Network and Data Location Aware Job Scheduling in Grid: Improvement to GridWay Metascheduler

Saumesh Kumar and Naveen Kumar

Indian Institute of Technology, Roorkee, India
saumeshkumar@gmail.com, navincse2005@gmail.com

Abstract

Grid Computing has enabled us to utilize the unused computing power (CPU cycles) of computers connected to networks (e.g. Internet). Nowadays, there are lots of scientific projects going on in the domain of High Energy Physics (HEP) and Grid infrastructure constitutes the core computing facility of these projects. One such project is LHC (Large Hadron Collider) deployed at CERN. These experiments produce and manage a large amount of data per day and run thousands of computing jobs to process that data. The applications for these experiments require large data transfers over the network from data sources to computing resources. It is the duty of meta-scheduler to allocate jobs to most appropriate resources, and to use network in an efficient way. In this work, a Network and Data Location Aware job scheduling has been proposed for data intensive jobs. The proposed scheduling algorithm takes into account network characteristics, disk read speed of data sources, and data locations of input files, as well as other computational factors (CPU power, memory, CPU load, e.t.c) when making scheduling decisions. This scheduling algorithm aims to minimize not only file staging (data transfer) time but also turnaround time of the jobs. There are extensions to GridWay that consider the network state when making scheduling decisions, but they have not used network information in an efficient way, and have not considered data locations of input files either. Thus, they are not efficient for data intensive jobs which require huge data movement over the network. The authors have improved the GridWay MetaScheduler with Network and Data Location Aware scheduling algorithm. The improved GridWay MetaScheduler has been tested for data intensive jobs. Results presented here shows that the data transfer time and turnaround time of jobs are reduced when network characteristics, data locations of input files, and disk read speed of storage drive at data sources are considered in the jobs scheduling process.

Keywords*: Grid Computing, Job Scheduling, Network and Data Location, Data Intensive Jobs, GridWay Metascheduler, Globus*

1. Introduction

Grid systems are highly distributed and dynamic in nature because of high degree of heterogeneity of jobs and resources. Grids are made of geographically distributed and independent virtual organizations that share their resources. In Grid computing environment, computing resources are under different administrative domain having their own access policy, local resource management system and other constraints. Grid technology has emerged as an enabling technology for many data and/or computing intensive applications. Through the use of Grid technologies it is possible to aggregate geographically dispersed heterogeneous resources for solving large-scale parallel applications in science, engineering

and commerce [1]. In Grid computing, individual users can access computers and data, transparently, without having to consider their location, operating system, account administration, and other details.

The goal of Grid computing is to create the illusion of a simple but large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. Job scheduling is one of the most complicated and kernel component of a grid system [4]. Unlike scheduling problems in conventional distributed systems, this problem is much more complex as new features of grid system such as its dynamic nature and the high degree of heterogeneity of jobs and resources must be tackled [5].

Grid resource management systems are most important components of grid systems. Resource managers such as batch schedulers, workflow engines, and operating systems exist for many computing environments. These resource management systems are designed to work under the assumption that they have complete control of a resource and thus can implement the mechanisms and policies needed for the effective use of that resource. Unfortunately, this assumption does not apply to the Grid. When dealing with the Grid, we must develop methods for managing Grid resources across separately administered domains, with resource heterogeneity, loss of absolute control, and inevitable differences in policy that is the result of heterogeneity. The scheduler needs to manage resources and jobs execution depending on the objectives and the requirements of the providers (resource owners) and consumers (resource users), and they need to continuously adapt to changes in the availability of resources. This requirement introduces a number of challenging issues that need to be addressed; namely, site autonomy, heterogeneous substrate, policy extensibility, resource allocation or co-allocation, online control and quality of service based scheduling.

Foster et. al. [1] defines the responsibilities of a resource management system on a grid as the "discovery of available resources for an application, mapping the resources to the application subject to some performance goals and scheduling policies and loading application to the resource in accordance with the best available schedule". Grid resource management systems also provide the functionality of job monitoring as well as their scheduling and submission. It manages resources constantly while dynamically updating the grid scheduler whenever there is a change in resource availability. The development or adaption of application for grid environment is being challenged by the need of scheduling a large number of jobs to resources efficiently. The scheduler is responsible to allocate the set of tasks/ jobs to available compute resources in accordance with the best available schedule.

The job scheduling problem in grid can be viewed as to arrange the pairs of jobs & resources (that is to obtain a resources scheduling sequence), which needs to consider some constraints. Its objectives are to minimize and/or maximize one or more parameters of the multi-dimensional QoS metrics. Some of these QoS parameters are followings:

- Minimizing jobs' completion time and execution cost.
- Minimizing data movement.
- Maximizing system throughput and resources' utilization.

Effective job scheduling in grid requires modeling of available resources on the grid nodes and computation requests of jobs, determining the current load on the system and predicts the job execution time. According to Casavant et. al. [2], the scheduling problem consists of three main components: consumers, resources and policies. The policy may specify, at the implementation level, the method of mapping jobs to resources. The policy chosen for implementing a scheduler affects both users and resource providers.

From the viewpoint of resources, a grid is composed of a set of resources and domain resource managers (DRMs), also called local resource management systems (LRMS). Each DRM coordinates and controls its local resources; the basic functionality of such local resource managers is load balancing at that resource. Currently many users directly submit jobs to local scheduler or indirectly do the same thing via Globus grid interfaces (e.g Globus Command Line Interface or Globus APIs). Users have to make decisions on their own about what kind of resources is suitable for their jobs and if a job fails, they have to manually resubmit it. Obviously, in a large-scale, dynamic and heterogeneous grid environment where various types of resource managers exist for managing their local computing resources; this is not an efficient way to manage job executions. So a global resource manager/scheduler (Meta-Scheduler) is needed to manage these local schedulers and appropriately negotiate access to different resources managed by different local schedulers, thus releasing the burden of job execution from the end-users. The meta-scheduler provides end-users with a single entry point to grid computing environment. It coordinates communications between multiple heterogeneous local schedulers. To schedule an individual job, the meta-scheduler typically queries information about available compute resources and their status from Grid Information System (e.g MDS of Globus Toolkit [6]), after determining which resource is suitable for the job, it dispatches the job to the desired resource or resources, where the job will be rescheduled by the local scheduler to a specific computing element (CE) within that resource.

In many scientific domains particularly High Energy Physics (HEP), experiments produce huge amount of the data that need to be processed and analyzed. Grid technology is best suited to develop applications for these kinds of experiments which require handling of huge data. These applications are characterized by high needs of data sharing and/or computing power. And it is grid technology which allow to aggregate unused computing power (CPU cycles) of geographically dispersed resources. It is the duty of the scheduler to allocate the jobs to the most appropriate resources. Anjum et. al. [7] had shown that the scheduling algorithms that focus only on maximizing resource utilization by mapping jobs to idle resources and disregard network cost, and cost associated with accessing remote data are unlikely to be efficient. Similarly, the scheduling decisions which force jobs movement towards the data without taking network load into consideration can lead to inefficiencies in performance, and can be responsible for large job queues and processing delay. L. Thomas et. al. [11] has also shown that the network characteristics must also be taken into account when making scheduling decisions to achieve optimal schedule. Most of the current meta-schedulers (like Nimgrod/G [9], Condor/G [10]) consider computing power and utilization of resources, budget (economy), and other factors, in their scheduling metrics. But, none of the existing scheduler takes into account network characteristics and data locations of input files when making scheduling decisions. According to L. Thomas et al. [11] and A. Anjum et al. [7], schedulers that do not consider network characteristics when making scheduling decisions might produce the schedule that is not optimal. For example, scheduler might decide that the most powerful or unloaded resources are best suited to run user's job. However, if there is some overloaded link in the path between data sources (data locations) and computing resource, a better choice would be to run the job on other less powerful resource with less loaded network. The scheduling algorithms that focus only on maximizing resources utilization by directing jobs to more powerful and lightly loaded resources are unlikely to be efficient. Simulations performed by Ranganathan and Foster [14] using a Data Grid simulation system, ChicSim, indicated that data location has to be taken into account as well. Therefore, network characteristics and data location of input files must also be considered when making scheduling decisions.

In recent years, technologies in high speed data communication like fiber-optics networks enable us to transfer data at rate of several hundred of Gbps, and high speed networks (e.g., Gigabit Ethernet) provides data transfer rates of 1 Gbps or more. In such high speed networks, the bottleneck to data intensive applications performance is not due to available network bandwidth, but because of data transfer rate of storage drive (like HDD) of data sources, because SATA (Serial Advanced Technology Attachment) interface by which storage drives are connected to computer, provides a maximum of 3 to 6 Gbps (theoretically) data transfer rate, practically limited between 60 to 110 MB/sec. So, for such high speed networks data transfer rate of storage drives of data sources must also be considered when making scheduling decisions for data intensive jobs to achieve optimal schedule.

In this paper, we have implemented network and data location aware jobs scheduling algorithm which takes into account network characteristics, data locations of input files, and disk read speed (disk latency) of data sources as well as other computational factors when making scheduling decisions. The GridWay Metascheduler [12] has been improved with the proposed job scheduling algorithm. GridWay Metascheduler is an open source meta-scheduling technology that performs job execution management and resource brokering on heterogeneous and dynamic Grids based on Globus Toolkit services. It enables large-scale, reliable and efficient sharing of resources, managed by different Local Resource Management (LRM) systems, such as PBS, LSF or Condor. GridWay Metascheduler is a part of Globus Toolkit [15] distribution.

2. Related Work

One of the first works considering *Network and/or Data Location Aware Job Scheduling* was proposed by K. Ranganathan and I. Foster in [14]. They propose an algorithm that takes care of replicating popular (most accessed) files in a way that is totally decoupled from the job scheduling itself. Their job allocation strategy consists then in just sending the jobs where the data resides. This approach tries to avoid the complexities related to a combined mechanism for data replication and job scheduling. By asynchronously replicating most demanded data, they expect to distribute workload among sites.

Data location awareness is an important factor to consider when performing job scheduling in grid because of the long time spent transferring data before jobs can actually start the computation. Considering the data location awareness factors, authors of [16] concluded that to reach an optimum scheduling both the data transfer delay and the expected queuing time at the available resource must be considered. However, they did not consider the policies to avoid unnecessary data replication. They also propose economic-based algorithm for selecting the best replica to transfer when the selected resource for a job is not holding the necessary data.

A. Anjum et. al. [7] proposed a Data Intensive and Network Aware (DIANA) meta-scheduling system for grid environment. For each job to be scheduled a handful of computing nodes are selected according to their computing power, queuing jobs and delay associated with data transfer. DIANA takes into account data, processing power and network characteristics when making scheduling decisions across multiple sites. The job scheduler provides global ranking of the computing resources and then selects an optimal one on the basis of the overall access and execution cost. Once the job's destination (computing node) has been chosen, DIANA selects the best replicas of the input files that need to be transferred to that node.

Scheduling decision of most of the existing grid schedulers (Condor/G[10], Nimgrod/G[9], GridWay [12] e.t.c) considers only some of the following scheduling parameters:

- computing power and memory of resources,
- computing resource utilization,
- jobs waiting and turnaround time,
- budget (economy) and deadline.

But none of the scheduler considers network characteristics, data locations of input files, and disk read speed of storage drives at data sources in their scheduling decision metric.

2.1. Improvements to GridWay Metascheduler in Literature

E. Huedo et. al. [8] presented a scheduling algorithm which aims to minimize not only job turnaround time but also data replication. The authors had also addressed the shortcomings of DIANA [7] and proposed a solution for uncontrolled data replication. Data Intensive applications involve the processing of huge amount of input data of grid jobs. If those jobs were just sent to most powerful computing resource available, regardless of the location of their required data, costly transfers would have to be performed in order to move data to computing site where jobs are scheduled to run. Authors had showed that optimal scheduling can be achieved by considering data transfer delay, job waiting and execution times. It is sometimes more convenient for a job to wait for transfer of the input data than to wait for a busy computing resource to offer a free slot where to run. However these approaches successfully optimized job efficiency, they would in general not minimize data replication. To solve the shortcoming of uncontrolled data replication, authors defined several new functions for considering data location in requirement expression of jobs and rank expressions of computing resources specified in job template file for GridWay Metascheduler.

L. Thomas et. al. [11, 17] presented a working implementation of a meta-scheduler which integrates network information when taking scheduling decisions. It was devised by extending features of the GridWay MetaScheduler. The authors presented a QoS brokering service to provide guaranteed reservation of network capacity along with computational resources for real-time applications. But this brokering service does not focus on input data locations. They have presented an entity called Grid Network Broker (GNB) and used Bandwidth Broker (BB) for reserving bandwidth between computing nodes. The monitoring of network and resources is carried out with a given frequency, called monitoring interval. As the GNB performs scheduling of jobs to computing resources, the effective bandwidth of individual monitoring intervals is updated independently, so that it is reflected to jobs that are being submitted to resources. Authors have used a network monitoring tool called *Iperf* [18] to monitor the bandwidth.

2.2. Inefficiencies in Existing Improvements to GridWay

It is observed from previous sections that existing improvements to scheduling algorithm of the GridWay suffers with following deficiencies:

- The scheduling algorithm presented by E. Huedo et. al. [8] only tackles the issues of the uncontrolled data replication. But the presented algorithm does not have the ability to estimate the time that transfer of given input files between two nodes will take. It does not consider the network state when making scheduling decisions.
- L. Thomas et. al. [11,17] has presented a scheduling algorithm which considers network characteristics when making scheduling decisions. But the scheduling

algorithm does not takes data locations of input files into account when making scheduling decisions. In the presented algorithm, the available network bandwidth had been measured between the node that submits jobs (that is, the node with GridWay) and every other node in the grid. It is not an efficient approach to measure the available bandwidth between scheduler node (with GridWay) and all other computing node. This might results in large overhead when the number of computing nodes connected to grid is very large. Therefore, efficient method would be to consider only matched resources of the jobs instead of all the computing nodes.

As pointed out earlier, in existing improvements to GridWay, the available bandwidth is measured between scheduler (GridWay) node and all other computing nodes. But instead of measuring available bandwidth between GridWay node and all other computing nodes, it would be more efficient to measure the network bandwidth between data sources of input files and matched resources for a job. Because what we need is to reduce (minimize) data transfer time between data sources and computing resources. So the network characteristics (bandwidth and latency) must be measured data sources and matched candidate computing resources of jobs rather than measuring network characteristics between scheduler and all other computing resources. This would result in less overhead, and an efficient network and data location aware scheduling to GridWay.

3. Network and Data Location Aware Job Scheduling in Grid

The network and data location aware job scheduling algorithm can be viewed as mapping of *n* independent jobs *J = {J1, J2, ..., Jn}* to a set of *m* resources *R = {R1, R2, ..., Rm}* with objectives of minimizing data transfer time of input files and turnaround time of the jobs. To capture the network characteristics, data requirements of jobs, disk read speed of data sources and other scheduling parameters, we have used following arrays/ matrices in our scheduling algorithm: *data_size[n], data_source[n], disk_speed[m], prop_delay[m][m], BW[m][m] and xfr_time[n][m]*. The entries of these arrays are following:

data_size[i] = size (in bytes) of data required for the execution of the job Ji, (1 <= i <= n).

data_source[i] = Rx; data source at which data for job Ji is stored, (1 <= i <= n).

disk_speed[j] = data transfer rate (in bytes per second) of storage drives of resource Rj,(1<=j<=m).

prop_delay[x][y] = propagation delay / network latency (in seconds) from resource Rx to Ry.

BW[x][y] = bottleneck (minimum) bandwidth (in bits per second, bps) from resources Rx to Ry, (1 <= {x, y} <= m).

xfr_time[i][j] = estimated file staging (data transfer) time of job Ji when scheduled on resource Rj.(1 <= i <= n), (1 <= j <= m).

We have assumed that the required data of a job is stored at only single data source. The entries of array *disk_speed[m]* can be obtained by disk utility tools like **hdparm** available in linux systems. The entries in *disk_speed[m]* also include rotational latency and seek time (i.e.

disk latency) as well as transfer rate. The entries of the arrays *prop_delay* and *BW* are obtained by open source utilities like **Iperf**. The matrix *prop_delay[m][m]* is not symmetric because the propagation delay in two direction (i.e. from resource *Rx to Ry* and vice-versa) might not be the same. Similarly, the matrix *BW[m][m]* is also not symmetric because bandwidth in two direction might be different.
The *xfr_time[n][m]* is used to select the candidate resource for which file staging (data transfer) time of jobs is minimum. The entries of *xfr_time[i][x]* are computed as follows:

If *BW[data_source[i]][x] < disk_speed[data_source[i]],*
then
*xfr_time[i][x] = prop_delay[data_source[i]][x] + (data_size[i] * 8) / BW[data_source[i]][x])*
Otherwise
xfr_time[i][x] = prop_delay[data_source[i]][x] + (data_size[i] / disk_speed[data_source[i]])

After computing the estimated data transfer time of each job, the scheduler schedules a job *Ji* to the resource *Rj* for which estimated data transfer time (i.e *xfr_time[i][j])* is minimum and whose rank is also higher than other candidate resources. A resource with higher rank is selected to reduce/ minimize the execution time of the jobs. So the overall turnaround time of the jobs is minimized.

Improvements to GridWay

As we have discussed earlier that for data intensive jobs, input files staging take significant amount of time of jobs completion time. So to reduce the file staging (data transfer) time, scheduler must also take into account data locations of input files and disk read speed (disk latency) of storage drive (e.g., HDD) of data sources along with network characteristics when making scheduling decisions. Therefore, a meta-scheduler must take into account network information, data location of input files, and disk read speed of data sources when making scheduling decisions to achieve optimal scheduling.

In this work GridWay Meta-Scheduler has been improved with the proposed scheduling algorithm. This meta-scheduler is chosen among others mainly due to the availability of its source code, and to the fact that it is a Globus project. And also, its capability to plug in new scheduler without modifying other components of it, because it uses an external and selectable scheduler module.

Existing GridWay MetaScheduler makes its scheduling decisions based on *Scheduling Policies*. These policies are provided by the administrator or user by means of tags in the job template file. These scheduling policies are used to assign a *dispatch priority* to each job and *suitability priority* to each resource. A given job can be executed on those resources that match its requirements. The matching resources are prioritized according to four policies: fixed, usage, failure and rank. The ranks (tag ***RANK***) of the resources are computed according to memory, CPU power and load on CPU. But none of these policies considers the available network bandwidth between data source and computing node, data locations and disk latency for data intensive jobs.

The Job scheduling algorithm of GridWay MetaScheduler has been improved to take into account the following parameters along with its existing computational parameters (i.e. CPU power, free memory, average load) when making scheduling decisions:

- Network characteristics (e.g., latency and bandwidth) between data sources and computing node.
- Data locations of input file for jobs.
- Disk read speed (disk latency) of storage drives at data sources.

Several scheduling parameters are gathered through the Globus GIS module. The modified scheduler of GridWay uses following tools to get the network and disk related information:

- **Iperf** [18] is a network performance monitoring tool. We have used it to capture the real time network characteristics (bandwidth, latency, throughput, e.t.c) between two or more nodes. The network information captured by Iperf is stored in a new parameter tag, called *Bandwidth*. The value of *Bandwidth* tag is then used by scheduler when making scheduling decisions. Iperf makes an estimation of the available bandwidth by sending a series of packets, with a certain size, through the specified port and for a certain time (default 10 sec).
- **Hdparm** [19] is s a command line utility for the Linux and Windows operating systems to set and view ATA hard disk hardware parameters. Disk read speed of data sources is measured by hdparm, and stored in a new parameter tag, called *Disk Read Speed*, which is used to compute the ranks (other parameters are also considered) of the data sources.

The improved GridWay MetaScheduler schedules the jobs by considering the fact whether a job is data intensive or compute intensive or both (data and compute intensive). Compute intensive jobs are usually scheduled to higher rank (more computing power, more memory, lightly loaded e.t.c) resources to minimize the execution time. Whereas, data intensive jobs are scheduled to the resources where data transfer cost as well as execution time is minimized, so the overall turnaround time of jobs is also minimized.

The modified GridWay has three new parameters tags: *Bandwidth*, *Disk Read Speed*, and *Data Location*. The value of these new tags is integrated in the process *RANK* computation of resources. The rank (***RANK***) of candidate resources in the modified GridWay is computed on the basis of following parameters:

- Network latency and bandwidth.
- Data locations of input files of the job.
- Total and free memory available at resource.
- CPU power (in GHz).
- Number of processing cores in resource.
- Average load on CPU.

Tags ***REQUIREMENTS*** and ***RANK*** allow users to specify the criteria used to select most appropriate resource to run their job. These tags are included in job template file. With the ***REQUIREMENTS*** tag, the user can set the minimal requirements needed to run the job, thus applying a filter on all the resources known to GridWay. The priority of the resource is computed as a weighted sum of every individual criteria we have discussed above [11].

The proposed scheduler measures network characteristics (bandwidth and latency) between the matched computing resources and the data source of the job. We have used **Iperf** to measure the bandwidth, and **Ping** to measure the latency between computing nodes and data

sources. This scheduler also uses the disk read speed (disk latency) of storage drive of data sources to estimate the data transfer time of input files of the job. The disk information is essential when the available network bandwidth between matched computing resources and data source of the job is higher than the disk read speed of the data source; because in that case, the bottleneck in data transfer rate would be due to limited disk read speed, not because of the available bandwidth between computing resource and data source. For data intensive jobs, scheduler schedules the job to the resource for which file staging (data transfer) time is minimum, and whose (computing resource) rank is relatively high (i.e. more powerful CPU, more free memory, lightly loaded) than other matched candidate computing resources.
The pseudocode of the proposed algorithm for GridWay is given below:

Algorithm: Network and Data Location Aware scheduling algorithm used in GridWay

1: for all unscheduled jobs
2: *jid* = Select a job from the list of unscheduled jobs
3: Select resources which fulfills the ***REQUIREMENTS*** of job (*jid*)
4: Now select a resource *Rj*, from the list of matched resources (from step 3), for which data transfer time for job *jid* is minimum and whose rank is higher than other matched resources.
5: Finally dispatch job *jid* to the selected resource *Rj*.

4. Experiments

Several experiments have been conducted in order to evaluate the functionality, and to measure the performance improvement experienced by ends users when network characteristics (bandwidth and latency), data locations and disk latency of data source are used in scheduling metrics. The network topology of grid infrastructure is shown in Figure 1. All the resources belong to the same administrative domain, and connected to the institute Local Area Network (LAN).

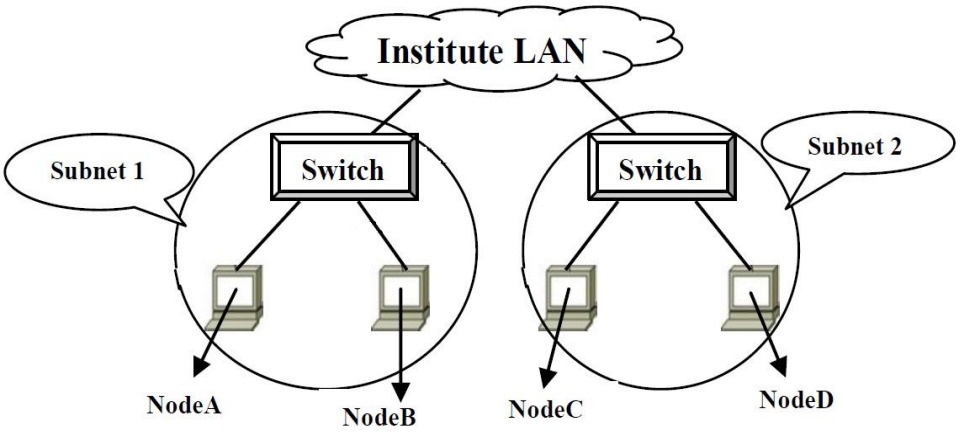

Figure 1. Network Topology of Grid Infrastructure

Testbed Description

The testbed is composed of different computational resources (Workstations and Laptops) interconnected by Ethernet LAN. The configurations of these resources are given in Table 1. There are two Workstations: ***NodeA***.*globus.grid*; ***NodeB***.*globus.grid*, and two laptops: ***NodeC***.*globus.grid*; ***NodeD***.*globus.grid* connected to the grid. As shown in figure 1, these nodes are connected to two different subnets: **Subnet1** and **Subnet2**. NodeA and NodeB are in Subnet1; and NodeC & NodeD are in Subnet2. We have used version 4.0.7 of Globus Toolkit on all resources to create grid. All of these resources are also used as data sources for storing jobs input files. The resources connected to grid have different operating system (OS), creating a heterogeneous grid environment. The GridWay Metascheduler (version 5.6.7) has been installed on **NodeC.** So NodeC carries out the scheduling tasks as well as execution of jobs scheduled to this host. This NodeC is also the central authority of *Virtual Organization* (VO), called VO manager. The VO that has been built on top of the resources, connected to grid, is shown in Figure 2.

Table 1. Configuration of Resources

Machine (FQDN)	CPU	RAM (GB)	HDD (GB)	OS
NodeA.globus.grid	Intel (R) Xeon CPU 3.20 GHz	2	80	CentOS (v. 5.5)
NodeB.globus.grid	AMD Opteron Processor 2.40 GHz	4	320	CentOS (v. 5.5)
NodeC.globus.grid	Intel Core 2 Dou 2.1 GHz	3	360	Ubuntu (v. 9.04)
NodeD.globus.grid	Intel Core 2 Dou 1.7 Ghz	2	320	Fedora (v. 9)

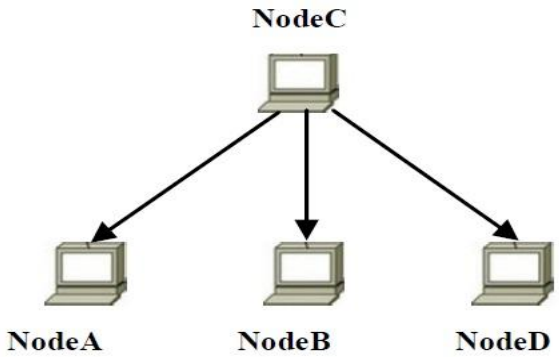

Figure 2. Virtual Organization

5. Results

To validate and evaluate performance results of the implemented network and data location aware job scheduling algorithm different jobs are run. The details of the jobs that we have used to evaluate the performance of the implemented scheduling algorithm are given in table 2. When performing the tests reported here, default parameters of scheduling configuration of

the GridWay have been used. The results of jobs execution presented here are collected from job monitoring command (**gwps**) of GridWay.

In order to evaluate the performance obtained by the improved GridWay meta-scheduler, we have conducted several tests with different jobs (see table 2). The new improved GridWay takes into account the network characteristics (to be precise bandwidth and latency), data locations of input files, and disk read speed of data sources when making scheduling decisions. For evaluating the performance, the load on each host in grid is initially negligible, and these jobs have been run many times.

Following section provides performance comparison between GridWay's default (GWD) and implemented (***N***etwork and ***D***ata location ***A***ware ***Sched***uler, **NDASched**) scheduling algorithm.

Table 2. Jobs Parameters

Job	Input Files Size (MB)	Data Source	Job Type
Job1	713	NodeC.globus.grid	Data Intensive
Job2	713	NodeA.globus.grid	Data Intensive
Job3	980	NodeB.globus.grid	Data Intensive
Job4	0	-	Normal

(A) File staging (data transfer) time: Figure 3 shows the average transfer time of different submitted jobs. It can be seen that data transfer time of jobs under network and data location aware scheduling (NDASched) is reduced. Both the scheduler schedule *Job1* to same host (*NodeC.globus.grid*) because of its rank computed at the time of scheduling, so transfer time is almost same in both scheduling. But for *Job2* and *Job3*, data transfer time in NDASched much less than data transfer time in default (GWD) scheduling. GWD, most of the times, schedules *Job2* and *Job3* to *NodeC.globus.grid.* Whereas, NDASched schedules *Job2* and *Job3* to either of the *NodeA.globus.grid* and *NodeB.globus.grid*. *Job4* has no input file required for its execution, so its transfer time only because of some control files transferred to remote host for jobs execution.

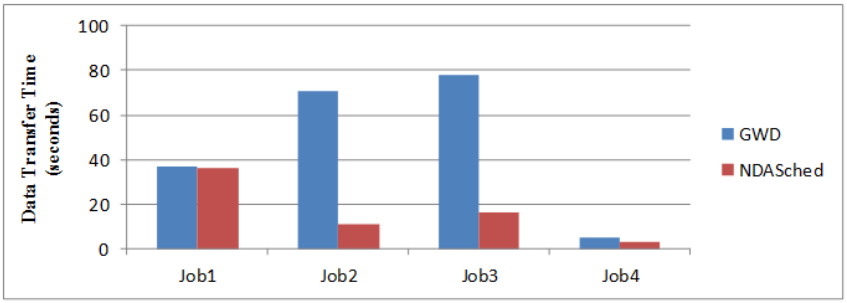

Figure 3. Comparison of Data Transfer Time

(B) Execution time: Figure 4 depicts that the execution time of jobs on grid. This execution time does not include the transfer time of jobs. When both the schedulers schedule job to the same host, then execution time is almost same (see *Job1* in figure 4). *Job2* and *Job3* are scheduled to different resources by both schedulers, so their execution time also differs because different resources have different computing power, and load on CPU.

Figure 4. Comparison of Execution Time

It is clear from above given comparison that using network characteristics, data locations of input files, and disk characteristics of data sources results in better job scheduling. The implemented network and data location aware scheduler (NDASched) takes longer time to schedule jobs as compares to that of GWD (GridWay's default scheduler). The GWD takes 8 seconds (average time) to schedule a job; whereas NDASched takes around 25 seconds for a job. This increased scheduling time is due to the overhead caused by spawning new processes to get the network characteristics and disk related information from *Iperf* and *Hdparm.* And we know that creating a new process is time consuming task in operating system. Therefore, NDASched takes a little longer time to schedule jobs.

6. Conclusion and Future Work

This works shows why network characteristics, data locations of input files, and disk read speed of data sources must be taken into account when scheduling data intensive jobs, not only to minimize file staging (data transfer) time over network, but also to minimize turnaround and waiting time of jobs in grid environment.

In this research work, Network and Data Location Aware scheduling algorithm is presented. The presented algorithm is tested by improving the existing GridWay MetaScheduler with the proposed scheduling algorithm. The results regarding the influence of the network characteristics, data locations, disk latency of data source, and jobs types variability are presented, showing that the improved GridWay can perform better job scheduling resulting to lower data transfer and turnaround time.

The current implementation of the proposed scheduling algorithm assumes that the input files required for the execution of a job are stored on a single data source. It can be further improved for the case in which input files of a job are stored on multiple data sources.

As we have discussed in last section that network and data location aware scheduler takes little longer time to schedule jobs than that of GridWay default scheduling. This scheduling time of proposed scheduling algorithm can be reduced by embedding the section of source code of *Iperf* and *Hdparm* which is responsible to get the desired information into the proposed scheduler. So the scheduler wouldn't need to spawn new processes each time we want to capture network characteristics and disk related information.

References

[1] Foster I, Kesselman C, "The grid: blueprint for a new computing infrastructure", Morgan Kaufmann, **(2004)**.

[2] Casavant TL, Kuhl JG, "A taxonomy of scheduling in general-purpose distributed computing systems", Software Engineering, IEEE Transactions, vol. 14, no. 2, pp. 141-154.

[3] Buyya R, "Economic-based distributed resource management and scheduling for grid computing", Citeseer, **(2002)**.

[4] Cheng L, Jin H, Qi L, Tao Y, "A flexible job scheduling system for heterogeneous grids", Advanced Parallel Processing Technologies, pp. 330-339.

[5] Carretero J, Xhafa F, Abraham A, "Genetic algorithm based schedulers for grid computing systems", Int'l Journal of Innovative Computing, Information and Control, vol. 3, no. 5, **(2007)**, pp. 1053–1071.

[6] Foster I, "Globus toolkit version 4: Software for service-oriented systems", Network and Parallel Computing, Springer, **(2005)**, pp. 2-13.

[7] McClatchey R, Anjum A, Stockinger H, Ali A, Willers I, Thomas M, "Data intensive and network aware (DIANA) grid scheduling", Journal of Grid Computing, vol. 5, no. 1, **(2007)**, pp. 43-64.

[8] Peris AD, Hernandez J, Huedo E, Llorente IM, "Data location-aware job scheduling in the grid. Application to the Gridway metascheduler", Journal of Physics, IOP Publishing, 062043, **(2010)**.

[9] Buyya R, Abramson D, Giddy J, "Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid", Published by the IEEE Computer Society, 283, **(2000)**.

[10] Frey J, Tannenbaum T, Livny M, Foster I, Tuecke S, "Condor-G: A computation management agent for multi-institutional grids", Cluster Computing, vol. 5, no. 3, **(2002)**, pp. 237-246.

[11] Tomás L, Caminero A, Caminero B, Carrión C, "Studying the influence of network-aware grid scheduling on the performance received by users", On the Move to Meaningful Internet Systems, OTM, **(2008)**, pp. 726-743.

[12] GridWay Metascheduler, "Metascheduling Technologies for the Grid", URL: http://gridway.org.

[13] Ranganathan K, Foster I, "Simulation studies of computation and data scheduling algorithms for data grids", Journal of Grid Computing, vol. 1, no. 1, **(2003)**, pp. 53-62.

[14] Ranganathan K, Foster I, "Decoupling computation and data scheduling in distributed data-intensive applications", High Performance Distributed Computing, IEEE, **(2002)**, pp. 352-358.

[15] Globus Toolkit, "Grid middleware", URL: http://www.globus.org.

[16] Cameron DG, Carvajal-Schiaffino R, Millar AP, Nicholson C, Stockinger K, Zini F, "Evaluating scheduling and replica optimisation strategies in OptorSim", IEEE Computer Society, vol. 52, **(2003)**.

[17] Tomas L, Caminero A, Caminero B, Carrion C, "Improving GridWay with network information: Tuning the monitoring tool", Parallel & Distributed Processing, IEEE, **(2009)**.

[18] Tirumala A, Qin F, Dugan J, Ferguson J, Gibbs K, "Iperf: The TCP/UDP bandwidth measurement tool", URL: http://sourceforge.net/projects/iperf/.

[19] Hdparm, "Disk utility for Unix systems", URL: http://linux.die.net/man/8/hdparm.