Proceedings of the Second International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2020)
IEEE Xplore Part Number: CFP20K58-ART; ISBN: 978-1-7281-4167-1

Efficient Resource Allocation with Score for Reliable Task Scheduling in Cloud Computing Systems

Dr. Vijayalakshmi A. Lepakshi[1]
Department of Computer Science
Maharani Lakshmi Ammanni College for Women
Bangalore, India

Dr. Prashanth C S R[2]
Department of Computer Science and Engineering,
New Horizon College of Engineering
Bangalore, India

***Abstract*—Parallel task execution in a heterogeneous cloud computing system emerges as NP-complete problem and in the literature, many heuristics behave differently when deployed in various environments. Efficient resource allocation improves reliability and leads to the completion of jobs and minimization of delays. In general, static task scheduling algorithms consider the earliest finish time (EFT) of the task to minimize the makespan. In this work, we propose a new heuristic called Efficient Resource Allocation with Score (ERAS) for reliable task scheduling in cloud computing systems, which considers temporal operational availability of Virtual Machines (VM) by considering various types of delays and EFT to assign a normalized score to the processor for scheduling tasks. Here, the allocation of VMs to tasks is based on a score given to each VM, by considering multiple criteria. The results show that ERAS algorithm gives better performance with increased reliability when compared to existing algorithms that consider only EFT for allocation.**

Keywords - Cloud computing; Heterogeneity; Reliability; Score; Task Scheduling; Virtual Machines

I. INTRODUCTION

IaaS Clouds provide on-demand, flexible, scalable infrastructure on a pay-as-you-use basis. These virtualized resources are leased with predefined CPU, storage, memory, and bandwidth capacity that suit a range of application needs [1].

A cloud computing environment provides services to the users using unified computing resources in the form of Virtual Machines (VMs) that share resources such as CPU, Storage, Memory, and Bandwidth on a host and multiple VMs can share the bandwidth by using network virtualization of a cloud [2], [13]. Heterogeneity involved in the cloud environment increases the complexity of task scheduling since the efficiency of task scheduling gets affected by various system parameters. In general, scheduling algorithms consider the performance of the VMs and allocate tasks such that makespan is reduced. But the task execution may be delayed due to the non-availability of resources such as VMs due to various reasons such as administrator's policies, failure of physical resources, network failure or increased load on the underlying resources [5]. Virtual Machine's availability is the percentage of the machine's computational resources for the given application. The VM is said to be less available i.e., less than 100% when the load on the virtual machine increases. Thus, scheduling algorithms address various challenges [6] derived from the characteristics of the cloud depending on various models. The application model consists of a single workflow, workflow ensembles or multiple workflows. The scheduling model comprises static, dynamic or hybrid scheduling [6]. Resource Provisioning [12] is from a static VM pool or an elastic VM pool. The resource model has a provider, storage network and virtual machine. Further VM leasing may be limited VMs or unlimited VMs, single VM or multiple VM types. The VM pricing model consists of dynamic pricing, static pricing, subscription-based or time unit-based pricing. The objectives of scheduling algorithms in the cloud environment are based on cost optimization by limiting the budget, minimization of makespan, meeting the deadline, workload maximization, VM utilization and maximization, energy consumption minimization, and developing awareness on their reliability and security.

Further, the user's QoS requirements are considered for selecting VMs instances, availability and VM types based on SLAs. Therefore, in this research work, multiple criteria such as the probability of the operational availability of VM and performance of VM equally to schedule tasks for a single workflow with an objective of high reliability and makespan minimization is considered.

The research work has been organized as follows: the problem statement is defined in Section II. Related work is presented in Section III. The proposed algorithm of ERAS is presented in Section IV. The results of the proposed algorithm ERAS will be presented in Section V. Finally, Section VI concludes the proposed research work.

II. PROBLEM STATEMENT

The primary research objective is to allocate resources such as virtual machines[VM] efficiently to tasks with a score based approach by considering multiple factors like EFT and operational availability of resources while minimizing the makespan for ensuring the precedence constraints with increased reliability.

A parallel application represented as DAG G = (T, E) is composed of T = {t_1, t_2, ….,t_n} sub-tasks and E={e_1, e_2,….,e_m} edges. The nodes in G represent sub-tasks. The edges e_{ij} between the nodes t_i and t_j represent the communication cost. Node weights represent the average computation cost of subtask among all available virtual machines. The sub-tasks are arranged in the various levels while satisfying the precedence constraints. A sub-task is executed upon satisfying the precedence constraints. During execution when a task is ready, the task scheduler places it in the ready queue of the allotted Virtual Machine. Depending on the local scheduling policy of the data center, tasks are executed along with the other jobs. Though the performance of the processor is high, delays may occur, since task execution takes place depending on the load of the underlying physical resource on which, virtual machine (VM) is deployed and its operational availability. The proposed algorithm considers operational availability and EFT in scheduling decisions hence it is reliable.

An example sample DAG is given in Fig. 1 below:

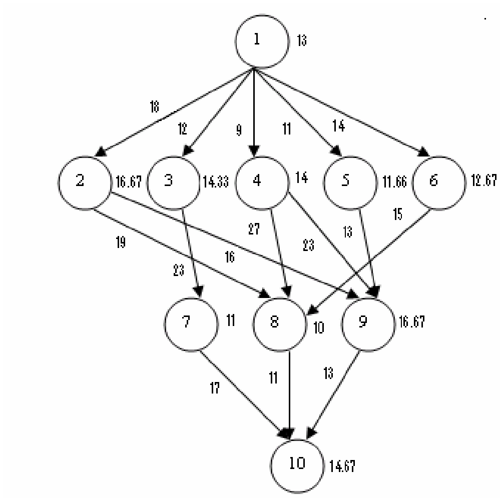

Fig. 1. A Sample Directed Acyclic Graph (G1)

III. RELATED WORK

We studied various task scheduling algorithms existing in the literature. Scheduling algorithms are categorized as static and dynamic. In general, static scheduling algorithms consider a parallel application modeled as DAG one at a time for scheduling in the heterogeneous environments whereas dynamic scheduling algorithms consider multiple independent DAGs for scheduling. In this work, we consider a single DAG for static scheduling with static VM provisioning.

In the cloud environment, resource provisioning is based on the Service Level Agreements. VMs are provided either statically or dynamically to meet the QoS requirements of users. In static VM deployment, VM migrations are not allowed and allocated resources will not be changed until the number of CPU cycles (as in SLA) is used for its execution [14]. There is a disadvantage with this scheme i.e., to meet the QoS requirements as in SLA, VMs are over-provisioned. To avoid overprovisioning dynamic allocation of VMs can be adopted. But in this research work, the limiting of static resource allocation technique and static scheduling considering reliability. IaaS cloud provides VMs on lease either homogeneous or heterogeneous depending on the user's request. These leases are based on the request placed by the user at the time of job submission with the required number of VMs, VM type with a requested CPU, storage and memory and bandwidth capacity. VM types are available at various prices that suit a range of application types such that user can choose VMs according to budget and deadline. Moreover, selection of heterogeneous VMs for task execution improves resource utilization.

Availability [8], [15] is an important factor used to assess the performance of systems that accounts for the reliability of the system. Availability of a system is classified in various types, including Instantaneous Availability – probability that a system will be operational at a specific time t; average uptime availability or mean availability - proportion of time available during a time period; steady-state availability - long term availability once the system becomes operational; inherent availability – is the steady state availability while considering only corrective maintenance downtime of the system; Achieved Availability – considers corrective maintenance and preventive maintenance downtimes and operational availability – actual average availability over a period of time that includes all sources of downtime, such as corrective maintenance downtime, preventive maintenance downtime, administrative downtime, logistic downtime as well as the time required to provision and de-provision the system. For VMs, the provisioning delay ranges from 30 seconds to 800 seconds and de-provisioning delay may vary up to as low as 3 seconds. Also, VM performance is degraded by at most 24% based on a normal distribution with a 12% mean and a 10% standard deviation. In addition to this, a network link's total available bandwidth is shared between all the transfers using that link results some amount of delays. All these kinds of delays cannot be neglected and must be taken into account. In the dynamic cloud scenario, the availability that the cloud user experiences is the operational availability. It is a posteriori availability based on the actual events that occur to the system. Thus considering operational availability, to allocate resources increases the reliability of task execution in a cloud environment. We use this operational availability in our work to allocate VMs to tasks. A cloud user requesting the highest level of operational availability will be charged more.

In the literature, multi-criteria decision making (MCDM) is used in many fields for decision making in the presence of multiple conflicting criteria [7]. There exist different types of normalization techniques for ranking that can be adapted to rank the criteria for decision making. In general, normalization is used to transform different criteria into a compatible measurement. In the literature, many MCDM techniques are used in a cloud computing environment for selection problems. The authors in [16] used an MCDM technique known as Analytic Hierarchy Process (AHP) in the SMICloud framework to rank the cloud services provided by different cloud providers.

We studied various static scheduling algorithms [10] intended for heterogeneous environments in the literature. A static task scheduling algorithm Heterogeneous Earliest Finish Time (HEFT) [9] for a bounded number of processors assumes all the resources are 100% available and gives the schedule to minimize the makespan. In HEFT, tasks in a parallel application are prioritized and are executed by allocating the best processor which gives the minimum Earliest Finish Time among all available processors. ECTS [11] for a bounded number of heterogeneous processors execute parallel applications assuming 100% availability of resources. In ECTS, tasks are prioritized level wise based on Expected Completion time. In HEFT and ECTS algorithms processor selection is done based on single criteria EFT. In this research work, cloud characteristics, static virtual resource provisioning and its operational availability for scheduling a parallel application modeled by Directed Acyclic Graph, in a heterogeneous cloud environment are considered.

ALGORITHM :
Begin ERAS
// VM represents set of Virtual Machines
// T= $\{t_1, t_2, \ldots, t_n\}$ represents set of Task nodes
//RA(t_i,vm_j) is the operational availability probability of vm_i for task t_i

For all t_i **in** T
Compute $rank_u(t_i)$
End For
RTL ← Starting Node
While *RTL* ***is*** *NOT NULL* **do**
t_i ← Node with maximum $rank_u$ in the *RTL*
For each vm_j **in** *VM*
$EST(t_i, vm_j) = \max(TA_{[j]}, \max_{n_m \in pred(n_i)}(EFT(t_m, vm_k) + c_{m,i}))$
$EFT(t_i, vm_j) = w_{i,j} + EST(t_i, vm_j)$
End For
For all vm_j **in** VM
Find a normalized score VM_{avail} for every virtual machine for each task using virtual machine operational availability
Find a normalized score VM_{eft} for each virtual machine using its eft
Find the total score of the virtual machine considering both criteria
Score = VM_{avail} + VM_{eft}

End For
Allocate task node t_i on virtual machine vm_j which provides its least *Score*
Update *TA[vm_j]* and *RTL*
End While
End ERAS

Fig. 2 ERAS algorithm

IV. EFFICIENT RESOURCE ALLOCATION WITH SCORE FOR RELIABLE TASK SCHEDULING IN CLOUD COMPUTING SYSTEMS (ERAS):

Our ERAS algorithm is intended for a bounded number of statically provisioned VMs. This is designed for the brokering component that schedules applications in the cloud environment. ERAS algorithm begins with predicting the VMs to be allocated based on the users request and VMs operational availability.

Stage 1: In the first stage, it finds the priority of the task nodes based on expected completion time level wise.

Stage 2: At the second stage, for each task in the priority list Earliest Finish Time is calculated for all the available VMs. At this stage we consider multiple criteria for the selection of the VM.

Step 1: We find a normalized score for each virtual machine based on the first criteria, virtual machine probability of operational availability. We use a linear dimensionless normalization technique to find the score VM_{avail}. We make certain assumptions to calculate the VM_{avail}. We define the following:

$VM_{prefavail}$ = 1.0 i.e 100% operational availability

$VM_{lowavail}$ =0.3 i.e 30% of operational availability

$$VM_{avail} = \frac{(VMprefavail - VMprobavail)}{(VMprefavail - VMlowavail)} * 10 \qquad (1)$$

This normalization in (1) generates a score VM_{avail} ranging from 0 to 10 for each VM where 0 indicates a perfect match for the criteria operational availability.

Step 2: Similarly, we find a normalized score for each virtual machine based on the second criteria, EFT of the task on each VM. We use a linear min-max dimensionless normalization technique to find the score VM_{eft}. To find this normalized score we make some assumptions as follows:

$VM_{prefeft}$ = min {EFT out of all VMs}

VM_{loweft} = max {EFT out of all VMs}

$$VM_{eft} = \frac{(VMprefeft - VMieft)}{(VMprefeft - VMloweft)} * 10 \qquad (2)$$

This normalization in (2) generates a score VM_{eft} ranging from 0 to 10 for each VM where 0 indicates a perfect match for the criteria EFT.

Step 3: Now we calculate a total score for each VM considering both criteria equally using (3) as follows:

$$VM_{score} = VM_{avail} + VM_{eft} \qquad (3)$$

Step 4: Now we map each task node to a VM based on the VM_{score} . A VM with least score is selected for executing a task.

V. EXPERIMENTAL RESULTS

In this section, we discuss the results of our experiment through simulation. Comparative results on the developed ERAS algorithm with other static scheduling algorithms such as HEFT and ECTS. ERAS algorithm is tested with an example graph with 10 nodes given in Fig. (1) with a randomly generated probability of operational availability for VMs. In addition to this, ERAS algorithm using randomly generated DAGs of various sizes and an application graph Gauss graph is examined. In this section, we discuss performance metrics used to compare ERAS with HEFT and ECTS, generation of random graphs and experimental results of ERAS algorithm.

A. Comparison Metrics[1]

The ERAS algorithm is compared with other existing algorithms for the heterogeneous environment using following metrics:

Makespan is Schedule Length of its output schedule and the main performance measure.

Schedule Length Ratio of an algorithm is defined as follows in (4):

$$SLR = \frac{\text{Makespan}}{\sum_{n_I \in CP_{min}} \min_{p_J \in Q}\{w_{ij}\}} \quad (4)$$

Speedup is computed as a ratio of sequential execution time to the parallel execution time (makespan).

B. Random DAG Generator:

Random directed acyclic graphs are generated using an algorithm for random graph generator given in [4]. This algorithm takes number of nodes as input and generates a weighted DAG with randomly generated edges. We generated random graphs of various sizes with one starting node and one exit node. Heterogeneity factor [9] for VMs processors speeds depends basically on the varied computation costs on processors i.e.,

$$\eta = \{0.1,\ 0.5,\ 1.0\}.$$

The average computation cost i.e W_i of each task t_i is randomly generated from a uniform distribution with range $[0, 2*W_{dag}]$. The average computation cost of the given graph W_{dag} is fixed randomly in the graph based on the graph size. The computation cost of each task t_i on VM_j is computed randomly from the following range:

$$W_i * (1 - \eta/2) <= W_{ij} <= W_i * (1 + \eta/2)$$

To implement ERAS algorithm we use randomly generated DAGs of various sizes and probability of operational availability of VMs as input to generate output schedules. We also implemented HEFT and ECTS algorithms for the same set of graphs and we also estimated the delay that results for that schedule based on the operational availability in the heterogeneous cloud environment.

C. Results:

C. Results: Experimental results are organized in three test suites as follows: We studied the performance of ERAS algorithm within the simulated environment using randomly generated graphs of various sizes. Various Application Graphs such as Montage, Epigenomics, SIPHT, and CyberShake are executed in the simulated CLOUDSIM environment. The experimental results are shown in test suite 2.

Test Suite 1: In test suite one, randomly generated DAGs of sizes 10,20,30,40 and 50 nodes are considered. Our ERAS algorithm is executed for these random DAGs. Each random DAG generated is executed 200 times with different probability of operational availability of VM and performance of the ERAS algorithm is compared in terms of different graph sizes. The same sets of random graphs are executed for HEFT, ECTS algorithm and probable delay for HEFT and ECTS are evaluated in the dynamic cloud environment. Results show that ERAS algorithm is more reliable than HEFT and ECTS. Comparative results are shown below in Fig. 3, Fig. 4 and Fig.5:

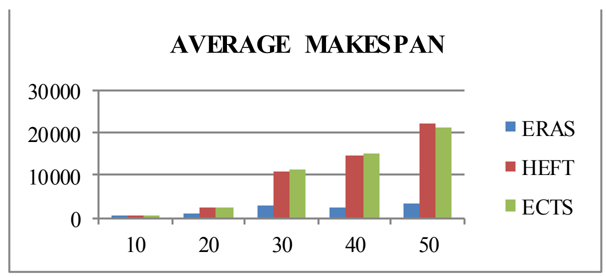

Fig. 3. Average makespan of various random graphs

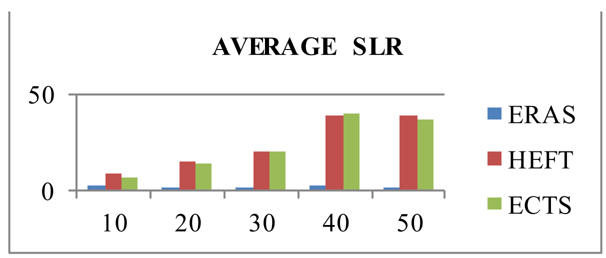

Fig. 4. Average SLR of various random graphs

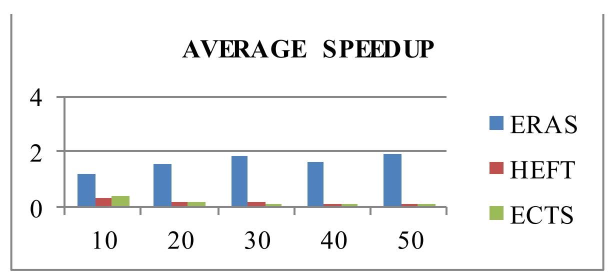

Fig. 5. Average speedup of various random graphs

Test Suite 2: In test suite two, an application graph with random computation costs of real-world problems such as Gaussian Elimination graph [9] is considered. We considered a matrix size (m=5) for our experiment with 14 nodes. The performance of ERAS is evaluated with probability of operational availability of VMs. and the probable delay that occurs for HEFT and ECTS schedule by considering the operational availability factor.

The comparative results are shown below in Fig. 6, Fig. 7 and Fig. 8.

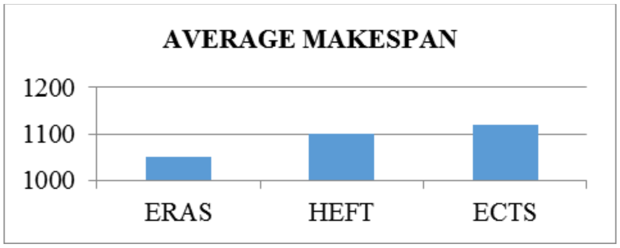

Fig. 6. Average Makespan of Gauss graph

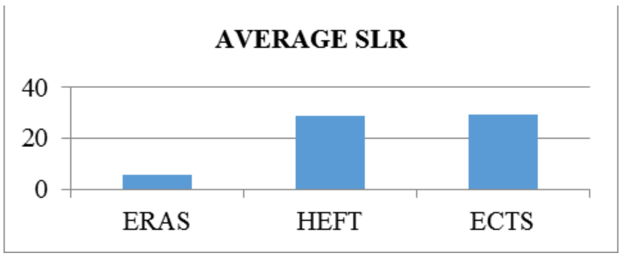

Fig. 7. Average SLR of Gauss graph

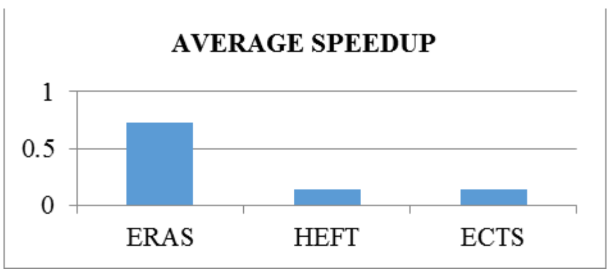

Fig. 8. Average Speedup of Gauss graph

Simulation Results of Application Graphs on WORKFLOWSIM:

The performance of ERAS is studied by executing a sample graph given in the Fig. 1 in the CLOUDSIM environment. Comparative results with HEFT are given in Fig. 9.

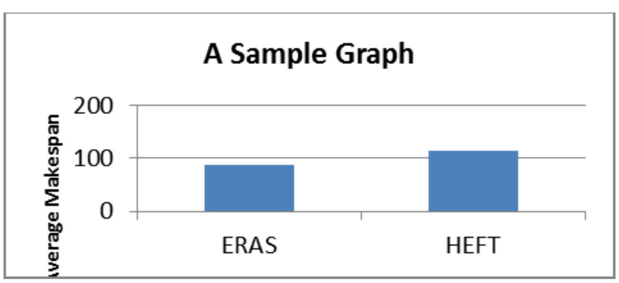

Fig. 9. Average Makespan of Sample DAG (G1)

Many scientific applications are modeled as workflows that can be processed efficiently in distributed cloud environments. In this research work, application graphs such as Montage, Epigenomics, SIPHT and CyberShake applications [6] are considered. The Montage application workflow is an I/O intensive astronomy application that is used to create custom montages of the sky based on input images. In the bioinformatics field, the CPU intensive Epigenomics workflow is used to automate the execution of various genome sequencing operations. CyberShake is an earthquake analysis application and SIPHT is a scientific application graph from the field of Bioinformatics used to automate the process of gene encoding. ERAS algorithm's performance is studied by executing these application graphs in the CLOUDSIM environment.

The overall performance improvement of ERAS algorithm is about 26% better than HEFT for an I/O intensive application graph Montage, 40% better than HEFT for a CPU intensive application graph Epigenomics, 23% better than HEFT for scientific application graph SIPHT, 27% better than HEFT for a CyberShake earthquake analysis workflow.

The performance of ERAS and comparative results with HEFT and ECTS of SIPHT, Epigenomics, Cybershake, and Montage application graph are shown below Fig. 10, Fig. 11, Fig. 12 and Fig.13 respectively.

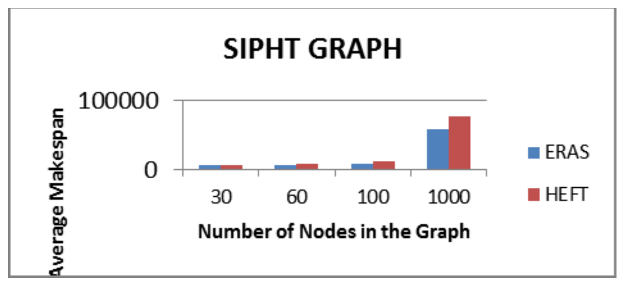

Fig. 10. Average Makespan of SIPHT Workflow

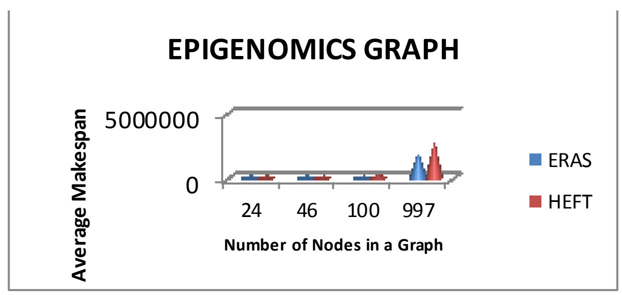

Fig. 11. Average Makespan of Epigenomics Workflow

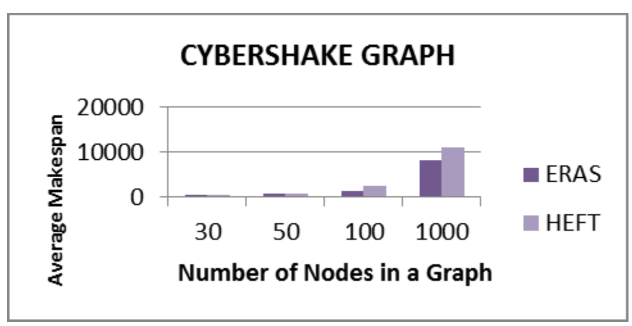

Fig. 12. Average Makespan of CyberShake Workflow

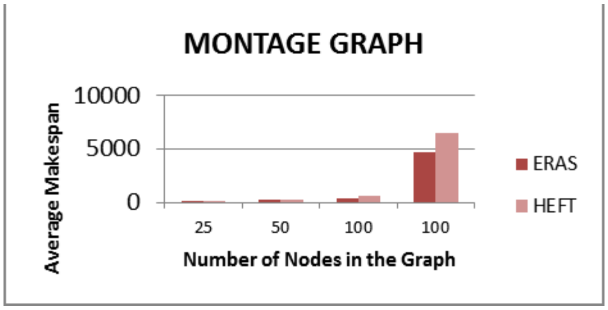

Fig. 13. Average Makespan of MONTAGE Workflow

Test Suite 3: In test suite three, we considered a random graph with 50 nodes. Performance of our algorithm ERAS is evaluated in terms of parallelism for various numbers of VMs such as 9, 12 and 15. Experimental results in Fig. 14 show that with an increase in the number of processors the makespan is minimized with increased reliability without causing any delays:

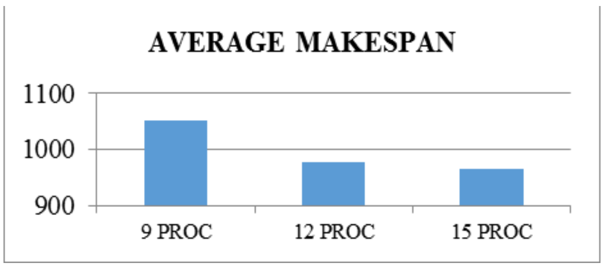

Fig. 14. Average makespan of 50 nodes random graph with varying number of Virtual Machines

VI. CONCLUSION

In this research work, a new heuristic called Efficient Resource Allocation with Score for reliable Task Scheduling in Cloud Computing Systems for a bounded number of heterogeneous virtual machines that work in two phases is proposed. In the first phase, tasks are prioritized based on the upward rank. In the second phase, a normalized score is calculated for VM considering the two criteria, probability of operational availability and EFT. Tasks are scheduled on a virtual machine with the least score among all VMs. ERAS algorithm is proven to be reliable for scheduling parallel applications structured as DAGs, on to a heterogeneous cloud environment where VMs availability is uncertain and shared by various users since it considers VM availability and EFT in scheduling decisions. By extensive performance evaluation, we demonstrate that the ERAS algorithm produces a reliable schedule with minimized makespan. Our ERAS algorithm is reasonably well and simple to implement in cloud environments. Our ERAS algorithm produces a similar schedule as HEFT for 100% operational availability. The performance improvement of ERAS is about 22% better than HEFT for a sample graph given in Fig. 1.

The overall performance improvement of ERAS algorithm is about 26% better than HEFT for an I/O intensive application graph Montage, 40% better than HEFT for a CPU intensive application graph Epigenomics, 23% better than HEFT for scientific application graph SIPHT and 27% better than HEFT for a CyberShake earthquake analysis workflow.

The simulation results show that the ERAS algorithm is more reliable for a dynamic cloud environment where VMs are highly uncertain in the cloud environment and available with varying operational availability for scheduling.

REFERENCES

[1] Rajkumar B., C.S. Yeo, Venugopal S., J. Broberg, I. Brandic, " Cloud Computing and Emerging IT platforms : Vision, Hype, and Reality for Delivering Computing as the 5th Utility ", Journal of Future Generation Computer Systems, Vol. 25, No. 6, pp. 599–616. (2009)

[2] Xu Wang, Beizhan Wang, Jing Huang, "Cloud Computing and its key techniques", 2011, 978-1-4244-8728-8, IEEE

[3] Gerasoulis A, and Yang. T, "A comparison of clustering heuristics for scheduling directed graphs multiprocessors" Journal of Parallel and Distributed Computing, 16(4):276-291.

[4] Yinfeng Wang, Z. Liu, W Yan, "Algorithms for Random Adjacency Matrixes Generation Used for Scheduling Algorithms Test", 2010 International Conference on Machine Vision and Human-Machine Interface (MVHI)

[5] Brent Rood and Michael J. Lewis," Multi-State Grid Resource Availability Characterization", IEEE 8th Grid Computing Conference, 2007

[6] Maria A. Rodriguez and Rajkumar Buyya, "A Taxonomy and Survey on Scheduling Algorithms for Scientific Workflows in IaaS Cloud Computing Environments", Concurrency and Computation", Practice and Experience (CCPE), ISSN: 1532-0626, Wiley Press, New York, USA, 2016

[7] Rohan K. Gavade, "Multi Criteria Decision Making: An overview of different selection problems and methods", IJCIT, Vol.5(4), 2014, ISSN: 0975-9646

[8] G. Schmidt, "Scheduling with Limited Machine Availability" European J. Operational Research, vol. 121, pp. 1-15, 2000.

[9] Haluk Topcuoglu, S. Hariri, Min-You Wu "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing", 1045-9219/02, 2002 IEEE

[10] Raj, Jennifer S., and S. Smys. "Virtual Structure for Sustainable Wireless Networks in Cloud Services and Enterprise Information System." *Journal of ISMAC 1*, no. 03 (2019): 188-205.

[11] R.Eswari and S.Nickolas : "Expected Completion Time based Scheduling Algorithm for Heterogeneous Processors", 2011, International Conference on Information Communication and Management IPCSIT vol.16, IACSIT Press, Singapore

[12] M. A. Salehi, B. Javadi, and Rajkumar B., Resource Provisioning based on Preempting Virtual Machines in Distributed Systems; Concurrency and Computation: Practice and Experience, Vol.26, No. 2, Pages: 412-433, ISSN: 1532-0626, Wiley Press, New York, USA, February 2014

[13] Abdul Razaque, Nikhileswara Reddy V, Nisargakumar Soni, Gunasree Janapati, K. Reddy V, "Task Scheduling in Cloud Computing", In Long Island Systems, Applications and Technology Conference (LISAT), 2016 IEEE, pp. 1-5. IEEE, 2016.

[14] S K Garg, Srinivasa K G, and Rajkumar B, " SLA - based Resource Provisioning for Heterogeneous Workloads in a Virtualized Cloud Datacenter", Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), LNCS 7016, Melbourne, Australia, October 24-26, 2011

[15] Shadaram, Mehdi, S. Smys, and Sherali Zeadally. "Introduction to the special issue on "Wireless systems: New technologies, resource optimization and security"." *Computers and Electrical Engineering 2*, no. 40 (2014): 289-290.

[16] S K Garg, Steve Versteeg, and Rajkumar B, "A Framework for Ranking of Cloud Computing Services ", June 2013, Future

Generation Computer Systems, Vol. 29, No. 4, Pg: 1012-1023, ISSN: 0167-739X, Elsevier Science, Amsterdam, The Netherlands