamic task scheduling algorithm with load balancing for heterogeneous computing system, Egypt. Inform J 2012;13 (2):135–45.
[14] Bittencourt LF, Madeira ERM. HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. J Internet Services Appl 2011;2 (3):207–27.
[15] Sakellariou R, Henan Z. A hybrid heuristic for DAG scheduling on heterogeneous systems. In: Parallel and distributed processing symposium, proceedings of 18th international. IEEE; 2004.
[16] Sih GC, Lee EA. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architecture. IEEE Trans Parallel Distrib Syst 1993;4(2):175–87.
[17] Chen H, Wang F, Helian N, Akanmu G. User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. In: National conference on parallel computing technologies (PARCOMPTECH). IEEE; 2013. p. 1–8.
[18] Li Jiayin et al. Online optimization for scheduling preemptable tasks on IaaS cloud systems. J Parallel Distrib Comput 2012;72(5):666–77.
[19] Yu J, Buyya R, Ramamohanarao K. Workflow scheduling algorithms for grid computing. In: Metaheuristics for scheduling in distributed computing environments. Berlin, Heidelberg: Springer; 2008. p. 173–214.
[20] Chen W, Deelman E. Workflowsim: a toolkit for simulating scientific workflows in distributed environments. In: E-science (E-Science), 2012 IEEE 8th international conference on. IEEE; 2012. p. 1–8.
[21] Bajaj R, Agrawal DP. Improving scheduling of tasks in a heterogeneous environment. IEEE Trans Parallel Distributed Syst 2004;15:107–18.
[22] Fahringer T, Jugravu A, Pllana S, Prodan R, Seragiotto C, Truong HL. ASKALON: a tool set for cluster and Grid computing. Concurr Comput: Pract Exp 2005;17 (2):143–69.
[23] Blaha P, Schwarz K, Madsen GKH, Kvasnicka D, Luitz J. wien2k. An augmented plane wave+ local orbitals program for calculating crystal properties; 2001.
[24] Rutschmann P, Theiner D. An inverse modelling approach for the estimation of hydrological model parameters. J Hydroinform 2005.