Scheduling Workflows with Data Host Reservation in Cloud Computing

Vijayalakshmi A Lepakshi [#1], Prashanth C S R [*2]
[#] Research Scholar, NHCE, VTU, Bangalore, India
[1] vjlepakshi@gmail.com
[*] HOD, Department of Computer Science and Engineering, NHCE, VTU, Bangalore, India
[2] drprashanthcsr@gmail.com

Abstract—**In cloud environment, availability of resources for job execution is highly dynamic and temporal in nature. Resources in cloud range from processing elements such as Virtual Machines (VMs), Data Servers, Network Bandwidth etc. More over VMs in cloud are extensively available for execution with elastic nature of IaaS cloud. In cloud, VMs share physical resources and they cannot store data required to process the application. In this scenario, when a task is executed on a VM, the data required for execution must be retrieved from cloud data storage servers. As these servers and network bandwidth required for transferring data are shared by the different VMs in the cloud, non-availability of these resources may account to overall task execution delay. In this paper, we propose, a novel static scheduling heuristic called Scheduling workflows with Data Host Reservation in Cloud Computing (SDHR). In the first stage we predict Data storage server's availability and upon requirement request is made to increase the availability of data hosts to avoid delays. In the second stage based on location of data, a VM that is associated with data host is selected and scheduled for execution. Our SDRH algorithm performs better by considering data host reservations than existing heuristics without reservations.**

Keyword- Cloud Computing, Directed Acyclic Graphs, Data Host, Makespan, Resource Availability, Reliability, Task Scheduling, Time Slots, Virtual Machines

I. INTRODUCTION

Cloud computing is a distributive computing [1] with interconnection of clusters and grids, which is the newly emerged trend that facilitates subscription oriented services such as platform, software applications and hardware infrastructure. These services are known as Software as a Service, Platform as a Service and Infrastructure as a Service and so on. Through these utility computing, end users can avoid upfront investment for establishing infrastructure for data storage and to provide computation power to uncertain or fluctuating demand. Cloud computing environment provides support for executing both computationally intensive and data intensive parallel applications. High performance can be achieved by scheduling the applications efficiently to the available resources. Cloud Service Providers (CSPs) offer services based on customized Service Level Agreements which define user's required Quality of Service parameters such that required reliability can be achieved in job completion. Parallel applications [2] are defined as a set of computational tasks and a set of data with dependencies. Scheduling parallel applications onto a heterogeneous environment is a well-known NP-complete problem.

IaaS clouds offer flexible scalable infrastructure for deploying the parallel applications and they also offer other resources such as storage services, network infrastructure to transport data. To take full advantage of these services, scheduling algorithms must consider many key characteristics of cloud. An important characteristic of cloud to be considered is dynamic state of the resource and the uncertainties that brings with it [3]. Cloud users can utilize these resources according to their requirement i.e., pay as you use basis. When users submit their jobs, based on the availability of resources jobs will be executed. Resource availability depends on administrators' policies. High resource availability leads to faster job execution else delayed job execution with non-available resources. In scheduling theory, basic assumption is that resources are always available for computation [4] i.e 100% available. Availability is the probability that the system is available continuously at any random period of time during a given interval. The wide range of policies for resource availability makes the problem of allocating resources to tasks even more challenging. As the cloud users do not have any control over the cloud infrastructure, there is no guarantee that resources are allocated to the cloud user unless some type of reservation is made beforehand [5]. Several researchers have addressed these issues and proposed the need for advance reservation (AR), to ensure that the specified resources are available for applications as and when required [6],[7],[8]. Common resources that can be reserved or requested are storage elements, compute nodes, network bandwidth or a combination of any of those. In general, reservation of the above mentioned resources can be categorized into two: immediate and advance. However, the main difference is the starting time between these two reservations. Immediate or best-effort reservation acquires the resources to be utilized instantly, whereas advance reservation delays their usage later in the future. Advance reservation [8] can be useful for several applications, such as: Parallel applications, where each task requires multiple compute nodes simultaneously for execution; Workflow applications, where each job may depend on the execution of other jobs in the application. Hence, it needs to wait for all of the dependencies to be satisfied before it can be executed. Therefore, by reserving resources in advance, users can get their jobs executed faster without delays leading to high reliability. Jobs reserving resources in advance are treated as high priority jobs. Hence reservation requests need to be checked for conflicts with currently running jobs and existing reservations.

Haizea [9] is a lease scheduler that provides resources for scheduling as advance-reservation and best-effort leases. In order to honour advance-reservation leases, Haizea pre-empts best-effort leases.

In this paper we propose a new static scheduling heuristic called Scheduling workflows with Data Host Reservation in Cloud Computing (SDHR). Since the schedule time of an algorithm is a key constraint for performance, we intend to deliver quality schedule with increased reliability. The SDHR algorithm in the first phase, checks for the probability of Data host availability and if all the resources are within the range of standard deviation of availability it proceeds to the next phase. Otherwise scheduler makes request for required amount of resources with a standard deviation factor, such that all the resources are within the standard deviation range and required reliability can be achieved. In the second phase, tasks are prioritized based on the upward rank of the task and the selected task is assigned to the resource that minimizes its earlier finish time by considering non available time slots based on insertion based policy.

The rest of the paper is organized in following sections as follows: In Section 2, the problem statement is defined. In Section 3, we discuss related work. In Section 4, a new static task scheduling heuristic SDHR is introduced. In Section 5, results of SDHR are presented. In section 6, the present research summary is given.

II. SCHEDULING PROBLEM

Scheduling Problem can be formally defined as follows: "Scheduling parallel applications modelled by Directed Acyclic Graph on to a Cloud, which is a heterogeneous interconnection of clusters, such that precedence constraints are satisfied and makespan is reduced with increased reliability".

A parallel application is represented by a Directed Acyclic Graph, $G = (V, E)$ where V is set of v number nodes and E is set of e number of edges between the tasks. Each edge $e(i,j)$ represents the precedence constraint such that task n_i completes execution before the task n_j. In a Directed Acyclic Graph, a task without any parent is called as an Entry node and a task without any child is called as an exit node. If there is more than one exit task, they are connected to a zero-cost pseudo exit task with zero-cost edges which do not affect the schedule. Each node label represents computation cost (expected completion time of the task) and each edge label represents communication cost (expected data transfer time) from current node to its successor node.

The main objective of the scheduling problem is to map tasks of a parallel scientific application to available resources and order their executions such that precedence constraints are satisfied and overall execution time is minimized with increased reliability. Finding an optimal solution to the problem of scheduling an application modelled by a Directed Acyclic Graph (DAG) onto a distributed system is known to be NP-complete. The complexity of task scheduling increases when scheduling is to be done in a heterogeneous cloud computing environment, where the virtual machines in the network are heterogeneous and take different amounts of time to execute the same task.

In the following Figure 1 an example task graph [10] is shown and Table 1 [10] shows computation cost matrix.

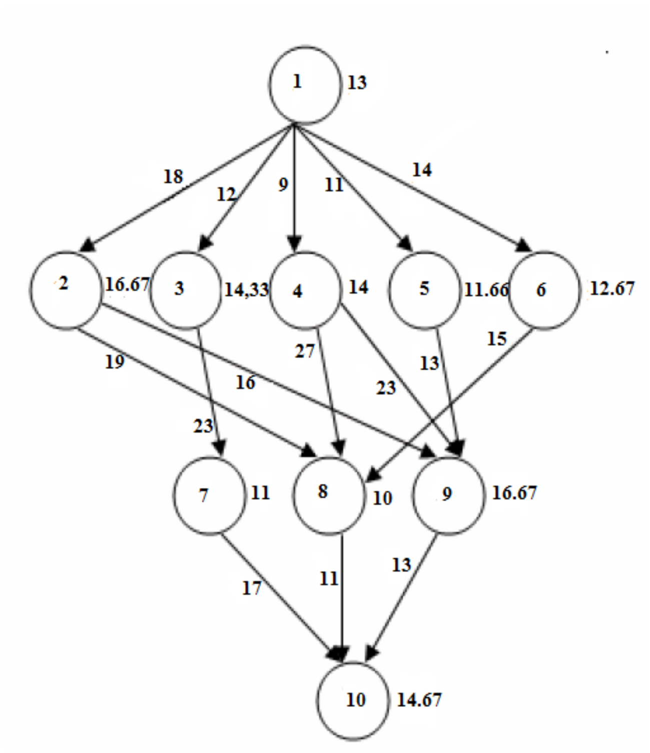

Fig.1 A sample DAG

TABLE IComputation Cost Matrix

Task	P1	P2	P3
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

A. Resource Model:

The predicted cloud computing environment is an inter-cloud environment and contains various heterogeneous virtual machines that belong to different providers and data centres that are located in various geographic locations. Virtual Machines are provisioned based on the location of the Data centres severs such that data transfer costs can be eliminated. In today's Big data era, many scientific applications use Big data stored in the Cloud Providers storage servers and VM cannot hold data that is required for computation. If an allocated VM for computation is away from data centre, additional resources such as high speed network and high network bandwidth are required leading to further increased costs and delays. Hence we use the concept bringing computation to the data than transferring data for computation by provisioning a VM with in the data centre's location and most of the providers do not charge for data transfers if a VM is provisioned with in their data centre. A Data Host serves all VM's located locally and also to other VM's located in the inter-cloud upon requirement. Hence, a data host may be available or not available depending upon the load. An inter grid gateway predicts the data storages servers availability based on the historical data and decision is taken based on the availability of data hosts.

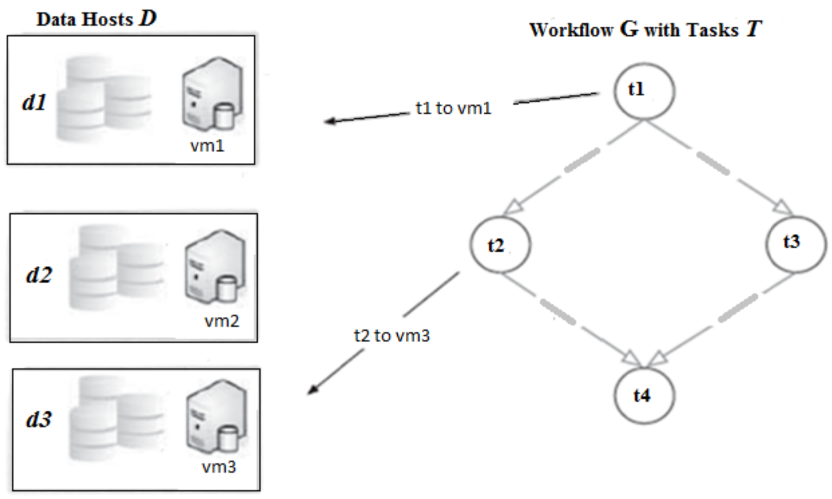

Fig.2 Resource Model

III. RELATED WORK

Many scientific applications are carried out worldwide to use existing infrastructures for conducting experiments produce huge amount of data. Data produced by such experiments are replicated and cached geographically in various locations. In this environment producing reliable time-efficient schedule is a challenge for parallel data-intensive applications that use distributed data and virtual machines for computing tasks. Many existing heuristic techniques either use single source data retrieval or multi-source parallel data retrieval for computing. These techniques may not produce time efficient schedules when data is independent of tasks and computing resources are not part of the datacentre that has the distributed data storage. These environments may result in delayed execution or incur high data transfer costs. When using distributed cloud environments, selecting computing resource and data host in a network based on their location is a basic building block. This selection should be such that execution time and cost of data retrieval is minimised. The authors in [11] propose Enhanced Static Mapping Heuristic (ESMH), in which data are transferred in parallel from multiple sources to a compute resource and selection of compute node is based on Earliest Finish Time. It considers both data transfer time and computation time for selecting a virtual machine for execution.

In the literature a large number of studies exists on various scheduling models, resource availability and advanced reservation i.e., reservation of resources prior to the execution so that delays can be avoided further improving reliability. Based on the availability of resources, advanced reservation can be done for static task scheduling. We studied various static scheduling algorithms for bounded number of heterogeneous processors such as HEFT, CPOP, ECTS and PETS etc. Our study includes resource availability and the concept of reserving resources in advance for task scheduling so as to improve reliability.

An application Scheduling algorithm Heterogeneous Earliest Finish Time HEFT [10] for bounded number of heterogeneous processors has two major phases. The first phase is a task prioritising phase for computing the priorities of all tasks. The second phase is known as processor selection phase in which tasks are selected in the order of their priorities and scheduled on best processor, which minimises the task's finish time. In Task Prioritizing phase each task is set with a priority using an upward rank value, $rank_u$, based on mean computation and mean communication costs. The sorted task list is prepared in the decreasing order of $rank_u$ providing a topological order of tasks and preserving the precedence constraints. In Processor Selection phase tasks are selected in the order of their priority and the selected task is assigned to the suitable processor, which minimises its earliest finish time using an insertion based approach. The time complexity of HEFT algorithm with *e* number of edges and *q* number of processors is O (*e* x *q*).

The Expected Completion Time based Scheduling (ECTS) [12] algorithm comprises of two phases: 1. Task Prioritization phase and 2. Processor Selection phase. During Task Prioritization phase, tasks are prioritized in two stages such as level wise task priority stage and task selection stage. In the first stage, at each level the priority of all tasks is computed by their Expected Completion Time (ECT). This Expected Completion Time is computed based on Average Computation Cost of tasks and Maximum Data Arrival Cost. In Processor selection phase, the processor that minimizes execution time of the selected task is allocated using the insertion-based scheduling policy while maintaining the precedence constraints among tasks.

Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System [13]:Performance Effective Task Scheduling Algorithm for Network of Heterogeneous system has the time complexity O(v+e)(p+log v). This algorithm consists of three phases: level sorting, task prioritization, and processor selection. In the first phase, the given DAG is traversed in a top-down fashion to sort tasks at each level and the tasks that are independent of each other are grouped together such that, tasks in the same level can be executed in parallel. In the second phase priority is computed and assigned to each task. Priority is computed based on the task communication cost and average computation cost. The Average Computation Cost (ACC) of a task is the average computation cost on all the available m processors. In the processor selection phase, the processor which gives minimum EFT for a task among all processors is selected for executing that task. It has an insertion-based policy to schedule tasks on a processor.

The Grid schedulers and resource brokers [14] require information about resource availability properties and also predictions about their future availability, to compare and select the most suitable resources, besides application execution time predictions on them.

The authors in [15] propose that Resource availability depends on administrators' policies. Also they analyse that availability of resources can be any five following states: Available resources to grid, User present, CPU threshold exceeded, Job eviction or Unavailable. Resource failure is not the only cause for unavailability of resources but user presence and high local CPU load etc., may account for unavailable resources. Prediction algorithms used to predict resource availability should not only consider hardware failure but should forecast other types of unavailability. They proposed failure-aware predictive grid scheduling that considers, why and how (not only when) resources become unavailable over a period of time. Availability is the probability that the system is available constantly at any random period of time during a given interval. Resource availability corresponds to the system ability to respond user requests including uptime percentage of cloud provider.

The authors in [16] define the Availability of services of a cloud provider in terms of storage as the ratio of the total time that the storage services are accessible during a given interval to the length of the interval. From application perspective available and unavailable resource durations during business hours and non-business hours can be analysed in terms of time and number of operations. Also, the authors in [17] propose that a parallel application requires uninterruptable resources for execution, hence the resources must be available for execution.

The authors in [18] classify the resources based on the policies for their availability in the Grid. Based on this criterion they identify three main classes of resources: dedicated, temporal and on-demand resources. The dedicated resources are meant to be always available to the users of Grid for production and experimental work. The temporal resources are available in the Grid as long as they are turned on for example Resources from university labs, but students very often turned them off while leaving the labs. Some resources are made available to the Grid users only on demand for large scale jobs. These are referred as on-demand resources. Moreover (un) availability of resources can be predicted hourly-basis, weekly-basis, monthly-basis or for a specified time interval.

The authors in [19] propose a model for grid resource availability. Their analysis and modelling results show that, computational resources in grid become unavailable at a high rate, negatively affecting the ability of grids to execute long jobs. They also analysed that resource availability can have a severe impact on the performance of the grid systems. They also proposed that the performance of a grid system can rise when availability is taken into consideration. The authors introduced following four models of grid resource availability information: 1. In this model, it is assumed that all resources are available at all the time i.e., Systems with Steady Availability (SA) 2. Systems with Known Availability (KA): This model assumes dynamic resource availability system. 3. Systems Automated Monitoring of Availability (AMA): This model assumes a dynamic resource availability system. It also assumes that a monitoring system provides the most recent resource availability information, which collects samples periodically on the grid for individual computing nodes' availability. If the high monitoring period is considered the monitoring information can be stale; if it is low, the monitoring overhead is unbearable for the grid. 4. Human Monitoring of Availability (HMA): This is similar to the AMA model, but assumes that the resource availability information is provided by the system administrator at fixed, but relatively large intervals: 1 week or 1 month for instance. The authors analyse that the performance is much better, when resource availability information is taken into account but even for a lowly utilized system the human administration of availability (HMA) change information results in 10 - 15 times more job failures than for an automated monitoring (AMA) solution.

Advanced reservation ensures that the specified resources are available for applications when required [6],[7],[8]. The issues related to resource availability can be addressed with advance reservation (AR). Also estimating the right amount of resources is important. By estimating proper resources an enterprise can avoid over-provision leading to wastage of resources and unnecessary costs or under-provision leading to failure of service or performance loss resulting end user dissatisfaction and economic loss [5]. When no advanced reservation is in place, resource unavailability may cause the failure of service by taking longer time to accomplish the task.

Advanced reservation is a process of reserving resources for future use. The resources such as CPU, Memory, Network bandwidth etc., can be reserved. The authors in [11],[20] propose a scheduling approach for Task Graphs by using advance reservation in a cluster environment that guarantees the availability of resources for execution in future. Advanced reservation facilitates concurrent access to resources for user applications to be executed in parallel. In a reservation based system, as the resources are guaranteed to be available for execution at the specified time, parallel and reserved jobs benefits significantly [20].

Maui Scheduler is an advanced scheduler used in clusters that supports Advance reservation, job accounting and QoS Policies etc. In Maui scheduler each reservation has three major components: a set of resources, a timeframe denoting starting and ending time, and an access control list (ACL) [21]. To reserve the resources, a user requests required resources by writing a task description which contains the exact required number of attributes, such as processing elements (PEs), memory, and hard disk. The access control list specifies which users, groups or jobs can use a reservation. Then, the Maui Scheduler will search for available resources based on the given task description and ACL. The Maui Scheduler uses a backfilling method to improve utilization, which executes smaller jobs waiting later in a queue, provided that they do not affect the start time of existing reservations.

Haizea is open source lease scheduler that is used for resource provisioning in clusters and datacentres [9]. Haizea provides Virtual Machine for scheduling on different types of leases such as best effort, advance-reservation and immediate. In order to honour advance-reservation leases, Haizea pre-empts best-effort leases.

IV. Scheduling workflows with Data Host Reservation in Cloud Computing (SDHR)

In this paper, we propose new heuristic called Scheduling workflows with Data Host Reservation in Cloud Computing (SDHR) algorithm. The SDHR algorithm is an application scheduling algorithm for static scheduling by considering availability of data hosts.

This algorithm works in two phases:

In the first phase, we check for the availability of data hosts and based on probability of available/non-available slots, data host reservation will be made. In general reservation of resources results in improved reliability but may lead to excess cost, over provisioning or resource underutilization as predefined constraints are to be met. Therefore estimating the right amount of resources is important such that an enterprise can avoid over-provision leading to wastage of resources and unnecessary costs or under-provision leading to failure of service or performance loss resulting end user dissatisfaction and loss of revenue. In this scenario we should decide judiciously the amount of advanced reservation of resources required such that reliability should be achieved. In this paper we propose a technique for data host selection. If the availability of all the data hosts is within the range of standard deviation of availability then it is possible to achieve the reliability in execution of workflow, so we need to pre-reserve the resources with the standard deviation factor so that availability of all resources are within the standard deviation range and required reliability can be achieved. The cloud scheduler estimates the availability of data hosts using above technique and produces data host availability matrix for each time slot for the future predicted interval based on the historical data. If data host is available to transfer data that is required for task execution then state of the data host is available, i.e., for the probability of data host availability > 0.5, a value 1 in the matrix otherwise a value 0 in the matrix for the time slot.

In the second phase there are two stages. In first stage tasks are prioritized based on upward rank of tasks. The upward rank of task n_i is defined recursively as follows:

$$rank_u(n_i) = w_i + \max_{n_j \quad \in(succ(n_i))} (c_{ij} rank_u(n_i))$$

Where succ(n_i) is immediate successors of n_j and c_{ij} is average communication cost.

Upward rank i.e., rank$_u$(n_i) can also be defined as the length of the critical path from task n_i to exit task including computation cost of n_i. Priority list is formed in which tasks are arranged in decreasing order of upward rank, which gives linear order of tasks that preserves precedence constraints.

In the second stage based on the priority, tasks are selected for execution. Availability of each data host's time slots is checked for the predicted time interval. Earliest Finish Time for a task n_i on virtual machine p_j is calculated considering non-available time slots.

$$EFT\ (n_i, vm_j)\ = w_{i,j} +\ EST\ (n_i, vm_j) + non\_available\ time\ slots\ on\ d_j$$

Selected task is allocated for execution to a virtual machine that gives minimum Earliest Finish Time among all available processors. Our algorithm SDHR is more reliable, as it considers Data host availability and computing time of virtual machine in scheduling decisions.

```
Begin SDHR
// N represents set of Nodes
// V represents set of Virtual machines
//D represents Data Hosts
// pda(1,$d_j$) is probability of availability of data host for predicted time interval generated randomly
//aslot($d_j$) contains next available starting position of data host
// SD is standard deviation of probability of Data host availability
//A($d_j$,t) is resource availability matrix for processor pj for a predicted time interval generated based
on PDA matrix
For all $vm_j$ in D
    ESD= Mpda($n_i$) - SD($n_i$) // Mpda is Mean of pda($n_i$ )
   If   (pda($d_j$) <  ESD))

        $pda(dj) = pda(dj) + SD(ni))$
  End If
 End For

For all $d_j$ in D
  Produce the Data host availability matrix A($d_j$, t) based on pda($d_i$)
// If predicted as available value for time slot is 1 else value for that time slot is 0
End for
 For all $n_i$ in N
        Compute $\text{rank}_u(n_i)$
End For
ReadyTaskList← Start Node
While ReadyTaskList is NOT NULL do
$n_i$ ← Node in the ReadyTaskList with the maximum ranku
For all $d_j$ in D
Compute aslot($d_j$)
$EST(n_i, vm_j) = \max(T_{available[j]}, \max_{n_m \in pred(n_i)}(EFT(n_m, vm_k) + c_{m,i}))$
If EST(ni,$vm_j$) <aslot($d_j$)
then EST($n_i$,$vm_j$)=aslot($d_j$)
end
          $EFT\ (n_i, vm_j)\ = w_{i,j} +\ EST\ (n_i, vm_j) + non\_available\ time\ slots\ on\ d_j$
End For
Map node $n_i$ on processor $vm_j$ which provides its least $EFT_{vm}$
Update T_Available[$d_j$] and ReadyTaskList
End While

End SDHR
```

V. EXPERIMENTAL RESULTS

We present the behaviour of our algorithm SDHR and comparative result of our algorithm SDHR with other static scheduling algorithm 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 Montage graph and Epigenomicsgraph[22], with randomly generated probability of data host availability and also based on the randomly generated available and non-available timeslots for the predicted time interval.

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

A. Comparison Metrics [10]

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

Makespan: Makespan is the main performance measure of a scheduling algorithm or Schedule Length of its output schedule.

Schedule Length Ratio: The SLR of an algorithm is defined as follows:

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

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.

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:

Random directed acyclic graphs are generated using random graph generator algorithm given in [23]. 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.

Our simulated framework first executes Random Directed A-cyclic Graph Generator Program to generate random directed acyclic graphs of sizes 10, 20, 30, 40 and 50. It takes number of nodes, number of processors, randomly generated computation cost matrix, and communication cost matrix.

To implement our SDHR algorithm we use randomly generated directed acyclic graphs of various sizes and probability of data host availability matrix and a random data host availability matrix for a predicted time interval based on probability of data host availability matrix as input to generate schedule outputs. We also implemented HEFT and ECTS algorithms for the same set of graphs and we also estimated the delay that results for HEFT and ECTS schedule based on the data host availability time slots matrix for the predicted time interval. Performance metrics are computed based on 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 3, our algorithm SDHR which considers resource availability for the predicted time interval is compared with existing HEFT and ECTS algorithms that considers 100% resource availability for scheduling. SDHR algorithm produces results which is about 22% better than HEFT and 16% better than ECTS in terms of performance metrics Makespan, SLR and Speedup. Our algorithm SDHR that considers non-available slots along with the available time slots in the predicted time interval based on probability of availability of data hosts and non-available slots vary each time.

Our algorithm is executed for different probability of data host availabilities for about 200 times and average of these executions are compared with existing algorithm. We also estimated the probable delay that occurs for HEFT schedule, ECTS schedule for the same data host availability for the predicted time interval.

Comparative results are shown as follows:

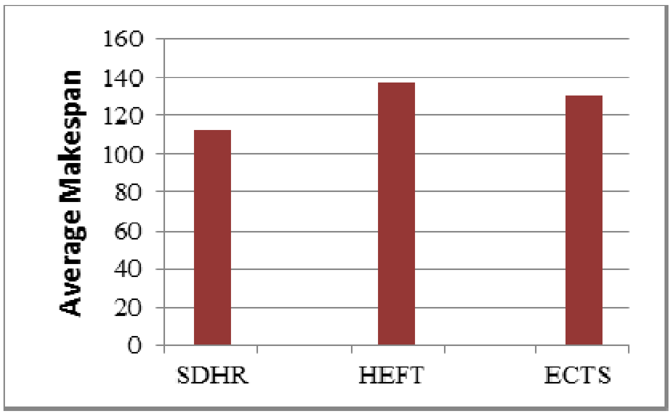

Fig.1 Average Makespan of Sample DAG given in section 2

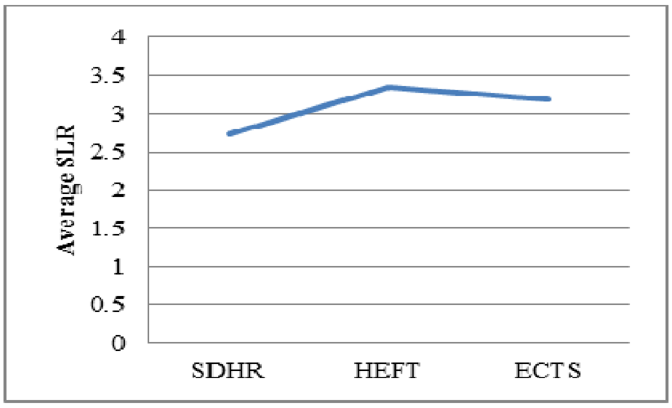

Fig.2 Average SLR of a sample DAG given in section 2

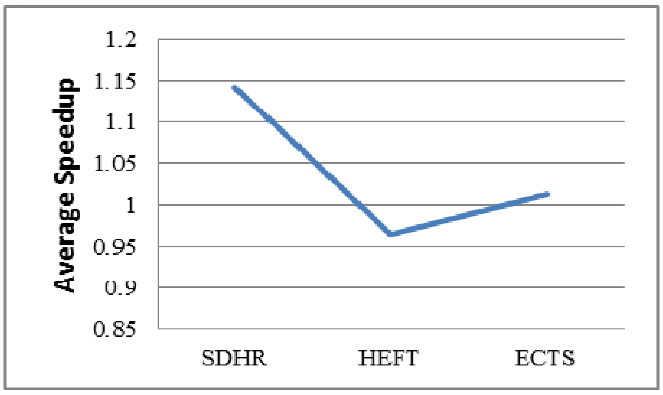

Fig.3 Average Speedup of sample DAG given in section 2

2) *Test Set two:*In test set two, we considered randomly generated DAGs of sizes 10,20,30,40 and 50 nodes. Our SDHR algorithm is executed for these random DAGs. Each random DAG generated is executed 200 times with different probability of processor availability and performance of the SDHR 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 probability of data host availability. Results show that SDHR algorithm is more reliable than HEFT and ECTS. For random graphs of various sizes, the overall performance improvement of SDHR in comparison with HEFT ranges from 10% to 41% and ECTS ranges from 15% to 26%.

Comparative results are shown below:

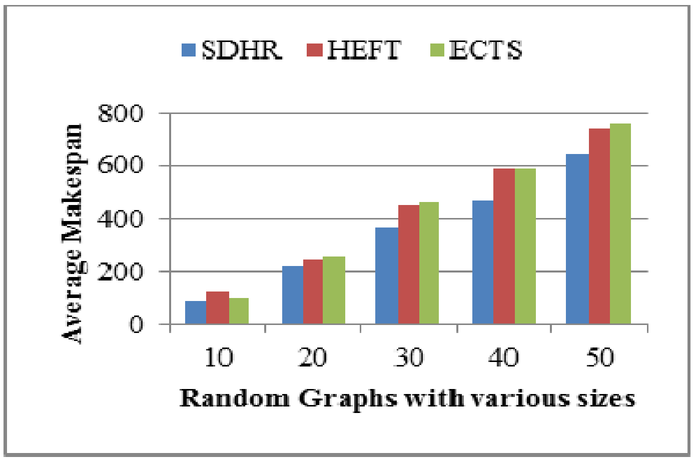

Fig.4 Average Makespan of Randomly generated various DAG's

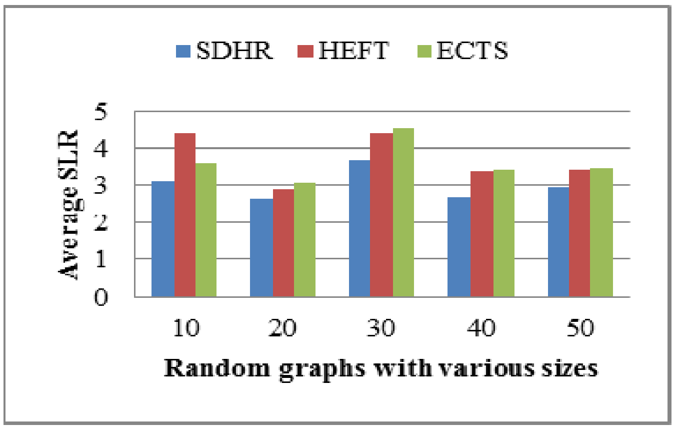

Fig.5 Average SLR of Randomly generated various DAG's

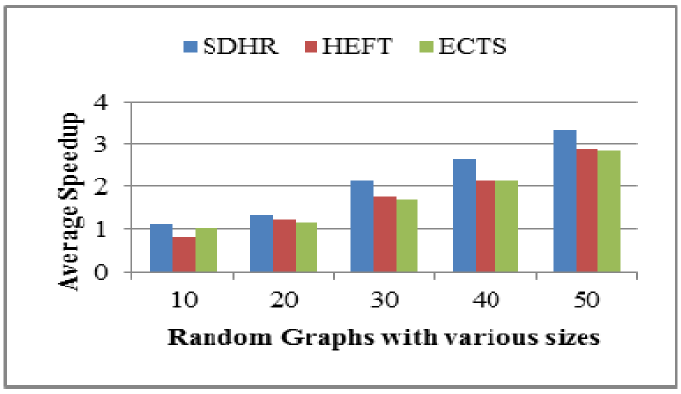

Fig.6 Average Speedup of Randomly generated various DAG's

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 two application graphs Montage graph [22] and Epigenomics graph [22]. 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.

The overall performance Improvement of SDHR algorithm is about 30% better than HEFT and 25% better than ECTS for an I/O intensive application graph such as Montage graph. The performance improvement of SDHR algorithm is about 22.9% better than HEFT and 8.4% better than ECTS for a CPU intensive application graph such as Epigenomics.

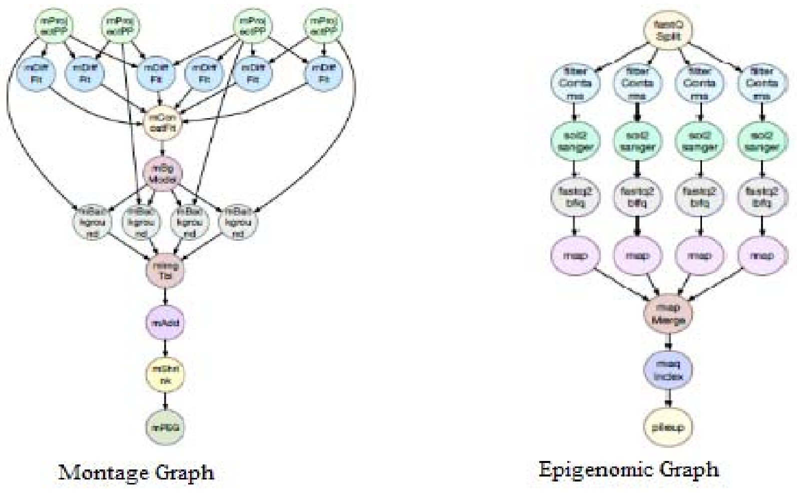

Fig.7Application graphs Montage and Epigenomics

The performance of SDHR and comparative results with HEFT and ECTS of I/O intensive Montage application graph are shown below:

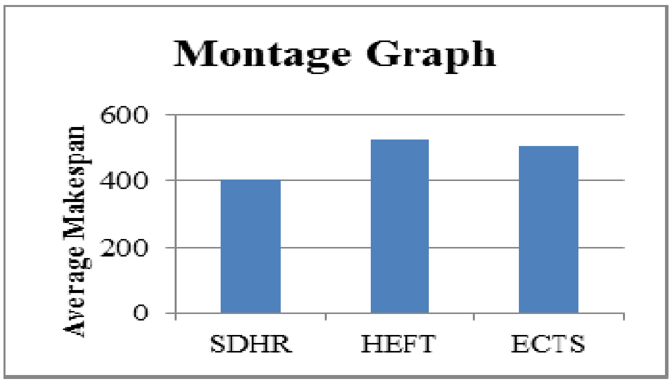

Fig.8 Average makespan of Montage graph

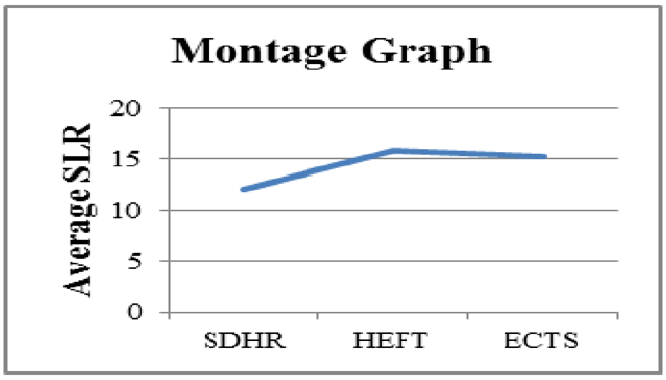

Fig.9 Average SLR of Montage graph

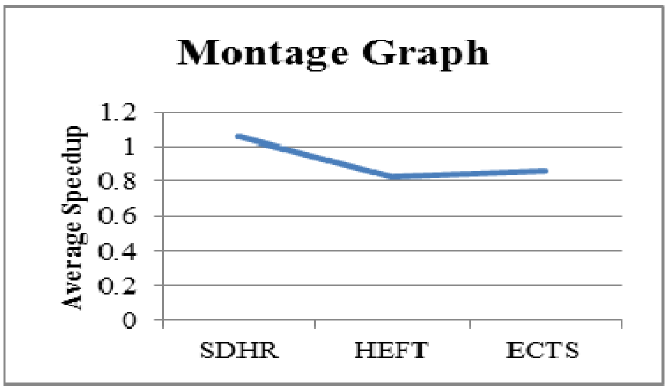

Fig.10 Average Speedup of Montage graph

The performance of SDHR and comparative results with HEFT and ECTS of CPU intensive Epigenomics application graph are shown below:

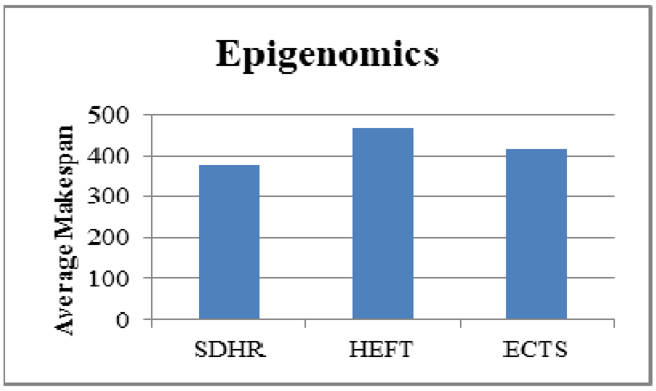

Fig.11 Average Makespan of Epigenomics graph

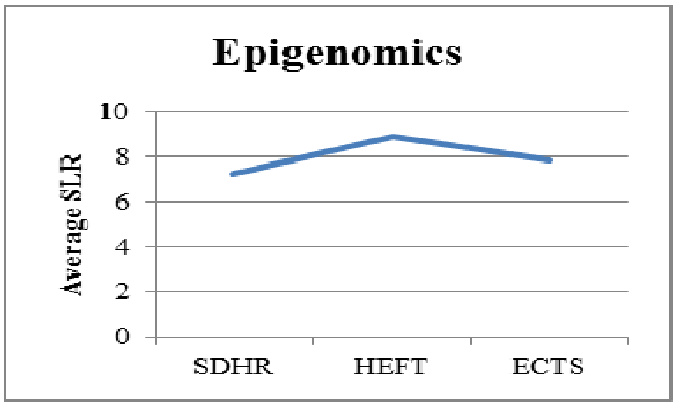

Fig.12 Average SLR of Epigenomics graph

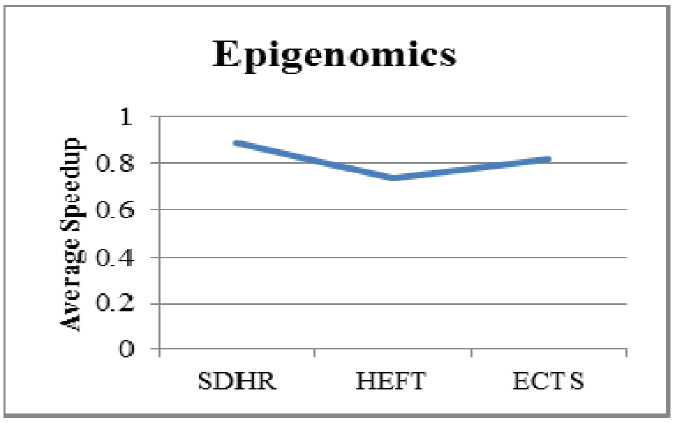

Fig.13 Average Speedup of Epigenomics graph

4) *Test Set four:* In test set four we considered a random graph with 50 nodes and performance of our algorithm SDHR is studied in terms of parallelism for various numbers of processors such as 9, 12 and 15 processors. Experimental results show that with increase in number of processors makespan is minimized with increased reliability without causing any delays.

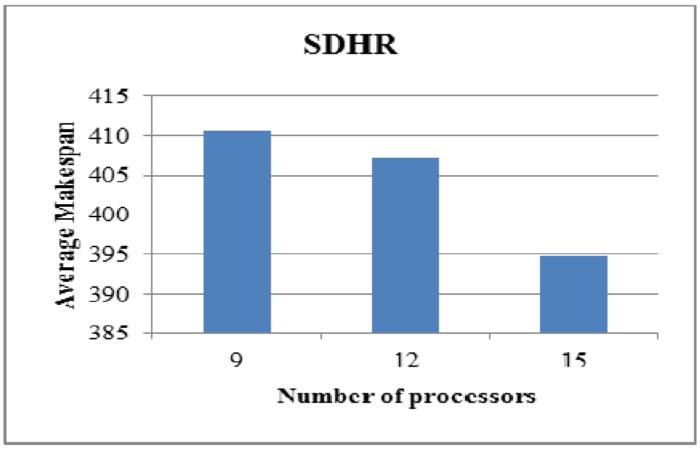

Fig.14 Average Makespan of a 50 node random graph with varying number of processors

A static scheduling algorithm HEFT 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.

The simulation results given in this section show that our SDHR algorithm is more reliable than HEFT and ECTS that considers 100% resource availability, as is incorporates Reservation of data hosts and considers resource availability for the predicted time interval during scheduling thus avoiding delays.

VI. Conclusion

In this paper we propose a new heuristic called Scheduling workflows with Data Host Reservation in Cloud Computing (SDHR) Algorithm for a bounded number of heterogeneous Virtual Machines that works in two phases. In the first phase, we estimate and request the need for right amount of resources based on the resource availability probability. In second phase, tasks are prioritized based on the upward rank and scheduled to a processor with minimum EFT considering Data Host available and non-available time slots for the predicted time interval. Our algorithm SDHR is proven to be reliable for scheduling parallel scientific applications structured as DAGs on to a cloud environment where resources are uncertain in nature and shared by various users. The performance of SDHR algorithm has been observed experimentally by using a sample 10 node graph, randomly generated various sized task graphs and application graphs such as Montage graph and Epigenomics graph.

For the example 10 node graph considered in Figure 1, average makespan, average SLR and average Speedup of SDHR is about 22% better than HEFT and 16% better than ECTS. The overall performance Improvement of SDHR algorithm is about 30% better than HEFT and 25% better than ECTS for an I/O intensive application graph such as Montage graph. The performance improvement of SDHR algorithm is about 22.9% better than HEFT and 8.4% better than ECTS for a CPU intensive application graph such as Epigenomics. For random directed acyclic graphs of various sizes 10,20,30,40 and 50 the overall performance improvement of SDHR in comparison with HEFT ranges from 10% to 41% and ECTS ranges from 15% to 26%.

The simulation results show that SDHR algorithm is more reliable with Data Host Reservation and also by considering resource availability for a predicted time interval than existing algorithms that assumes 100% resource availability for scheduling where resources are highly uncertain in the cloud environment.

References

[1] RajkumarBuyya, Chee Shin Yeo, SrikumarVenugopal, James Broberg, and IvonaBrandic, 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] J. Yu, R. Buyya, (2008) "Workflow scheduling Algorithms for Grid computing", metaheuristics for scheduling in distributing computing environment, Springer, Berlin.
[3] Maria A. Rodriguez and RajkumarBuyya, "A Responsive Knapsack-based Algorithm for Resource Provisioning and Scheduling of Scientific Workflows in Clouds "
[4] G. Schmidt, "Scheduling with Limited Machine Availability", European J. Operational Research, vol. 121, pp. 1-15, 2000.
[5] Kondo, D., Fedak, G., Cappello, F., Chien, A. A., and Casanova, H. "Resource availability in enterprise desktop grids." Tech.Rep.00000994,INRIA,2006.
[6] J. MacLaren,editor. Advance Reservations: State of the Art(draft). GWD-I,Global Grid Forum, June 2003. http://www.ggf.org
[7] A. Schill, F.Breiter, and S.Kuhn. Design and Evaluation of an advance reservation protocolon top of RSVP. In proceedings of the IFIP TC6/WG6.2 4th International Conference on Broadband Communications (BC'98), Stuttgant, Germany, April 1998.
[8] W.Smith,I.Foster,and V. Taylor, Scheduling with advanced reservations. In proceedings of the International Parallel and Distributed Processing Symposium(IPDPS '00), cancun, Mexico, May 1-5, 2000.
[9] Haizea Scheduler, http://haizea.cs.uchicago.edu/
[10] HalukTopcuoglu, Salim Hariri, Min-You Wu "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing" 1045-9219/02/$17.00 © 2002 IEEE
[11] SurajPandey and RajkumarBuyya, "Scheduling Workflow Applications Based on Multi-source Parallel Data Retrieval in Distributed Computing Networks", doi:10.1093/comjnl/bxr128, Published by Oxford University Press on behalf of The British Computer Society.
[12] 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
[13] 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
[14] FarrukhNadeem , RaduProdan, Thomas Fahringer, "Availability-based Resource Selection Risk Analysis in the Grid", CoreGRID Technical Report Number TR-0169 August 19, 2008
[15] Brent Rood and Michael J. Lewis," Multi-State Grid Resource Availability Characterization", IEEE 8th Grid Computing Conference, 2007
[16] YaserMansouri, Adel NadjaranToosi and RajkumarBuyya, Brokering Algorithms for Optimizing the Availability and Cost of Cloud Storage Services, Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom 2013, IEEE CS Press, USA), Bristol, UK, Dec. 2-5, 2013.
[17] Sulistio A., Schiffmann W., Buyya R. (2006) Advanced Reservation-Based Scheduling of Task Graphs on Clusters. In: Robert Y., Parashar M., Badrinath R., Prasanna V.K. (eds) High Performance Computing - HiPC 2006. HiPC 2006.
[18] FarrukhNadeem, RaduProdan, Thomas Fahringer "Characterizing, Modeling and Predicting Dynamic Resource Availability in a Large Scale Multi-Purpose Grid", Eighth IEEE International Symposium on Cluster Computing and the Grid 978-0-7695-3156-4/08 IEEE DOI 10.1109/CCGRID.2008.29
[19] AlexandruIosup, Mathieu Jan, OzanSonmez, Dick H.J. Epema, "On the Dynamic ResourceAvailability in Grids", 1-4244-1560-8/07/$25.00 © 2007 IEEE 26 , 8th Grid Computing Conference
[20] Anthony Sulistio and RajkumarBuyya, "A Grid Simulation Infrastructure Supporting Advance Reservation", Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004, ISBN: 0-88986-421-7, ACTA Press, Anaheim, California), November 9-11, 2004, MIT Cambridge, Boston, USA
[21] Maui Cluster Scheduler http://www.clusterresources.com/ pages/products/maui-clusterscheduler.php, 2008
[22] Maria A. Rodriguez and RajkumarBuyya, 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.

[23] 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

AUTHOR PROFILE

Vijayalakshmi A. Lepakshiis currently Assistant Professor, Department of Computer Science, Maharani Lakshmi Ammanni College for Women(Autonomous), Malleswaram, Bangalore, India. She completed M.Sc in Computer Science from Sri Krishnadevaraya University, Anathapur, Andhra Pradesh, India. She has completed M.Phil in Computer Science from The Global Open University, Nagaland.

Dr. Prashanth C S Ris currently the Dean, Professor and Head of the Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore-India. He obtained his B.E in Computer Science from Bangalore University, M.S in Computer Science from the University of Texas at Dallas, and Ph.D in Computer Science from Auburn University, USA. Dr. Prashanth has published extensively in the areas of High Performance Computing and Cloud Computing. He is also on the advisory board of several reputed international conferences and journals.