A Network Performance Aware QoS Based Workflow Scheduling for Grid Services

Shinu John[1] and Maluk Mohamed[2]
[1]Department of Computer Science and Engineering, St Thomas College of Engineering and Technology, India
[2]Department of Computer Science and Engineering, MAM College of Engineering and Technology, India

Abstract: *Grids enable sharing, selection and aggregation of geographically distributed resources among various organizations. They are now emerging as promising computing paradigms for resource and compute intensive scientific workflow applications modeled as a Directed Acyclic Graph (DAG) with intricate inter-task dependencies. Job scheduling is an important and challenging issue in a grid environment. There are various scheduling algorithm proposed for grid environments to distribute the load among processors and maximize resource utilization while reducing task execution time. Task execution time is not the only parameter to be improved; various Quality of Service (QoS) parameters are also to be considered in job scheduling in grid computing. In this Research we have studied the existing QoS based Task scheduling, work flow scheduling and formulated the problem. The possible solutions are developed for the problems identified in existing algorithms. The scheduling of dependent task (work flow) is more challenging than independent task scheduling. The scheduling of both dependent and independent tasks with satisfying QOS requirements of users is a very challenging issue in grid computing. This paper proposes a Novel Network aware QoS workflow scheduling method for Grid Services. The proposed scheduling algorithm considers network and QoS constraints. The goal of the proposed scheduling algorithm is to implement the workflow schedule so that it reduces execution time and resource cost and yet meets the deadline imposed by the user. The experimental result shows that the proposed algorithm improves the success ratio of tasks and throughput of resources while reducing makespan and workflow execution cost.*

Keywords: *Grid scheduling, QoS, DAG, execution time, deadline, trust rate.*

Received June 25, 2014; accepted September 7, 2015

1. Introduction

Grid computing is fundamentally based on a kind of parallel and distributing computing, namely cluster computing and point-to-point computing which can handle diverse computing services. This was achieved by the high speed internet and powerful processors that support middle wares without disturbing computer's normal job. The major differences among Grid computing and traditional distributed systems are as follows:

- There is no middle control over the computers.
- General-purpose protocols are used.
- The Quality of Services is typically very high.

As the speed of the internet increases, the distinction between two Personal Computers (PCs) functioning next to each other in a single building, or in a city or country regularly fades out. Consequently, users accomplish their tasks on geographically shared sources. The major concept followed in a Grid environment is that we can use the entire computational power of individual systems equally so that we utilize all resources separately. Similarly, it is aimed to develop an approach to connect the incredible computational power of the complete universe, where the cost is directly dependent on the amount of computational power being used. Grid scheduling has turned to be a major challenge. The significant challenges of scheduling in Grid environment are as follows:

- Resources are generally shared among users, so there is a competition among them.
- The scheduler is not in control of the sources.
- The number of available sources changes continuously.
- Sources are located at dissimilar management sites.
- Sources are heterogeneous.
- The majority of the workflow applications are data-centric and hence a large amount of data transfer is needed between two sites.

Scheduling in the grid can be considered as static and dynamic approaches. In the static method, each task is assigned once to a resource and its estimated cost of computation is prepared in advance of actual execution. On the other hand, dynamic scheduling is where a system is not conscious of the run-time behaviour of the application before execution.

The problem of task scheduling in workflows can be described by a Directed Acyclic Graph, known as (DAG). Scheduling is an optimization problem in the

context of traditional homogeneous or heterogeneous parallel computing, but in a grid environment, it is entirely dissimilar. Moreover, heterogeneity and substantial communication overheads, issues correlated to different administrative domains, are to be considered during a resource allocation to an application. All these problems might hinder the utilization of parallelism [8, 9]. The current challenges in scheduling workflow applications in grid computing are resource distribution in grids and competition for resources, etc., [3]. To overcome these problems, the concept of reserving resources in progress through the resource brokers [4] is resorted to. A resource broker is a common gateway to access grid resources. They make workflow implementation in Grids easy and reduce execution time. But, scheduling workflows considering users' Quality of Service (QoS) needs are not considered in current Grid workflow management systems. Pricing for a service is dependent on the required QoS level. In general service providers charge a higher price for higher QoS. Consequently, users may not require completing the workflows before the actual deadline. Instead, they favour using cheaper services with lower QoS that are enough to cater their needs. Keeping this in mind, this [19, 20] work presents a QoS-based workflow management, which aims to reduce execution cost, while satisfying users' QoS requirements

This research examines the fundamentals of workflow management, based on QoS requirements of service Grids and proposes a novel Network performance aware QoS workflow scheduling for Grid Services. The goal of the proposed scheduling algorithm is to implement the workflow schedule to reduce execution time, resource cost and yet meet the deadline imposed by the user. To resolve scheduling issues powerfully for large-scale workflows, we first calculate the priority value for each node of the DAG based on the priority value by partitioning the workflow tasks and readying the workflow for scheduling. A deadline assignment scheme is also developed to allocate the overall deadline over each partition. Based on the partitioning of tasks, a network aware QoS workflow scheduling algorithm is offered.

The motivation is that it predicts the trust of the resource node during task execution and then makes scheduling decisions regarding the trust of successful execution of tasks.

2. Related Works

Nadia and Zimeo [9] proposed a time and cost-constrained scheduling approach that, follows the data parallelism model, is capable of organizing scientific and business workflow tasks (or other application tasks) on pools of resources to reduce overall execution time. This research considers the difficulty of allocating heterogeneous computing resources to data parallel tasks of a Grid application with the fundamentals of QoS constraints, specifically time and cost. This algorithm doesn't evaluate the overhead. Moreover, reservation and execution time prediction techniques are explored at the middleware level to support the effectiveness of the algorithm with non-dedicated resources and more complex execution environments.

Gharooni-Fard *et al*. [4] presents a new genetic method known as "chaos-genetic approach" to resolve the scheduling problem addressing the user's budget and deadline. In this work, the cost of a service is normally related to the quality of the service it provides. Generally, the service providers charge more in response to a higher quality of service. Further, users may not always require completing workflows earlier than they need. Cheaper services with lower QoS enough to meet the user's requests are sometimes preferred. Therefore, a trade off between the time and monetary cost needs to be considered. In the above case the QoS parameters like reliability, stability, throughput and efficiency are not considered together.

Amalarethinam and Selvi [3] present a novel method for scheduling called Efficient Dual Objective Scheduling (EDOS) to improve the resource utilization in a grid and reduce makespan through advance reservation of resources and priority based task scheduling. In the EDOS approach, scheduling of reserving the resources in advance for complete workflow is static [12], but the mapping of resources to an executing task is dynamic. In this method, imputation of necessary resources is easier as the number of tasks in the DAG is identified in advance. This work is not focused to reduce communication cost. The algorithm does not consider failure of a task or resource during scheduling.

Abrishami *et al*. [1], and Abrishami and Naghibzadeh [2] introduce a scheduling technique for workflow, the critical path heuristics which attempt to schedule critical tasks initially. A novel QoS-based workflow scheduling algorithm based on the proposed model is known as Partial Critical Paths (PCP), which attempts to reduce the workflow execution cost with a user specified deadline. To improve performance, a high degree of concurrency is achieved for all time by running various instances concurrently. A scheduler must deal equally with potential deadlock problems due to oversubscribed resources and improving performance, as considered by makespan and/or throughput.

Wu *et al*. [16] presents a Scientific Workflow Automation and Management Platform (SWAMP), which allows scientists to collect, implement, examine, manage, and guide computing workflows in shared environments through a unified web based user interface. SWAMP also incorporates a specially considered mapping engine that automatically maps, conceptual workflows to underlying networks to

achieve Minimum End-to-end Delay (MED) and Maximum Frame Rate (MFR). The time changing environment of system and network resource's availability makes it a challenging problem to achieve an accurate estimation of the execution time of a module or transfer time over a link in real networks.

Abrishami *et al.* [1, 2] present an innovative heuristic scheduling algorithm for workflow based on QoS constraints, named Partial Critical Paths (PCP) algorithm. The goal of the PCP algorithm is creating a schedule that reduces total execution cost of a workflow while demanding a user-specified deadline for total execution time. This approach has two steps: Deadline Distribution and Planning. In the initial step, the whole deadline of the workflow is shared between individual tasks, so that it can execute it before the user's deadline and thereby reduce its execution cost. In the second step, the scheduler chooses the cheapest service for a task; likewise the entire workflow is completed before its sub deadline. This algorithm is not efficient for parallel pipelines. This method has greater time complexity, as a relatively large number of rescheduling is needed during execution of the algorithm.

Vasques and Veiga [15] introduces a framework for decentralized scheduling with effectiveness based scheduling algorithm that assumes partial request fulfilment to overcome the shortcomings of definite solutions. This scheduling algorithm assumes partial requirement execution based on information produced by users. Su *et al*. [13] presents a cost-efficient task-scheduling algorithm using dual heuristic strategies. The initial step dynamically maps tasks to the cost-efficient resources based on the idea of Pareto dominance. The second step complements the first and minimizes the monetary costs of non-critical tasks.

This method computes scheduling plans that ensures makespan as good as the best known algorithm while considerably reducing monetary costs. Task management is to represent a set of tasks with a workflow diagram, which can capture task decomposition, the communication between subtasks, and cost of computation and communication. But in [2, 11] the monetary cost does not include storage, network resources and communication cost. This work incorporates penalties for violating consumer-provider contracts.

Rahman *et al*. [10] proposed a Dynamic Critical-Path (DCP) based adaptive workflow scheduling approach for grids, which resolves efficient mapping of workflow tasks to grid resources dynamically by estimating the critical path in the workflow task graph at every step. However, this method is described for mapping tasks to resources, and is static, in the sense that the schedule is calculated once for a task graph. This algorithm extends the DCP algorithm to map and schedule tasks in a workflow to heterogeneous resources in a dynamic grid environment.

Wu *et al*. [17] deals with QoS scheduling based on following strategies:

1) The task owned higher priority should be scheduled prior to tasks with lower priority.
2) Atask should be completed as soon as possible.

In this, TS-QoS approach estimates the priority of the tasks based on their specific attributes and then sort them based on priority. The algorithm then calculates the completion time of each task on different resources, and schedules them to a resource which can complete the task as soon as possible according to the sorted task queue. But in this procedure priority varies dynamically and an increase helps to resolve the "starvation" difficulty followed by First Come First Serve (FCFS) concept. The scheduling is efficient and load balancing by QoS uses priority and execution time.

Hasham *et al*. [5] offers a pilot job idea that has intelligent data reprocess and job execution approach to reduce allocation, queuing, implementation and data access latencies. A pilot job is one that is accountable for setting up the necessary completing environment and for controlling the execution of a real job. A pilot job allows the traditional grid submission method; though, a job will bypass it as a pilot job downloads it from a global scheduler queue for completion. Through the aid of this method, a pilot job can support the job of finding all or several of its necessary files in the cache managed on the worker nodes. A job can begins implementation as quickly, if scheduled to a pilot job, thus minimizing queuing and allocation delays. If a job has completed its execution; a pilot job instantly notifies its execution status to the scheduling and monitoring mechanism for reducing delays.

Yousaf and Welzl [18] proposed an algorithm, network based Heterogeneous-Earliest-Finish Time (HEFT) where such data transfers are fixed to their practical execution time. A HEFT planned with fixed data transfers presents practical execution time for the schedule. Following this phase, HEFT is functional again with fixed communication costs and this is continual till a better schedule is found. Ijaz *et al*. [8] proposed a novel approach for mapping the DAG based application to available machines effectively. In this method A Minimal Latest Start Time algorithm was used to schedule tasks which have the latest schedule time by taking into consideration the start time of tasks. This method assigns priority to tasks based on the level of the tasks. Low level tasks will have high priority and high level tasks will have low priority.

Hassan and Abdullah [6] proposed a novel semantic-based scalable decentralized grid Resource Discovery (RD) framework to achieve an effective resource discovery in grid computing. Grid RD framework is built by integrating ontology, Peer-to-Peer network and intelligent agents with two aspects

called description of the resource information and resource information.

Hsu *et al.* [7] proposed an online scheduling approach for multiple mixed-parallel workflows in grid environments. Online Workflow Management (OWM) includes four processes called Critical Path Workflow Scheduling (CPWS), Task Scheduling, multi-processor task rearrangement and Adaptive Allocation (AA). It also includes three data structures called online workflows, a grid environment and a waiting queue similar to offline workflow scheduling [19].

3. Workflow Scheduling in Grid

The proposed algorithm for workflow scheduling employs the Grid-Architecture model considering resource organization and networking. Grid-Architecture integrates distributed resource brokering and allocation services as part of a cooperative resource sharing environment. The Grid-Architecture, $G_A = \{ R_1, R_2, ..., R_n\}$ contains many grid sites, n, with each grid site providing its resource to the framework. Every grid site in the architecture has its own resource defining R_i which consists of the description of the resource, it is willing to provide. R_i can involve information about the number of processors, CPU architecture, operating system type, memory size, secondary storage size, etc.,

In this research, $R_i = \{p_i, a_i, s_i, o_i\}$, which involves processors denoted p_i, processor architecture denoted a_i, their speed denoted s_i, and installed operating system type denoted o_i. Resource brokering, indexing and allocation in Grid-Architecture are facilitated by a Resource Management System (RMS) known as Grid-Architecture Model (GAM). Figure 1 shows an instance of Grid-Architecture resource distributing model containing of Internet-wide shared parallel resources. Every contributing grid site manages its own service represented as three entities: Grid Resource Manager (GRM), Resource Manager (RM) and Distributed Information Manager (DIM) or Grid. The Grid Resource manager is responsible for scheduling locally submitted workflows.

Resource Manager performs other activities to facilitate federation wide job submission and migration process like answering GRM queries related to local job queue length, expected response time, and current resource utilization status. Here, it considers the scientific workflow applications as a case study for the proposed scheduling approach. A Scientific workflow application can be modeled as a Directed Acyclic Graph (DAG), where tasks in the workflow are denoted as nodes in the graph and the dependencies among tasks are denoted as directed arcs among nodes.

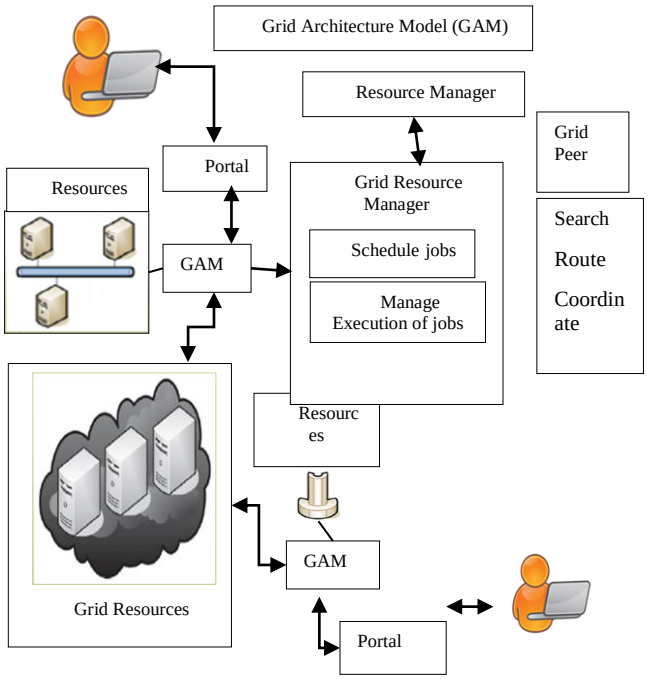

Figure 1. Grid Architecture Model (GAM).

We focus on scheduling workflow application, which has a collection of tasks. Our technique maintains allocation of various tasks in a workflow Management System in the Grid-Framework (shown in Figure 2), if the total number of processors demand for implementing all tasks in a workflow are not available in a single Grid site. In our model, every task needs accessibility to one processor within a Grid site.

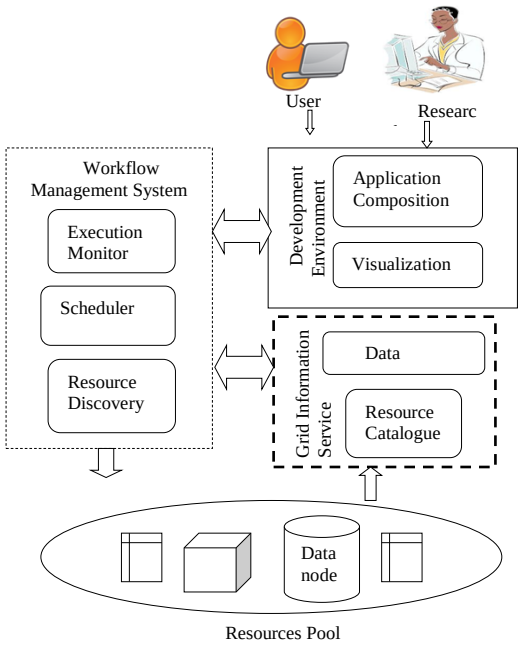

Figure 2. Workflow management system.

If at any time no resource is able to suggest a single processor as requested by a resource maintain object, then the declared object is stored in the coordination space and kept pending and when one of the Grid sites publishes a resource permit and contributes one

available processor. Sites of grid resource permit the following in certain time interval approaches for

1. Task scheduling.
2. Resource provisioning.
3. Resource coordination as specified in the paper [2, 11].

4. Grid Workflow Scheduling based on QoS Constraints

The completion time and resource cost are important dual constraints in QoS to implement workflows on "pay-per-use" services. Users usually would like to get the tasks completed at minimum costs within available time. In this section we present Network performance aware QoS scheduling method for workflow and the algorithm that allows the workflow system to minimize completion time while delivering results within the deadline.

4.1. Analysis and Assessment of Evidence Gathered

This model represents an efficient workflow application as a Directed Acyclic Graph (DAG). Let G = <V, E, W, D> be a DAG is a node-weighted, time constraint and edge-weighted directed graph, where V =<n_1, n_2, n_3... n_n> is the set of task nodes, with each node denoting a task, E is the set of weighted edges representing precedence constraints between nodes in V. Let *D* be the user defined deadline for the execution of workflows.

In this workflow, we call a task an entry with exit task being represented as V_{entry} and V_{exit}. The entry task is defined as a task which lacks a parent. Similarly the exit task represents a task that lacks a child.

4.2. QoS Constraints for DAG

4.2.1. Failure Rate Analysis

The weight on each edge $E_{ij} \in$ W, represents the amount of data being transmitted from task node n_i to task node n_j. Grid resources are represented as R = {R_1, R_2 . . . R_M}. For each task $n_i \in$ V, the weight on each node, T(n_i), represents execution time on each resource node: T(n_i) = {t_1(i),t_2(i),. . . ,t_M(i)}, where t_j(i) represents execution time of n_i on R_j. d_{ij} denotes network delay from R_i to R_j, namely network bandwidth. Let b_{ij} be a binary number that represents whether i^{th} task is assigned to j^{th} resource, 1 for assigned and 0 for not assigned. Let ps_{ij} be the probability of j^{th} resource node not failing during the running of i^{th} task on j^{th} node.

The probability of the system not to fail is expressed in Equation (1).

$$P_r = \xi \prod_{j=1}^{M} \prod_{i=1}^{N} (ps_{ij}^{b_{ij}(t_j(i)+T_{iat})})$$
$$T_{lat} = \sum_{p \in prec(i)} \sum_{k=1}^{M} (E_{pi} \times b_{pk} \times d_{kj}) + SL_j \quad (1)$$

T_{lat} denotes execution latency of task n_i, including the time that task n_i spends to fetch needed data from the preceding nodes and the scheduling length of resource node $P_j(SL_j)$. prec(i) represented as p is a set of immediate precursor of task i. k represents the p^{th} precursors on k^{th} resource. d_{kj} is the delay between node k and j. ξ is the fix up parameter used to modify it based on the theoretical result of Pr. It is experimentally set in the range of (0.1 - 1.0).

The Failure Rate (FR) is defined as the product of resource failure rate and the execution time of the task as follows:

$$FR_{ij} = (1 - ps_{ij}) \times (t_j(i) + T_{lat}) \quad (2)$$

4.2.2. Stability Analysis

Stability is defined as the variability in the performance of a service. α is a computational unit, measured in mips. $\alpha_{avg,i,j}$ is the observed average performance of the task i who leased resource j, $\alpha_{s1a,i,j}$ is the promised values in the SLA, ST is service time and n the total number of tasks. For computational resources, it is the deviation from the performance specified in SLAs.

$$S_{ij} = \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{\alpha_{avg,ij} - \alpha_{sla,ij}}{\frac{ST}{n}} \quad (3)$$

4.2.3. Throughput Analysis

Throughput is the number of tasks executed by the resources per unit of time. It is slightly different from the Service Response Time metric, which measures how fast the resource is provided. $t_j(i)$ represents the execution time of n_i on R_j and n is the total number of tasks.

$$T_{ij} = \frac{n}{t_j(i) + d_{ij}} \quad (4)$$

4.2.4. Efficiency Analysis

Grid system efficiency indicates the effective utilization of leased resources. So, a higher value for efficiency indicates that the overhead will be smaller.

$$Ef_{ij} = \frac{t_j(i)}{t_j(i) + d_{ij}} \quad (5)$$

4.3. A Novel Network Aware QoS Scheduling for Grid Workflow Services

Our proposed scheduling algorithm is based on the Dependable Grid Workflow Scheduling (DGWS) [14] and consists of four phases: priority, partitioning, deadline constraints and scheduling. The sample

workflow DAG graph is represented in Figure 3. The proposed method is presented in Algorithm1.

- *Priority Phase*: In the first phase, a weight is allocated to every node and edge of DAG; based on the mean of all feasible values of the cost of a node (or edge, correspondingly) on every resource (or grouping of resources correspondingly). With this weight, upward priority value is calculated and each node of the DAG is assigned a priority value. The upward priority value of node *i*, $priority_u$ (n_i), is recursively defined according to Equation (6).

$$priority_u(n_i) = \bar{a}_i + \max_{n_k \in succ(n_j)} (1/\bar{d} \times E_{ik} + priority_u(n_k))$$
$$\bar{a}_i = \sum_{j=1}^{M} \frac{t_j(i)}{M} \quad (6)$$

Where

- $\bar{d} = \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{d_{ij}}{(2 \times C_M^2)}$
- $C_M^2 = \frac{M!}{M!(M-2)!}$

Where $\bar{a}_i$ is the mean weight of task node i, succ (n_i) is a set of instant successors of task node i and $\bar{d}$ the mean of network delay among any two nodes.

- *Partitioning Phase*: Next Phase, based on the priority value ($priority_u(n_i)$) of the task nodes of the DAG is sorted in descending order. By means of this order, they are separated into different partitions as follows. The initial node is added to a first partition. If there is dependency, a new partition is created and the new partition's number is the current partition is increased by one, and then the node with the least priority value is the element of the new partition. The final outcome is a set of ordered partitions.

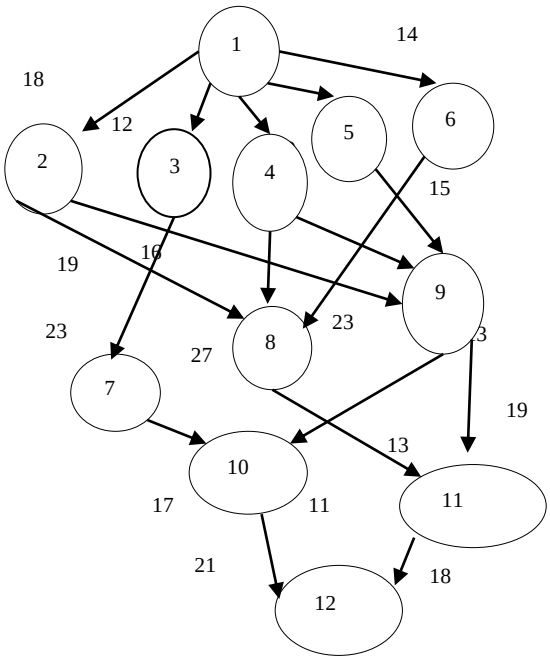

Figure 3. A sample workflow DAG graph.

- *Deadline Constraints Phase*: After task partitioning in the DAG, we allocate the overall deadline among every n_i in G. The deadline d[n_i] allocated to any n_i is a sub-deadline of the overall deadline *D*. This paper assumes the following deadline allocation methods:

- *Method 1*: The collective sub-deadline of any independent path among two dependent tasks should be equal. A dependent task cannot be implemented till all tasks in its parent task partitions are executed. Consequently, instead of waiting for other independent paths to be executed, a path capable of being completed in advance is executed on slower but cheaper services.
- *Method 2*: The collective sub-deadline of any path from $n_i(V_{entry} \in n_i)$ to $n_j(V_{exit} \in n_j)$ is equivalent to the overall deadline D. This method confirms that once each task partition is estimated within its allocated deadline, the entire workflow implementation assures the user's defined deadline.
- *Method 3*: Any allocated sub-deadline should be greater than or equal to the minimum processing time of the consequent task partition. If the allocated sub-deadline is less than the minimum processing time of a task partition, its anticipated execution time will beyond the capacity, its execution services can handle.
- *Method 4*: The whole deadline is separated over task partitions are part of their minimum processing time.

- *Scheduling Phase*: The scheduling phase follows in two approaches:

1. Without partitioning the workflow.
2. With partitioning the workflow.

4.3.1. Without Partitioning the Workflow

In this approach all tasks in the workflow are mapped to a trust rate to resources to minimize the execution of the whole workflow. In this scheduling, the algorithm is to develop workflow schedule based on the QoS constraints like time, cost, reliability, stability, throughput and efficiency.

4.3.2. With Partitioning the Workflow

The scheduling phase makes an efficient schedule for advance reservation and dynamic execution. The advanced reservation and dynamic execution methods are explained in [12, 21]. The schedule assigns each workflow task to a particular service so that they meet users' deadline at low execution cost. We resolve the workflow scheduling problem by separating the entire problem into several task partition scheduling problems. Once every task partition has its hold sub-deadline, it is capable of discovering a local best schedule for every task partition. If every local

schedule promises that their task execution will be finished within their sub-deadline, the entire workflow execution will be executed within the overall deadline. Task in resources, if any partition is predicted which does not meet the deadline it will mean rescheduling the partitioned task with a new possible sub partition as shown in Algorithm 2. Likewise, the outcome of the cost minimization solution for every task partition leads to an optimized cost solution for the whole workflow. Consequently, an efficient workflow schedule can be simply constructed by local best schedules.

Finally, following the ascending order of partitions' number with deadline, tasks within every partition are scheduled. In our method, tasks in every partition are scheduled based on the trust rate of the resource. But, if any partition is predicted which does not meet the deadline it will mean rescheduling the partitioned task with a new possible sub partition. Then it reschedules the sub partition based on the trust rate of resources.

Consequently, we chose the method with a successful task execution is lowest and schedule the task to it. This way, we minimize execution time and cost of workflow.

Algorithm 1: QoS scheduling algorithm.

Input: A workflow G = <V, E, W, D> graph
Output: Scheduling the task in workflow
1. Allocate Weight value for V and E over $\forall V_i \in G$ *and* $\forall E_{ij} \in W$
2. for all $V_i \in G$ *do*
3. Calculate $priority_u(n_i)$ *using (6)*
4. V_i *are sorted in descending order of* $priority_u(n_i)$
5. Calculate FR_{ij}, S_{ij}, T_{ij}, EF_{ij} *using (2) (3) (4) (5)*
6. Compute Trust Rate of Task i on Resource j on

$$TR_{ij} = \sum_{j=1}^{M} \sum_{i-1i=1}^{N} b_{ij}((S_{ij} \times T_{ij} \times EF_{ij})/FR_{ij}) \quad (7)$$

$$\text{Max}\ (\textstyle\sum_{j=1}^{M}\sum_{i=1}^{N} b_{ij}((S_{ij} \times T_{ij} \times EF_{ij}) / FR_{ij})\ )$$

7. Partitioning nodes considering the descending order of their upward priority
Group $(n_{i..j-1}) \in priority(n_{i...j-1}) < priority(n_{j...k-1})$ *i=1...p-1, j= p...q-1, k=q…r-1 for different p,q,r form different group* $(Q=\{P_1,P_2,\ldots,P_n\})$
8. for all $P_i \in Q$ *do*
9. Compute TR_{ij}
10. if (min(completion_time(Q) < min (completion time (G)) then
11. Schedule for i based on Q
12. else
13. Schedule for i based on G.
14. dl[i] ← *get expected completion time of i*
15. if $(dl[i] \geq D[i])$ *then*
16. Reschedule Pi with new possible sub partition

Algorithm 2: QoS rescheduling algorithm

Input: A task partition graph G=<V, E, W, D> delayed partitioned task P_i
Output: a new schedule for unexecuted tasks in the workflow
1. *for all* $i \in P$
2. *Calculate* $priority_u(P_i)$ *using (6)*
3. *End for*
4. P_i *are sorted in the descending order of* $priority_u(P_i)$
5. *Partitioning nodes* (P_i) *into sub partition* (SP_i) *by considering their upward priority*
6. *for all* $i \in SP$
7. *Compute* TR_{ij} *using (7)*
8. *End for*
9. *Schedule task with* $maxTR_{ij}$
Repeat the process till all tasks are completed in the workflow.

5. Results and Discussion

Experimental grid setup was performed using gridsim by which we created the 4 Grid Information Service (GIS),10 users, 10 jobs, 27 resources with different bandwidth and 2 routers (Table 1). Each resource runs multiple tasks.

Table 1. The router Initialization.

Router_Name	Baud_rate (GB/s)	Prop_delay (ms)	Maximum transmission unit (mtu(byte))
Router 0	1	10.0	1500
Router 2	1	10.0	1500

The following are the graphical results of the implemented workflow schedule namely Network Performance Aware QoS workflow scheduling algorithm for Grid Services and parameters considered for the comparison of the methods include:

- Makespan.
- Execution Cost.
- Throughput.
- Average Success Ratio.

Makespan is calculated as the execution time of the entire workflow, which equals the variation between start time of V_{entry} in the workflow and output occurrence time of V_{exit} in that workflow as shown in Figure 4. In the graph, the amount of tasks varies from 50 to 500 along the x-axis and average makespan workflow is in using along y-axis various from 0 to 5000.

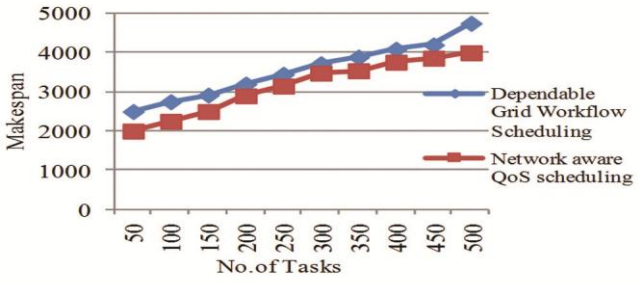

Figure 4. Makespan graph.

It can be conditional from the graph that makespan of Network aware QoS scheduling algorithm is lesser than Dependable Grid Workflow Scheduling. Network

aware QoS scheduling algorithm for workflow is 11.80% faster than Dependable Grid Workflow Scheduling.

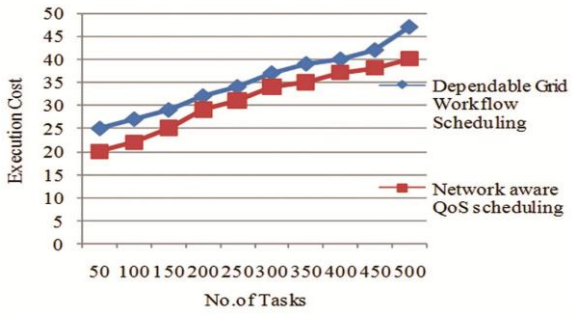

Figure 5. Execution cost graph.

The costs related to a specific resource can differ depending on the time taken when the resource is managed. By performing a complete analysis of the resource costs through one or more particular periods, we decide the optimum times for scheduling activities that involve the resource. Figure 5 represents the cost of resource utilized. In Figure 5, the number of tasks ranging from 50 to 500 is taken along x-axis and execution cost workflow is taken along the y-axis ranging from 0 to 100. In the graph the cost of Network aware QoS scheduling algorithm for workflow is 12.28% lesser than Dependable Grid Workflow Scheduling.

Throughput refers to the performance of tasks by a computing service or device over a specific period. It measures the amount of completed work against the time consumed and may be used to measure the performance of a processor, memory and/or network communications. Throughput is measured by calculating the amount of data transferred between locations during a specified period, generally resulting as Kilo Bytes Per Second (KBPS).

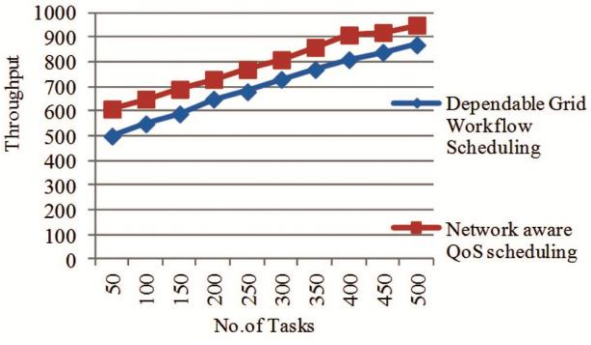

Figure 6. Throughput Graph.

Figure 6 represents the overall throughput calculation where the number of tasks ranging from 50 to 500 is taken along x-axis and throughput is taken along y-axis ranging from 0 to 1000. The graph shows that throughput of Network aware QoS scheduling algorithm is 11.76% higher than the Dependable Grid Workflow Scheduling.

Success Ratio is defined as the ratio of the number of successful tasks to the number of all tasks. Figure 7 represents the average success ratio of resources where the number of tasks ranging from 50 to 500 is taken along x-axis and average success ratio is taken along y-axis ranging from 0 to 100.

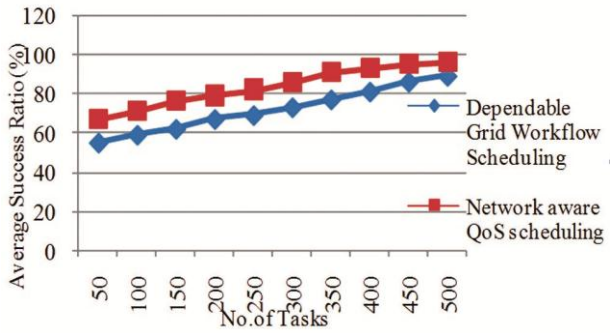

Figure 7. Average success ratio graph.

From the results of the graph the success ratio of Network aware QoS scheduling algorithm is 17.06% higher than Dependable Grid Workflow Scheduling.

6. Conclusions and Future Work

In existing, grid workflow system has not addressed the high dynamic feature and dependable workflow scheduling. The present work proposes an efficient scheduling mechanism by first analysing the DAG and scheduling a workflow management system based on QoS. The proposed framework presents, an innovative model known as Network Performance aware QoS based Workflow scheduling for Grid Services that reduces the cost of task completion while meeting the deadline. Based on our work, a Network Performance Aware QoS Workflow scheduling algorithm is implemented, which considers QoS constraints and the workflow with and without the partition method. For the workflow the priority values are calculated for each task on a node, based on priority value. The work flow is partitioned into groups. The user can specify the deadline for each task in the partition. The resources are selected based on minimum resource cost and meet the deadline of each task. We also described task partitioning, deadline constraints and resource availability for efficient scheduling of task execution.

Our algorithm improves the success ratio of tasks and throughput of resources and reduces makespan and the execution cost of workflow. In future application based QoS parameters may be developed to obtain more accurate resource selection on user requirements.

References

[1] Abrishami S., Naghibzadeh M., and Epema D., "Cost-Driven Scheduling of Grid Workflows Using Partial Critical Paths," *IEEE Transactions*

on Parallel and Distributed Systems, vol. 23, no. 8, pp. 1400-1414, 2012.

[2] Abrishami S. and Naghibzadeh M., "Deadline-Constrained Workflow Scheduling in Software as A Service Cloud," *Scientia Iranica*, vol. 19, no. 3, pp. 680-689, 2012.

[3] Amalarethinam G. and Selvi K., "An Efficient Dual Objective Grid Workflow Scheduling Algorithm," *International Journal of Computer Applications*, vol. 33, no. 1, pp. 7-12, 2011.

[4] Gharooni-Fard G., Moein-Darbari F., Deldari H., and Morvaridi A., "Scheduling of Scientific Workflows using AChaos- Genetic Algorithm," *in Proceedings of International Conference on Computational Science*, Netherlands, pp. 1439-1448, 2010.

[5] Hasham K., Peris A., Anjum A., Evans D., Hufnagel D., Huedo E., Hernández J., McClatchey R., Gowdy S., and Metson S., "CMS Workflow Execution using Intelligent Job Scheduling and Data Access Strategies," *IEEE Transactions on Nuclear Science*, vol. 58, no. 3, pp. 1221-1232, 2011.

[6] Hassan M. and Abdullah A., "A New Grid Resource Discovery Framework," *The International Arab Journal of Information Technology*, vol. 8, no. 1, pp. 99-107, 2011.

[7] Hsu C., Huang K., and Wang F., "Online Scheduling of Workflow Applications in Grid Environments," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 860-870, 2011.

[8] Ijaz S., Munir E., Anwar W., and Nasir W., "Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment," *The International Arab Journal of Information Technology*, vol. 10, no. 5, pp. 486-492, 2013.

[9] Nadia R. and Zimeo E., "Time and Cost-Driven Scheduling of Data Parallel Tasks in Grid Workflows," *IEEE Systems Journal*, vol. 3, no. 1, pp. 104-120, 2009.

[10] Rahman M., Hassan R., Ranjan R., and Buyya R., "Adaptive Workflow Scheduling for Dynamic Grid and Cloud Computing Environment," *Concurrency Computation Practice Experience*, vol. 25, no. 13, pp. 1816-1842, 2013.

[11] Rahman M., Ranjan R., and Buyya R., "Cooperative and Decentralized Workflow Scheduling in Global Grids," *Future Generation Computer Systems*, vol. 26, no. 5, pp. 753-768, 2010.

[12] Smith W., Foster I., and Taylor V., "Scheduling with Advanced Reservations," *in Proceedings of International Parallel and Distributed Processing Symposium*, Cancun, pp. 127-132, 2000.

[13] Su S., Li J., Huang Q., Huang X., Shuang K., and Wang J., "Cost-Efficient Task Scheduling for Executing Large Programs in the Cloud," *Parallel Computing*, vol. 39, no. 4-5, pp. 177-188, 2013.

[14] Tao Y., Jin H., Wu S., Shi X., and Shi L., "Dependable Grid Workflow Scheduling Based on Resource Availability," *Journal of Grid Computing*, vol. 11, no. 1, pp. 47-61, 2013.

[15] Vasques J. and Veiga L., "A Decentralized Utility-Based Grid Scheduling Algorithm," *in Proceedings of the 28th Annual ACM Symposium on Applied Computing*, Coimbra, pp. 619-624, 2013.

[16] Wu Q., Zhu M., Gu Y., Brown P., Lu X., Lin W., and Liu Y., "A Distributed Workflow Management System with Case Study of Real-Life Scientific Applications on Grids," *Journal of Grid Computing*, vol. 10, no. 3, pp. 367-393, 2012.

[17] Wu X., Deng M., Zhang R., Zeng B., and Zhou S., "A Task Scheduling Algorithm Based on QoS-Driven in Cloud Computing," *in Proceedings of ITQM in Elsevier*, China, pp. 1162-1169, 2013.

[18] Yousaf M. and Welzl M., "Network-Aware HEFT Scheduling for Grid," *The Scientific World Journal*, vol. 2014, pp. 1-13, 2014.

[19] Yu J., Buyya R., and Ramamohanarao K., *Meta-Heuristics for Scheduling in Distributed Computing Environments*, Springer, 2008.

[20] Yu J., Buyya R., and Tham C., "QoS-based Scheduling of Workflow Applications on Service Grids," *in Proceedings of the 1st IEEE International Conference on E-Science and Grid Computing*, Australia, pp. 1-9, 2005.

[21] Zhao H. and Sakellariou R., "Advance Reservation Policies for Workflows," *in Proceedings of Job Scheduling Strategies for Parallel Processing*, Saint-Malo, pp. 47-67, 2007.

Shinu John is a Professor in the department of Computer Science and Engineering at the St. Thomas College of Engineering and Technology, Kannur, India. He obtained his Ph. D. from Anna University, Chennai. He received his M.E. and B.E. degrees in Computer Science and Engineering from Anna University, India and the M.S. University, Tirunelveli, India respectively. He is a member of the System Software Group at MAM College of Engineering, India and has published many papers in various national, international journals and conferences. His research interests include Grid Computing, Mobile Computing and Computer Networks. He is a life member of Computer Society of India, the Indian Society for Technical Education (ISTE), Institution of Engineers and a member of IEEE since 2006.

Maluk Mohamed obtained his Ph.D. from the Indian Institute of Technology (IIT) Madras in 2006, Masters in Engineering from the National Institute of Technology, Tiruchirappalli in 1995 and Bachelors from the Bharathidasan University in 1993. He is currently a professor in the Department of Computer Science and Engineering, M.A.M. College of Engineering, India. He coordinates research activities for the System Software Group at MAMCE. His research interests include distributed computing and its family ie; grid computing, mobile computing, cloud computing and wireless sensor networks, software engineering and distributed databases. He has guided 5 Ph. D, 1 M.S. (By research) and 32 M. Tech., scholars and is currently guiding 5 Ph.D., and 3 M. Tech., research scholars. He is a member of the Board of Studies, in Anna University and JNTU Anantapur. He is the principal investigator for a number of funded projects like Cyberspace Security and Cloud API.