DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4214

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

CLOUD WORKFLOW SCHEDULING BASED ON STANDARD DEVIATION OF PREDICTIVE RESOURCE AVAILABILITY

Vijayalakshmi A. Lepakshi
Research Scholar
Dept. of Computer Science and Engineering
New Horizon College of Engineering
Bangalore, India

Dr. Prashanrh C S R
Professor and Head
Dept. of Computer Science and Engineering
New Horizon College of Engineering
Bangalore, India

Abstract: Cloud computing has emerged as a new paradigm that provides services to the user on pay-as-you-use basis. Cloud computing's Infrastructure as a service provides various resources to user such as computing, storage and bandwidth etc., where Computing services can be provided in the form of Virtual Machines. Parallel applications submitted for execution on these Virtual Machines can be completed without any delays only when the Virtual Machines are available for execution. In general, resources in cloud are shared by various users and their availability is dynamic and unpredictable due to various reasons such as capacity of underlying hardware and number of users sharing the resources and various other reasons. Hence, non-availability of these allocated resources may cause delays in completion of execution of parallel applications. In this scenario, existing algorithms without considering the delays that may occur due to non-availability of Virtual Machines may not perform better and are less reliable in terms of completion of jobs. In this paper, we propose a new heuristic called Workflow Scheduling based on Standard Deviation of Predictive Resource Availability in cloud computing considers the dynamic nature of cloud resources and its dynamic availability in scheduling decisions and produces reliable schedules..

Keywords: Cloud Computing, Resource Availability, Reliability, Task Scheduling, Virtual Machines

I. INTRODUCTION

Cloud computing [1] has emerged as a new paradigm that provides computing as utility like water, electricity, gas and telephony etc., on pay-as-you-use basis. Cloud computing also provides storage and networking resources to users along with computing as a service. Cloud providers provide computing resources in the form of Virtual Machines (VM's) to its consumers based on Service Level Agreements. The advanced microprocessor technologies and advanced software technologies have increased the ability of commodity hardware such that many scientific applications and high performance applications can run on these Virtual Machines efficiently. These Virtual Machines isolate applications running on them from underlying hardware as well as other VM's. Thus enterprises and individual users can outsource their application execution to cloud resources while reducing the setup and maintenance costs of infrastructures of their own.

Provisioning Resources [2] to support heterogeneous applications is challenging due to dynamic nature of Cloud computing. Even though applications can run on cloud resources efficiently on shared cloud data centre infrastructures concurrently, cloud providers do not provide performance guarantee. Moreover, due to different types of High Performance applications and web applications executed on cloud with various Quality of Service (QoS) requirements, resource provisioning is much harder.

Resources are highly dynamic due to various reasons such as sharing of underlying hardware, interconnected network, load on the servers; number of users, Service Level Agreements between users and providers and administrator policies of data centre, etc., which further makes the availability unpredictable. The authors in [3] characterize the availability of resources in grid environment in different states ranging from non-availability to availability based on the transitions it takes. Thus considering availability of these resources in scheduling decisions improves reliability and reduces unpredicted delays in execution of jobs.

Task scheduling algorithms [4] that consider availability of resources in scheduling decisions improve resource utilization while producing the reliable schedules to avoid unpredicted delays in cloud environments where the idle time of resources can be utilized by other applications. Hence there is a need for effective task scheduling algorithms in cloud computing. Task scheduling can be accomplished either at compile-time or run time. When the user provides application characteristics such as task execution times, data dependencies between the tasks in advance, then scheduling can be accomplished with static task scheduling model.

The rest of the paper is organized as follows: we define problem statement and resource model in section 2. Related work is discussed in Section 3. In Section 4, we introduce a new heuristic SDPRA. Experimental results and comparative results are presented in Section 5. Conclusion of present research is given in Section 6.

II. PROBLEM STATEMENT AND RESOURCE MODEL

In this work the main objective of the scheduling algorithm is to produce reliable schedule that considers the availability of Virtual Machines in scheduling decisions. Various factors such as the physical characteristics of underlying hardware, number of users sharing the resources, load on the host, and failures of the physical devices, network failures and time that takes to recover from failures, account for availability of Virtual Machines. Apart from all these factors, when Virtual Machines are allocated to tasks for execution there may be provisioning and de-provisioning

delays which contribute to overall delays in completion of tasks. In this work the availability of Virtual Machines are predicted for certain time interval based on the historical data available in the cloud environment by the schedulers. We predict the availability of VM for each and every time unit in the predicted time interval. Each time slot is considered as one time unit required for execution. During scheduling the availability of VM is considered for every time slot used on that machine and probable predicted delays are incorporated and decision will be taken.

In this work we consider scientific applications modelled as Directed Acyclic Graphs, i.e., a graph without cycles, directed edges and dependencies among the nodes. Each node represents a task and edges between the tasks represent communication or data transfer costs. A workflow W consists of a set of tasks T={t1, t2,… tn}and a set of edges E. An edge e_{ij} = (t_i, t_j) exists in DAG if there exists data dependency between t_i and t_j. The task t_i is called as a parent task whereas t_j is known as child task. The child task cannot run until all its parent tasks have executed and the data is transferred to child task. A node without parent node is considered as a start node and a node without children is considered as exit node. The overall execution time required by the DAG is considered as the required to execute the start node to until it completes the execution of exit node.

The problem can be formally stated as "Scheduling parallel application modelled as Directed Acyclic Graph on to a heterogeneous cloud computing environment, in which resources such as Virtual Machines are provisioned for execution with unpredictable availability, to minimize the makespan while producing reliable schedules."

An example DAG is given below:

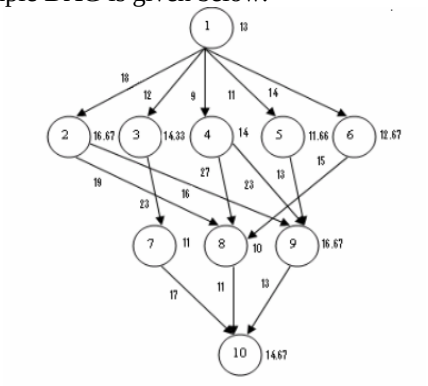

Figure 1 A Sample DAG

Table 1 Computation Cost Matrix

Task	vm_1	vm_2	vm_3
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

III. RELATED WORK

An extensive study has been done related to task scheduling algorithms, resource provisioning and dynamic nature of availability of resources in the distributed cloud environment. In general, task scheduling algorithms are designed to meet various objectives. The authors in [5] characterize the objectives of scheduling algorithms as cost minimization, makespan minimization, workload maximization, VM utilization maximization, energy consumption minimization, reliability aware and security aware.

The authors in [6] propose an algorithm; Heterogeneous Earliest Finish Time (HEFT) is an application scheduling algorithm for bounded number of heterogeneous processors that considers the resources are 100% available for allocation. HEFT algorithm works in two phases: n first phase, the tasks are prioritized based on the upward rank of tasks and during second phase, tasks are selected in the order of their priority and the task is assigned to the processor, which minimises its earliest finish time using an insertion based policy.

The authors in [7] propose Expected Completion Time based Scheduling (ECTS) algorithm for bounded number of processors that considers resources are 100% available for allocation. ECTS algorithm works in two phases; during first phase tasks are prioritized level wise based on Expected Completion Time (ECT). The ECT is calculated based on Average Computation Cost of tasks and maximum data arrival cost. During the second phase, a processor is selected for allocation that minimizes the execution time of the selected task using insertion-based policy while preserving the precedence constraints among the tasks.

The authors in [4] propose an algorithm Standard Deviation of Probability of Processor Availability (SDPA) based task scheduling algorithm for bounded number of processors that considers resource availability in scheduling decisions. SDPA algorithm works in two phases; during first phase, priority list is formed based on the upward rank of the tasks and during the second phase, tasks are selected for execution in the order of priority list and a processor that minimizes the execution time is allocated based on insertion based policy by considering the standard deviation of probability of processor availability in scheduling decisions.

Resource providers [5] in cloud offer a wide range of VM types to its users. Clouds offer VM instances with varying configurations in terms of compute, storage and network bandwidth such that they are optimal for certain types of applications. Resource providers scale dynamically in and out their resource pool. Resource provisioning algorithms use strategies such as static and dynamic resource provisioning for workflow executions. In static resource provisioning all the decisions regarding the VM pool configuration is done before the execution of the workflow. On the other hand, in dynamic provisioning all the decisions are taken at runtime regarding provisioning and de-provisioning of VM's during the execution of workflow. In static resource provisioning, once the VM pool is determined, the leased resources remain active throughout the execution of the workflow and are released back to the provider when workflow execution completes.

For Virtual Machines, the provisioning delay [5] ranges from 50 seconds to 883 seconds and de-provisioning delay

may vary up to as low as 3 seconds. IaaS cloud providers make no guarantees on this delay and vary from provider to provider and VM type to VM type. Also, the performance of a VM 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.

The authors in [3] distinguish resources in grid environment in multiple states based on administrator's policies, failures in the underlying hardware, workload on the host machines etc., also they propose techniques for the resource availability prediction based on historical data and how the availability transits from available state to un-available state or vice versa.

The authors in [8] propose multi-state prediction algorithms that take length of time or estimated application execution time as input and uses historical data of resource's availability for predicting the probabilities of that resource. They proposed two approaches to analyse the resource's availability history; in the first approach they examined a resource's past N days of availability behaviour during the interval being predicted and in the second approach they examined a resource's most recent N hours of activity immediately preceding the prediction time interval.

ALGORITHM :
Begin SDPRA
// *N* represents set of Nodes
// VM represents set of statically provisioned virtual machines
// ppa(n_i,vm_j) is predicted probability of availability of virtual machine for the predicted time interval
// SD(n_i) standard deviation of probability of availability on available virtual machines

//phase 1: task prioritization
For all n_i **in** *N*
Compute ECT(ni) level wise
End For

ReadyTaskList ← *Start Node*

Generate the priority list based on the highest priority of a task i.e maximum ECT at each level

// phase 2: scheduling phase

For all vm_j in VM
$EST(n_i, vm_j) = \max(T_{available[j]}, \max_{nm \in pred(ni)}(EFT(nm, vm_k) + c_{m,i}))$
$EFT(, vm_j) = w_{i,j} + EST(n_i, vm_j)$
End For

For all vm_j in VM
Calculate average ppa(n_i) based on computation cost required for task execution

ESD= Mppa(n_i) - SD(n_i) // Mppa is Mean of average ppa(n_i)

If ((EFT(n_i,vm_j) ~= max(EFT(n_j))) AND (pap(n_i,vm_j) < ESD))

$EFT(n_i,VM_j) = EFT(n_i,VM_j) + (EFT(n_i,VM_j) * SD(n_i))$
Else
$EFT(n_i,VM_j) = EFT(n_i,VM_j)$
End If
End For
Map node n_i on processor p_j which provides its least EFT_{vm}
Update $T\_Available[vm_j]$ and *ReadyTaskList*
End While

End SDPRA

IV. CLOUD WORKFLOW SCHEDULING BASED ON STANDARD DEVIATION OF PREDICTIVE RESOURCE AVAILABILITY

In this paper we propose a new heuristic called Cloud workflow Scheduling based on Standard Deviation of Predictive Resource Availability (SDPRA). The SDPRA algorithm is an application scheduling algorithm for bounded number of statically provisioned heterogeneous Virtual Machines that considers probability of resource availability for every time unit required for execution in the prediction

time interval and considers the averaged probability of resource availability for the computation time required on a resource in scheduling decisions.

The SDPRA algorithm works in two phases. In the first phase, the tasks are prioritized level wise. In level wise task priority, at each level the priority of all tasks are computed by their Expected Completion Time (ECT). The ECT is calculated based on the tasks average computation cost (ACC) and maximum data arrival cost (MDAC).

The average computation cost (ACC) of a task ti is computed by dividing the sum of computation cost on each VM by the the number of available VM's.

$$ACC(ti) = \Sigma \frac{wij}{m} \quad (1)$$

Where, j ranges from 1 to m number of virtual machines.

The MDAC of task is the highest amount of time that the task needs to spend to receive data among its predecessors. The Expected Completion Time (ECT) of a task is calculated as sum of the average computation cost of that task and the maximum data arrival cost of the same task.

$$ECT(t_i) = ACC(t_i) + MDAC(t_i) \quad (2)$$

At each level that with highest ECT will get the highest priority. Thus the priority list is formed for the given workflow represented as DAG.

During the second phase, the scheduler predicts the probability of resource availability for every time unit in the predicted time interval based on the historical data for all the available Virtual Machines. A selected task from the priority list assigned to a VM that minimizes the execution time considering the standard deviation of averaged predicted resource availability for the time required for execution and by incorporating the probable delay that occurs in execution using insertion based policy.

$$EFT(n_i,VM_j) = EFT(n_i,VM_j) + (EFT(n_i,VM_j) * SD(n_i)) \quad (3)$$

Our algorithm SDPRA is more reliable as it considers the probable delays based on the predicted probability of resource availability in its scheduling decisions.

V. EXPERIMENTAL RESULTS

We present the behaviour of our algorithm SDPRA and comparative result of our algorithm SDPRA with other static scheduling algorithms like HEFT and ECTS. For testing the algorithm we used an example graph with 10 nodes given in the Figure 1, randomly generated graphs of various sizes and also considered real life application graph such as Gauss graph [6], Montage graph, Cybershake graph and Epigenomics graph [5], with generated predicted probability of resource availability based on random normal distribution with mean 0.5 and standard deviation of 0.2 for the predicted time interval.

In this section we discuss comparison metrics, algorithm used for Random DAG generation and results..

A. *Comparison Metrics [4, 6,7]*

The SDPRA algorithm is compared with other existing algorithms based on the following metrics:

a) Makespan: Makespan or schedule length is the overall execution of all the tasks from start node to exit node in a DAG and is the main performance measure of a scheduling algorithm.

b) Schedule Length Ratio (SLR): The best scheduling algorithm is the one that gives the lowest SLR of a graph. Average SLR values are considered for performance evaluation of task graphs. The SLR is the ratio of the parallel execution time to the sum of weights of the critical paths tasks on the fastest processors.

c) Speedup: Speedup of scheduling algorithm is computed by dividing sequential execution time by the parallel execution time (makespan). The sequential execution time of a DAG is calculated by assigning all sub-tasks to a single processor which minimizes the cumulative computation costs. .

B. *Random DAG Generator:*

Our experimental set up considers a random graph generator algorithm given in [9] to generate Random directed acyclic graphs. This algorithm takes number of nodes as input and generates a weighted directed acyclic graph, where number of edges is generated randomly, based on number of nodes. Heterogeneity factor η [10] for virtual machine speeds depends basically on the range percentage of computation costs on virtual machines provisioned i.e., η = {0.1, 0.5, 1.0}

There is a significant difference in task's computation cost among processors when the range percentage of computation costs is high. The average computation cost wi of each task ti in the graph is randomly generated from a uniform distribution with range [0, $2*W_{dag}$], where W_{dag} is generated randomly based on the number of nodes in the graph.

$$W_i * (1 - \eta/2) <= W_{ij} <= W_i * (1 + \eta/2) \quad (4)$$

Our simulated framework first executes Random Directed acyclic Graph Generator Program to generate random directed acyclic graphs of various sizes. It takes number of nodes, number of virtual machines required as input and generates a random directed acyclic graph with randomly generated computation cost matrix, and communication cost matrix.

To study the performance of our SDPRA algorithm we used randomly generated directed acyclic graphs of various sizes such as 10,20,30,40 and 50 and randomly generated predicted probability of availability for predicted time interval using random normal distribution with mean 0.5 and standard deviation of 0.2 to generate output schedule. For the same set of graphs we also implemented HEFT and ECTS algorithms and estimated the delay that results for HEFT and ECTS schedules based predicted probability of availability for the predicted time interval. Performance metrics are computed based on output schedules.

C. *Results:*

Experimental results are organized using four test sets of graphs as follows:

1) Test Set one: In test set one, we considered the sample graph given in the above section 2, our algorithm SDPRA which considers predicted resource availability for the predicted time interval and the performance of SDPRA is compared with existing HEFT and ECTS algorithms that

considers 100% resource availability for scheduling. SDPRA algorithm produces resultswhich is about 27% better than HEFT and 19% better than ECTS in terms of performance metrics Makespan, SLR and Speedup. Our algorithm is executed for different predicted probability of availabilities for about 200 times and averaged results of these executions are compared with existing algorithms. We also estimated the probable delay that occurs for HEFT schedule, ECTS schedule for the same predicted probability of availability for the predicted time interval.

Comparative results of the sample graph given in section 2 are shown in Figure 2, Figure 3 and Figure 4 as follows:

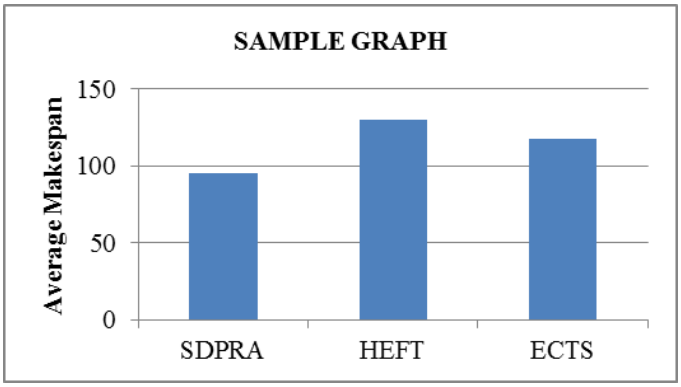

Figure 2 Average makespan of sample graph

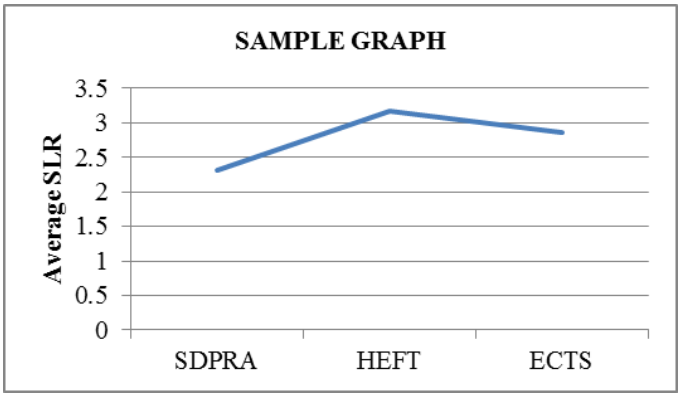

Figure 3 Average SLR of sample graph

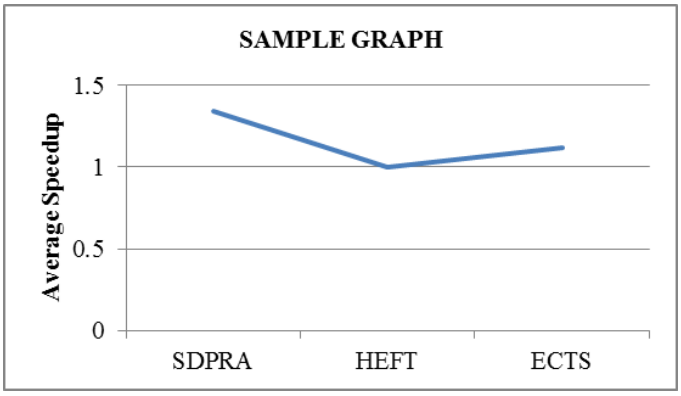

Figure 4 Average Speedup of Sample graph

2) Test Set two: In test set two, we considered various randomly generated DAGs of sizes 10,20,30,40 and 50 nodes. Our SDPRA algorithm is executed for these random DAGs. Each random DAG generated is executed 200 times with different predicted probability of availabilities for predicted time interval and performance of the SDPRA algorithm is compared in terms of different graph sizes. The same sets of random graphs are executed for HEFT algorithm as well as ECTS algorithm and probable delay for HEFT and ECTS is evaluated for different predicted probability of availabilities for predicted time interval. Results show that SDPRA algorithm is more reliable than HEFT and ECTS. For random graphs of various sizes, the overall performance improvement of SDPRA in comparison with HEFT ranges from 45% to 94% and ECTS ranges from 50% to 90% and at an average 71% for HEFT and 70% for ECTS. Comparative results are shown in the Figure 5, Figure 6 and Figure 7 below:

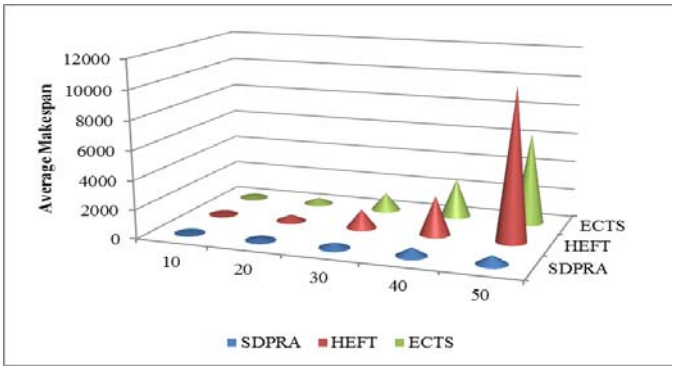

Figure 5 Average Makespan of randomly generated graphs of various sizes

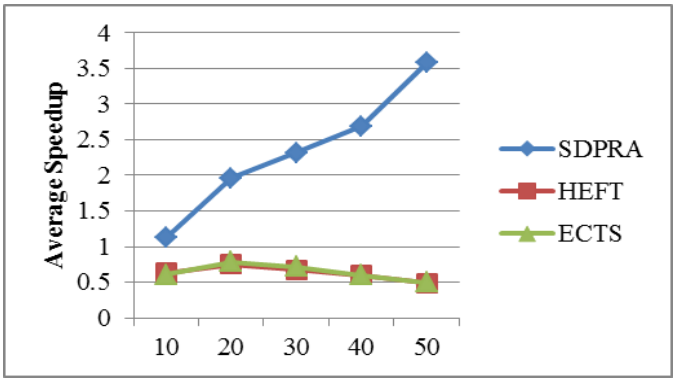

Figure 6 Average SLR of randomly generated graphs of various sizes

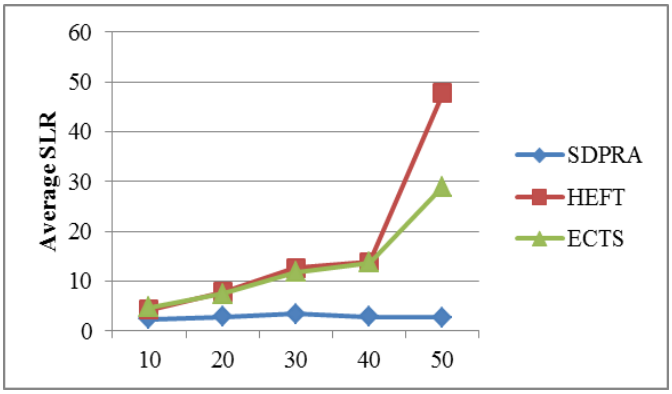

Figure 7 Average Speedup of randomly generated graphs of various sizes

3) Test Set Three: Application Graphs - Many scientific applications are modelled as workflows that can be processed efficiently in distributed cloud environments. In this research work we consider application graphs such as Gauss gpaph, Cybershake graph, Montage graph and Epigenomics graph. The Montage workflow is an I/O intensive astronomy application that is used to create custom mosaics of the sky based on a set of input images. During the execution of the workflow, the geometry of the output image is calculated from that of the input images. Astronomers can generate composite large images of a region of the sky using various input images. In the bioinformatics field, the CPU intensive Epigenomics workflow is used to automate the execution of various genome sequencing operations. Cybershake workflow is a data and memory intensive earthquake hazard characterisation application used by the South California Earthquake Centre. Gaussian Elimination graph is a compute intensive scientific application.

The overall performance improvement of SDPRA algorithm is about 80% better than HEFT and 74% better than ECTS for an I/O intensive application graph such as Montage graph. The performance improvement of SDPRA algorithm is about 69% better than HEFT and 70% better than ECTS for a CPU intensive application graph such as Epigenomics. The performance improvement of SDPRA algorithm is about 73% better than HEFT and 76% better than ECTS for a data and memory intensive application graph such as Cybershake. The overall performance improvement of SDPRA algorithm is about 57% better than HEFT and 60% better than ECTS for a scientific application graph such as Gaussian Elimination graph.

The performance of SDPRA and comparative results with HEFT and ECTS of I/O intensive Montage application graph are shown in the Figure 9, Figure 10 and Figure 11 below:

Figure 8 Montage Graph

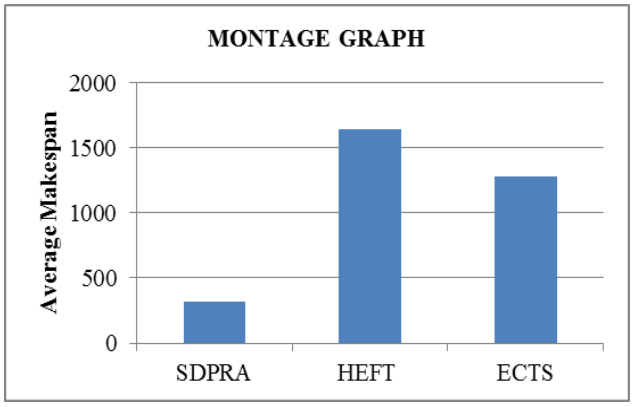

Figure 9 Average Makespan of Montage Graph

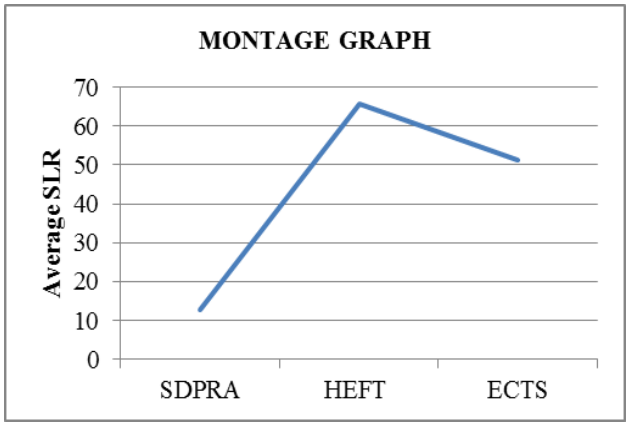

Figure 10 Average SLR of Montage Graph

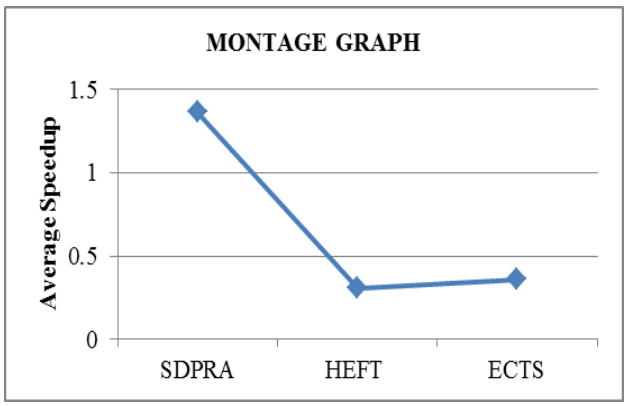

Figure 11 Average Speedup of Montage Graph

The performance of SDPRA and comparative results with HEFT and ECTS of CPU intensive Epigenomics application graph are shown in the Figure 13, Figure 14 and Figure 15 below:

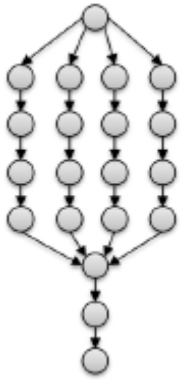

Figure 12 Epigenomics Graph

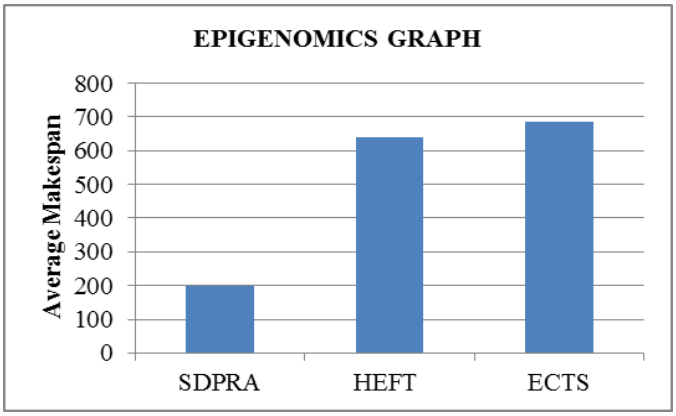

Figure 13 Average Makespan of Epigenomics Graph

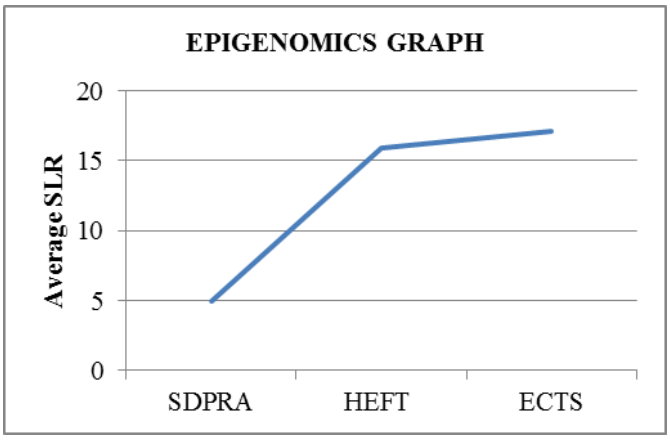

Figure 14 Average SLR of Epigenomics Graph

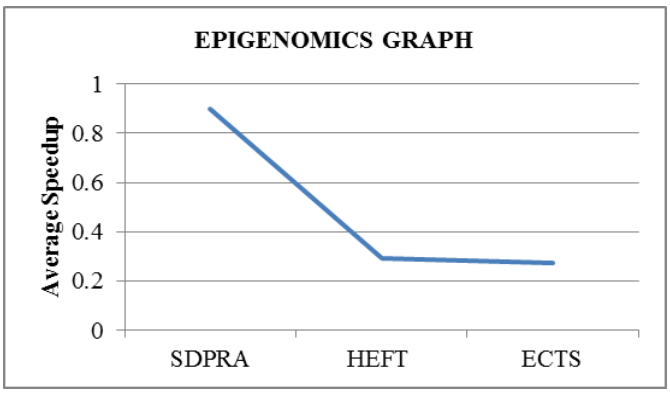

Figure 15 Average Speedup of Epigenomics Graph

The performance of SDPRA and comparative results with HEFT and ECTS of CPU intensive Gaussian Elimination application graph are shown in the Figures 17, Figure 18 and Figure 19 below:

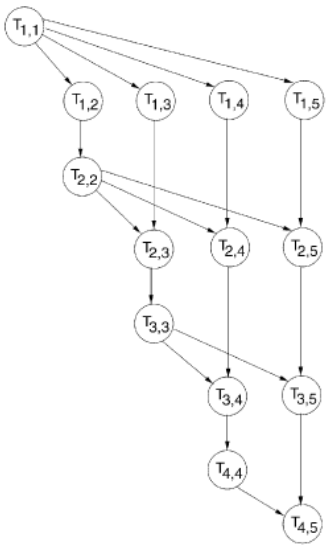

Figure 16 Gaussian Elimination Graph

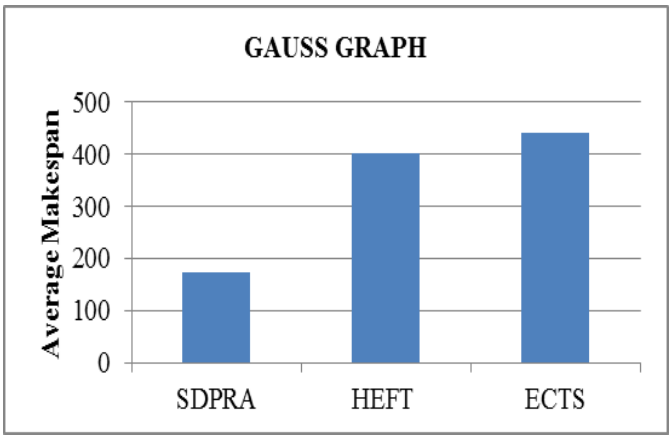

Figure 17 Average Makespan of Gauss Graph

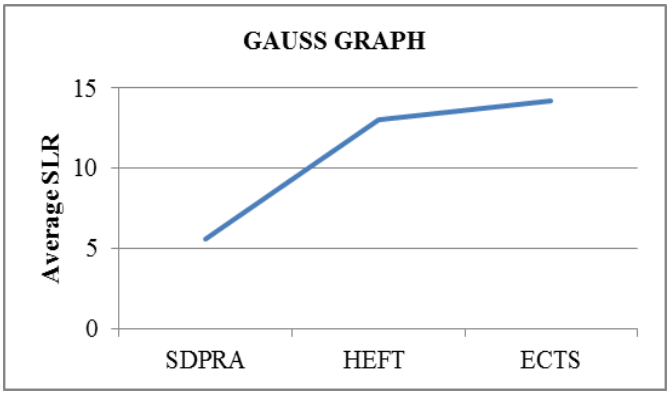

Figure 18 Average SLR of Gauss Graph

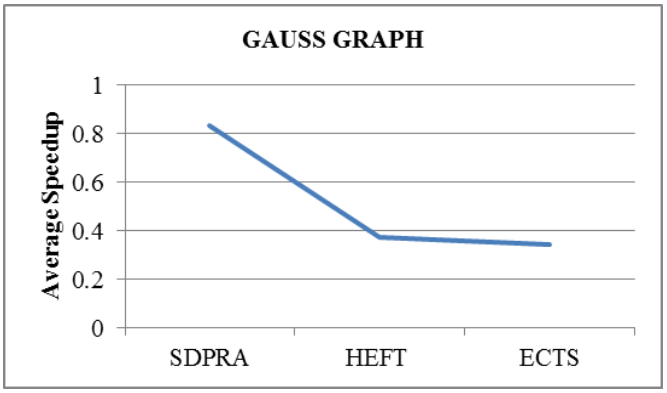

Figure 19 Average Speedup of Gauss Graph

The performance of SDPRA and comparative results with HEFT and ECTS of CPU intensive Cybershake application graph are shown in the following Figure 21, Figure 22 and Figure 23 below:

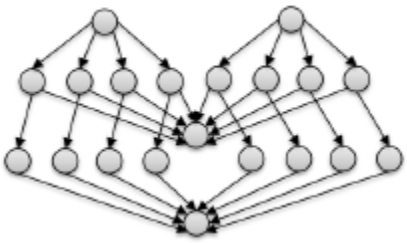

Figure 20 CyberShake Graph

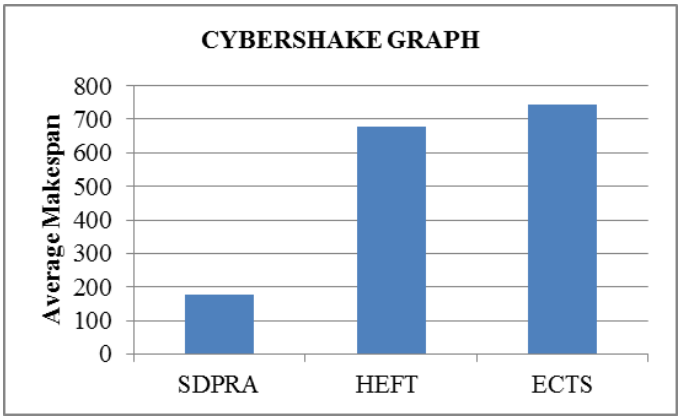

Figure 21 Average Makespan of CuberShake Graph

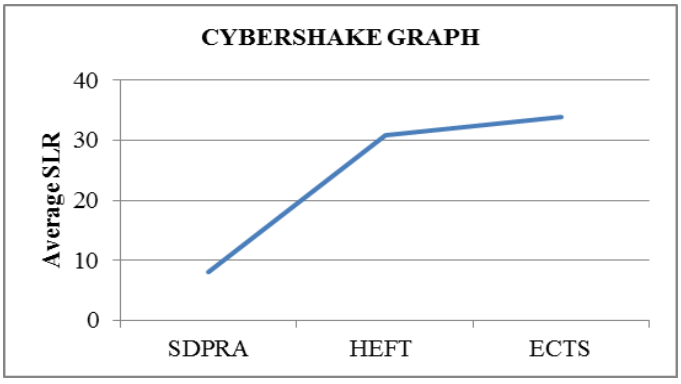

Figure 22 Average SLR of CyberShake Graph

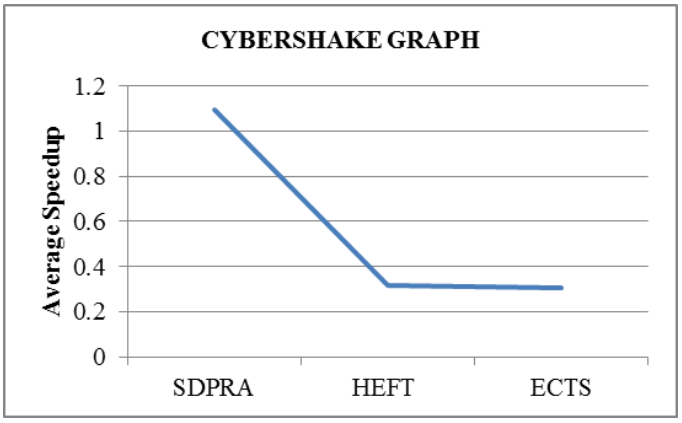

Figure 23 Average Speedup of CyberShake Graph

4) Test Set four: In test set four, a random graph with 50 nodes is considered to study the performance of our algorithm SDPRA in terms of parallelism for various numbers of processors such as 9, 12 and 15. Experimental results in the Figure 24 show that makespan is minimized with increasing number of processors with increased reliability.

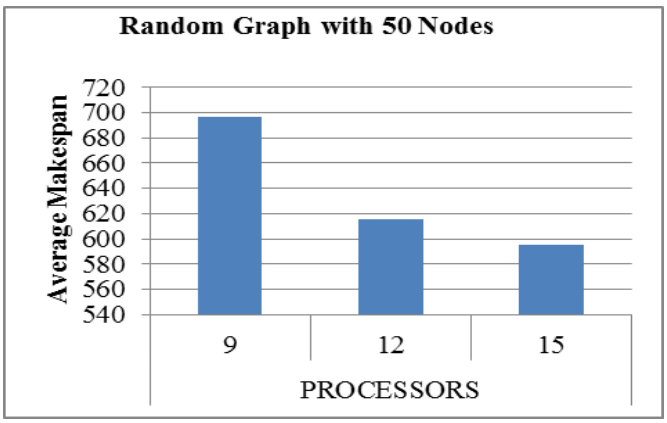

Figure 24 Average Makespan of a randomly generated 50 node graph with varying number of Virtual machines

Static scheduling algorithms HEFT and ECTS for bounded number of processors, in which tasks are prioritized and scheduled to a virtual machines based on Earliest Finish Time assume that all processors are 100% available for allocation. In real cloud environment, due to various reasons resources may not be available and delays may occur and it may not achieve the required makespan that leads to low reliability and unpredicted delays.

The simulation results given in this section show that our SDPRA algorithm is more reliable than HEFT and ECTS that considers 100% resource availability, as is considers predicted probability of availability for the predicted time interval in scheduling decisions and avoids unpredicted delays.

VI. CONCLUSION

Resources allocated for workflow execution in the cloud environment exhibit unpredictable and unstable performance

due to sharing of underlying resources as well as virtual resources by various users as their availability is highly dynamic and show significant impact in scheduling [5]. In this paper we propose a new heuristic called Cloud Workflow Scheduling based on Standard Deviation of Predictive Resource Availability (SDPRA) for a bounded number of statically provisioned heterogeneous Virtual Machines that considers resource availability factor in scheduling decisions. SDPRA algorithm works in two phases. In the first phase, tasks are prioritized based on Expected Completion Time level wise and a priority list has been formed. In the second phase, the scheduler predicts the probability of availability of the Virtual Machines based on historical data for each time unit for the predicted time interval and tasks are scheduled to a processor with minimum EFT considering the delay that occurs due to probability of availability at each time slot required. Our algorithm SDPRA is upheld to be reliable for scheduling workflow applications structured as DAGs on to a heterogeneous cloud where availability of resources are dynamic, unpredictable and shared by various users. Our algorithm outperforms HEFT and ECTS in the environment where resource availability is considered. The performance of SDPRA algorithm has been witnessed experimentally by using an example graph, randomly generated graphs of various sizes and application graphs such as Gauss graph, Montage graph, Cybershake graph and Epigenomics graph.

For the example 10 node graph considered in Figure 1, average makespan, average SLR and average Speedup of SDPRA is about 27% better than HEFT and 19% better than ECTS. The overall performance Improvement of SDPRA algorithm is about 80% better than HEFT and 74% better than ECTS for an I/O intensive application graph such as Montage graph. The performance improvement of SDPRA algorithm is about 69% better than HEFT and 70% better than ECTS for a CPU intensive application graph such as Epigenomics. The performance improvement of SDPRA algorithm is about 73% better than HEFT and 76% better than ECTS for a data and memory intensive application graph such as Cybershake graph. The performance improvement of SDPRA algorithm is about 57% better than HEFT and 60% better than ECTS for a scientific application graph such as Gaussian Elimination graph.

For random directed acyclic graphs of various sizes 10,20,30,40 and 50 the overall performance improvement of SDPRA in comparison with HEFT ranges from 45% to 94% and ECTS ranges from 50% to 90% and at an average 71% for HEFT and 70% for ECTS.

The simulation results show that SDPRA algorithm is more reliable with predictive resource availability for a predicted time interval, than existing algorithms that assumes 100% resource availability for scheduling where availability of resources are unpredictable in the cloud environment.

VII. REFERENCES

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems, Volume 25, Number 6, Pages: 599-616, ISSN: 0167-739X, Elsevier Science, Amsterdam, The Netherlands, June 2009.

[2] Linlin Wu, Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya, SLA-Based Resource Provisioning for Hosted Software-as-a-Service Applications in Cloud Computing Environments, IEEE Transactions on Services Computing (TSC), Volume 7, Number 3, Pages: 456-485, ISSN: 1939-1374, IEEE Computer Society Press, USA, July-September 2014.

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

[4] Vijayalakshmi A. Lepakshi, Prashanthe C S R, Standard Deviation of probability of processor based task scheduling algorithm in cloud computing, IEEE International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15), 2015, DOI: 10.1109/ITACT.2015.7492641

[5] 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), Volume 29, No. 8, Pages: 1-23, ISSN: 1532-0626, Wiley Press, New York, USA, April 25, 2017.

[6] Haluk Topcuoglu, Salim Hariri, Min-You Wu "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing" 1045-9219/02/$17.00 © 2002 IEEE

[7] 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 (2011) © (2011) IACSIT Press, Singapore

[8] Brent Rood and Michael J. Lewis, "Grid Resource Availability Prediction-Based Scheduling and Task Replication," Journal of Grid Computing, 2009

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

[10] E. Ilavarasan P. Thambidurai and R. Mahilmannan, "Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System" Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPDC'05) 0-7695-2434-6/05 $20.00 © 2005 IEEE