Data analysis with the DIANA meta-scheduling approach

Journal of Physics: Conference Series **119** (2008) 072004 doi:10.1088/1742-6596/119/7/072004

Data Analysis with the DIANA Meta-Scheduling Approach

Ashiq Anjum[1], **Richard McClatchey**[1] **and Ian Willers**[2]
[1]CCS Research Center, UWE Bristol
[2]CERN, Geneva Switzerland
E-mail: {ashiq.anjum; richard.mcclatchey; ian.willers}@cern.ch

Abstract. The concepts, design and evaluation of the Data Intensive and Network Aware (DIANA) meta-scheduling approach for solving the challenges of data analysis being faced by CERN experiments are discussed in this paper. Our results suggest that data analysis can be made robust by employing fault tolerant and decentralized meta-scheduling algorithms supported in our DIANA meta-scheduler. The DIANA meta-scheduler supports data intensive bulk scheduling, is network aware and follows a policy centric meta-scheduling. In this paper, we demonstrate that a decentralized and dynamic meta-scheduling approach is an effective strategy to cope with increasing numbers of users, jobs and datasets. We present "quality of service" related statistics for physics analysis through the application of a policy centric fair-share scheduling model. The DIANA meta-schedulers create a peer-to-peer hierarchy of schedulers to accomplish resource management that changes with evolving loads and is dynamic and adapts to the volatile nature of the resources.

1. Introduction

Data collection and data storage rates are growing at an exponential pace in high energy physics analysis. For example, the LHC experiments will need to store, access and analyze around 10 Petabytes per year which will require the use of around 200 Teraflop/s of processing power. By 2015, particle physicists will be using Exabytes of storage and Petaflop/s of computation [1]. If the current trends continue, such an explosion of data would clearly outstrip our ability to make meaningful use of it. With the unstoppable growth in data collection, meta-scheduling can play an important role in the way massive data sets are analyzed.

Future decision making systems will require quicker and more reliable algorithms, architectures and models for job scheduling to enable efficient discovery and analysis. Current approaches for job scheduling strongly depend on a single instance of a scheduler and cannot provide scalable solutions for bulk scheduling. The efficient utilization of a Grid infrastructure (around 200 sites in the LCG Grid) and data/results is not a trivial exercise if the scheduler is unable to submit jobs to a site which cannot process them quickly. There should be a scheduling mechanism that can help users in the rapid analysis of their jobs by selecting optimal sites and that can support 'chaotic' analysis where thousands of users submit millions of jobs and various schedulers should cooperate to handle the tasks which cannot be managed by a single scheduler. There should also be a mechanism to reduce the data transfer between sites since replication cost is too high for data intensive applications and jobs should be submitted to sites which have the best chance of executing them as soon as is practical. There should also be a mechanism whereby the scheduler should not have jobs lingering in queues for an unlimited time; these jobs should get a fair chance of execution at the earliest and each job and user

should get an optimum quality of service from the scheduling and resource management system. The DIANA Scheduling objective is that by using novel data scheduling algorithms and architectures, the information in a range of distributed datasets should be extracted quickly and efficiently with the same or a reduced set of resources. This should not only reduce the data transfer and event finding time (which can currently take days) but will also make significant savings in system resources.

2. The DIANA Scheduling Approach

We analyzed the scheduling problem in hand and tried to investigate the issues which are major sources of sub-optimal performance for the scheduling systems. We realized that existing scheduling systems build on the client-server model and are not fault tolerant. There is no mechanism that can help different meta-schedulers to intercommunicate and withstand the jobs pressure at a single site. There is no mechanism which can offload the job load on a site to remote peers and collaboratively work with other sites. Schedulers may be subject to failure or may not perform efficient scheduling when they are exposed to millions of jobs requiring different quality of service needs and different scheduling. They may not be able to re-organize or export scheduled jobs which could result in large job queues and long execution delays. What is required is a decentralized scheduling system which not only automatically exports jobs to its peers under potentially severe load conditions (such as with bulk jobs), but at the same time it manages its own scheduling policies, whilst queuing jobs and monitoring network conditions such as bandwidth, throughput and latency.

We also realized that the queuing mechanism that is required to improve such a scheduling problem should follow a carefully planned queuing scheme. In existing scheduling systems, most queuing approaches are based on a First-In-First-Out policy and other similar considerations which can lead to a poor quality of service and jobs hanging for days. Such approaches are absolutely not feasible where users want immediate response to their jobs or the jobs are critical and can not wait more than a predefined time. There should be a mechanism to associate priorities to each job inside the queue, depending on the user profile and job requirements with the scheduler servicing high priority jobs preferentially to improve the quality of service provision to users. There should also be a mechanism for exporting jobs to least loaded sites and for scheduling 'out of turn' high priority jobs. There should also be a mechanism to provide optimal utilization of all resources to avoid resource starvation. An efficient queuing approach is one of the most important considerations for ensuring quality of service and smooth functioning of the Grid operations for a diverse range of users and applications.

We also found that the best match between jobs and resources is central to optimizing the matchmaking process. We therefore need to embed the network information into the scheduling algorithm to improve the efficiency and the utilization of a Grid system. The overall goal is to minimize the computing time for applications which involve large-scale data. Hence, the centralized scheduling algorithms that focus only on maximizing processor utilization by mapping jobs to idle processors, and disregarding network costs, queue times, job priorities and costs associated with accessing remote data are unlikely to be efficient [2]. Similarly, the scheduling decisions which always force the job movement towards the data without taking the Grid "weather", the network load and the data size and location into consideration can lead to significant inefficiencies in performance and can be responsible for large job queues and processing delays.

Considering the effects of and then compensating for network characteristics can avoid making these less-than-ideal scheduling decisions. Our big challenge thus becomes finding means to express these requirements in a format that the meta-scheduler engine can understand. This engine should use mathematical techniques to make decisions and generate the overall behaviour of the system based on the global network characteristics. The basic job scheduling algorithm at each site should be driven by some weighting value calculated for each potential target location which is a function of the available network characteristics, the processing cycles, job priorities, queue length and input and output "sandboxes" of data and the one having least cost should be given priority. The DIANA (Data Intensive and Network Aware) scheduling approach builds upon three considerations as outlined in the following sections.

2.1. Architectural considerations for the DIANA Meta-Scheduler

A meta-scheduler coordinates the communication between multiple heterogeneous local schedulers that typically manage clusters in a LAN environment. These local and meta-schedulers form a hierarchy and individual schedulers sit at different levels in the hierarchy as discussed by Mausolf in [3]. Each local scheduler can cooperate and communicate with its siblings through a meta-scheduler, however, each meta-scheduler cannot communicate with other meta-schedulers of other sites or Grids. Communication is only possible between local schedulers and the meta-scheduler. In the P2P approach, a meta-scheduler and a local-scheduler make a hierarchy at each site where global decisions are managed by the meta-scheduler whereas local control and allocations are made by the local scheduler. The meta-scheduler on each site has access to global information and all meta-scheduler instances communicate with each other to share the cost and load information. Meta-schedulers do not take global decisions at a single central point; rather many sites can participate in the scheduling decisions through sharing the information in their cost matrices. Each site can have information on load, queue size etc., can monitor its processing nodes and then propagate this information to other peers. Local and certain global policies are managed at the site level instead of a central hierarchical management.

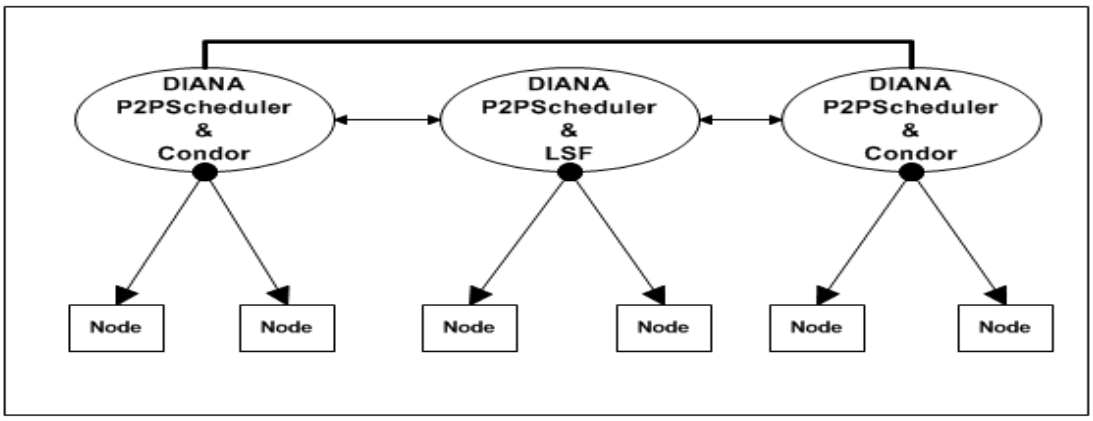

Figure 1: P2P Communication between Schedulers

In the absence of this mechanism, it is possible that some of the jobs might be lost by the scheduler due to timeouts before they get an execution slot, insufficient space in the queue to hold the jobs or the fact that the frequency of submission cannot be handled by the central scheduling site. DIANA is a Data Intensive and Network Aware meta-scheduler which performs global meta-scheduling in a local environment and different meta-schedulers work in a P2P manner. Each site has a meta-scheduler that can communicate with all other meta-schedulers on other sites as shown in Figure 1. The scheduler is able to discover other schedulers with the help of a discovery mechanism. We do not replace the local schedulers in this architecture rather we have added a layer over each local scheduler so that site meta-schedulers can talk directly to each other instead of getting directions from a central meta-scheduler.

2.2. Algorithmic considerations for Job placement

In the proposed scheduling algorithm, we want to maximize CPU utilization and throughput and minimize turnaround time, waiting and response time. A scheduling algorithm is created which forms an important element of the matchmaking process. The following are the targeted metrics within the scheduling process by which the optimization level of the scheduling system needs to be gauged:

- Queue and waiting time

- Processing and execution time
- Input data transfer time
- Executable transfer time
- Results transfer time

The total time to execute a job in a Grid environment will be the sum of all of these times. There are three major cost estimates which need to be calculated for the scheduling algorithm: the network, computation and data transfer costs and scheduling optimization will be based on these estimates. These estimates further depend on the parameters discussed in the following sections. The importance of each cost function can be adjusted by allocating weights to these parameters. These weights are dynamically assigned and their allocation procedure is discussed in the following sections.

2.2.1 Network Cost

By far the most important factor affecting the data intensive scheduling process is that of network cost. The load, capacity and availability of network links used during data transfers may heavily affect the Grid application performance. In order to provide the right quality of service (QoS) to Grid applications and hence to optimize scheduling, it is important to first understand how the network is performing and to determine the level of quality of service that currently exists in the network.
TCP throughput can be obtained by combining the losses and the Round Trip Times (RTTs) using Mathis's formula [4] for deriving the maximum TCP throughput. Given the historical measurements of the packet loss and the RTT, we can calculate the maximum TCP bandwidth for a certain amount of time for various groups of sites. Mathis's formula describes a useful formula for the upper bound on the transfer rate:

$$\texttt{Rate} < \left(\frac{MSS}{RTT}\right) \times \left(\frac{1}{\sqrt{loss}}\right)$$

Equation 1: TCP throughput calculation

Where rate is the TCP transfer rate, MSS is the maximum segment size (fixed for each Internet path, typically 1460 bytes), RTT is the round trip time (as measured by TCP) and loss is the probability of packet loss. We assign weights to each value depending upon the importance of the parameters to calculate an aggregate value of the network cost (NetCost):

$$NetCost \propto \frac{Losses}{Bandwidth}$$
$$where$$
$$Losses = RTT \times loss \times W2 \times Jitter$$

Equation 2: Calculation of the Network Cost

Where W is the weight assigned to each parameter depending upon the importance of a parameter. On the Internet, the network partitions a message into parts, called packets, with a certain size in bytes. A typical packet contains perhaps 1,000 or 1,500 bytes. Therefore loss is measured in bytes per second, RTT is measured in milliseconds and jitter is a number. Bandwidth is measured in bits per second and therefore network cost is in seconds. In extreme circumstances, it can be in hours if the network is performing poorly. All weights are assigned subject to a cost and a higher cost will lead to a higher weight. We can manipulate these weights to prioritize certain parameters in the algorithm.

2.2.2 Computation Cost

The second important cost which needs to be part of the scheduling algorithm is the computation cost. Jin et al. describe a mathematical formula [5] to compute the processing time of a job:

$$\text{Computation Cost} = \frac{Q_i}{P_i} \times W5 + \frac{Q}{P_i} \times W6 + SiteLoad \times W7$$

Equation 3: Formula for the Computation Cost

Where Q is the total number of waiting jobs on all the sites, Qi is the length of the waiting queue on site i, Pi is the computing capability of the site i (the total number of processors at site i) and SiteLoad is the current load on that site. SiteLoad is calculated by dividing the number of jobs running by the processing power of that site. The Qi/Pi ratio computes the processing time of the job. The unit of the computation cost is time (minutes or hours).

2.2.3 Data Transfer Cost

The third most important cost aspect in data intensive scheduling is the Data Transfer Cost (DTC) which includes input data, output data and executables. Park and Kim describe a mathematical technique [6] to calculate the aggregate data transfer time which includes all three parameters. Here we do not use bandwidth only to calculate the data transfer cost, rather we use the Network Cost (NC), as calculated in Section 2.2.1. A higher network cost will lead to a longer data transfer time and therefore a higher data transfer cost. Similarly a larger data size will take more time to transfer and therefore the cost required to transfer this data will be higher. From this discussion, we can deduce that the network cost and data transfer cost are proportional to each other and therefore:

$$DTC = W8 \times Data \times NC \qquad \ldots\ldots\ldots (I)$$

We can further expand the equation I into the following expression:

Data Transfer Cost (DTC) = Input DTC + Output DTC + Executables transfer cost

$$DTC = (InputData \times NC) + (OuputData \times NC) + (executable \times NC)$$

Equation 4: Data Transfer Cost

In this equation, we consider three elements for data transfer. All data transfer costs are basically the time consumed in transferring the data, therefore the units balance on both sides of the equation. Executables mentioned in the data transfer cost are application data and user code which will be submitted for execution.

2.2.4 Total Cost

Once we have calculated the cost of each stake holder, the total cost is simply a combination of these individual costs as calculated in Sections 2.2.1, 2.2.2 and 2.2.3 thus:

Total Cost C = Network Cost + Computation Cost + Data transfer Cost

The main optimization problem that we want to solve is to calculate the cost of data transfers betweens sites (DTC), to minimize the network traffic cost between the sites (NTC) and also to minimize the computation cost of a job within a site. So, for any particular job we calculate the data transfer costs across all sites and choose the one with the minimum network, data transfer and compute cost.

2.3. Queue Management for the DIANA Meta-scheduler

In conventional client-server scheduling architectures, local schedulers handle their queues at the site level whereas a meta-scheduler has a global queue. However, in the DIANA architecture, there is one DIANA meta-scheduler at each site, i.e. the DIANA P2P meta-scheduler layer sits on top of one or many local schedulers at each site. This leads to a scalable and self-organizing meta-scheduling behaviour which was previously missing in some of the conventional client-server scheduling architectures. Each meta-scheduler has a queue management mechanism where it can queue the incoming jobs in a Scheduler Queue as shown in Figure 2, and the meta-scheduler assigns priorities to the incoming jobs.

At each site there are two queues. One is the meta-scheduler queue and the other is the queue of the local scheduler. The meta-scheduler queue deals with the global jobs and takes into account the Grid information whereas the local queue is site specific. Jobs cannot be migrated if the meta-scheduler has scheduled them to any local scheduler and they will have to wait in the local scheduler queue until they get the execution slot. Only the jobs from the DIANA meta-scheduler queue are exported to other sites. In contrast, once a job is allocated to a local scheduler at a site, it is never exported and waits in the local queue until assigned to a processor. All the prioritization of jobs, policy enforcement, migration and job steering issues are handled at the DIANA P2P level whereas the local scheduler works exactly in the same fashion as before once the job has been allocated to it.

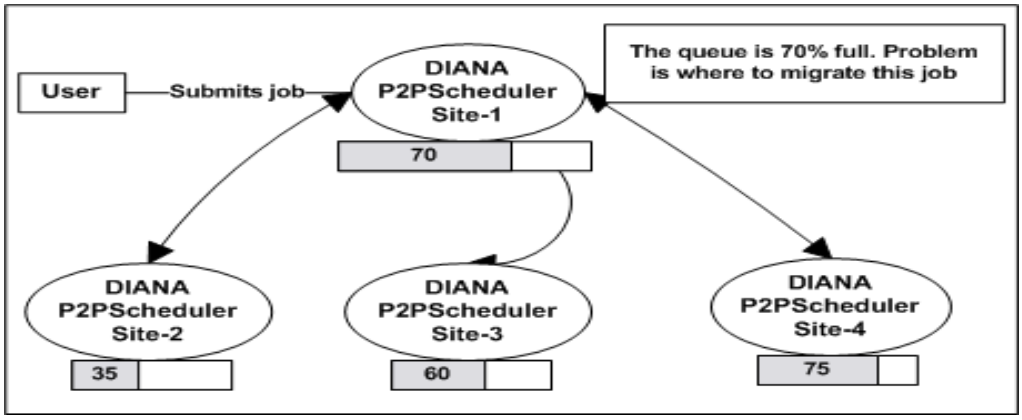

Figure 2: Queue Management in DIANA P2P Scheduler

The re-prioritization algorithm may result in the migration of jobs from low priority to high priority queues or vice versa. The reprioritization technique militates against aging since the jobs are assigned new priorities on the arrival of each new job and each job gets its appropriate place in the queues according to the new circumstances. In the case of congestion in the queues, the algorithm will migrate the jobs to some remote site where there are fewer jobs waiting in the queues.

3. Results and Discussion

3.1 Architecture related results

We present here the results of the scalability tests for the DIANA scheduling approach. These are simulation results since it was not feasible to deploy the complete DIANA system on such a large number of sites. In these tests, we assumed that there was a meta-scheduler on each node (here, a node corresponds to a site), and all the nodes worked in a P2P manner. As shown in figure 3, the number of nodes and the number of jobs scheduled to the Grid were increased gradually to test which algorithm gave the steepest increase in time taken. In this test, jobs of a processing requirement of 3 MFLOP and a bandwidth load of 1 MB were launched to the Grid. The Round Robin algorithm has a steep linear curve showing that it is the most non-scalable of the candidates. The FLOP (**Fl**oating point **O**perations

Per **S**econd) shows too much variation in this case, although, on the whole, it clearly is more scalable than Round Robin. The DIANA approach has the best performance; it shows a nearly linear increase, and hence it is very scalable. This also indicates that DIANA is a suitable approach for large scale Grids since it can support increasing numbers of nodes.

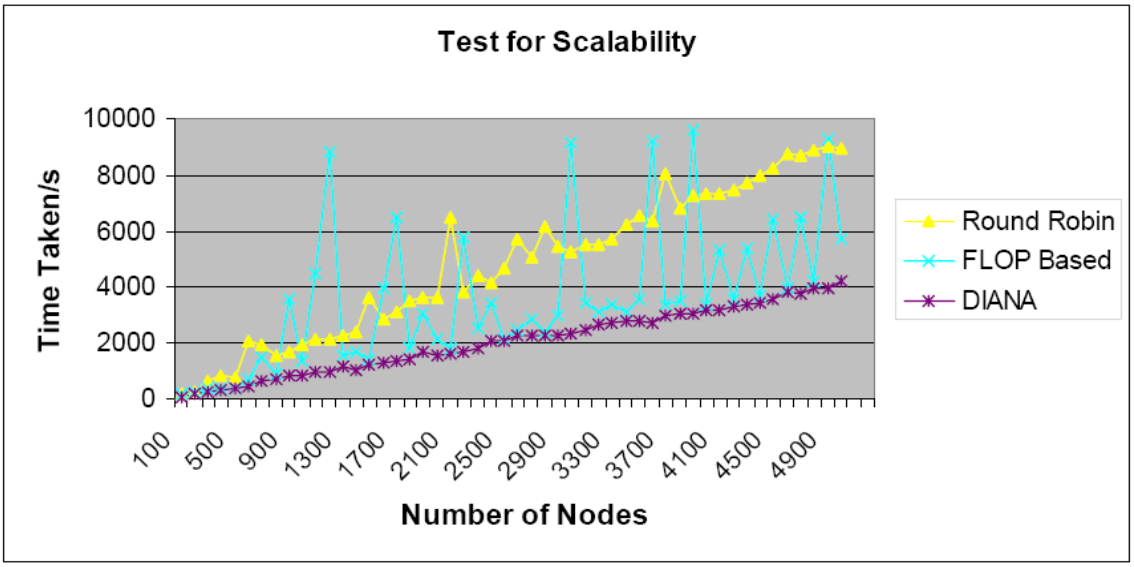

Figure 3: Scalability Test for DIANA

3.2 Algorithm related results

We first submitted jobs on the GILDA Testbed without the DIANA Scheduler and measured the queue, execution and data transfer times. After this, jobs were re-submitted following the algorithm employed in the DIANA Scheduler which includes the measurement of queue, execution and data transfer times. We took a particular computationally intensive job from the CMS experiments which produced a very large amount of data. We selected this job because its execution time is of the order of minutes, in order to minimise the effect of varying network characteristics. Tests are performed by submitting jobs through GILDA's user interface. The client machine had a Pentium based 2.4 GHz processor and 1 GB RAM. The network card was of 100 Mbps capacity.

The DIANA Scheduling is equally applicable to short and long duration jobs. For the longer jobs it is the execution time which will vary and accordingly queue times will also increase. The execution cost will remain the same with time since, once a job is submitted, whether it is a long or a short job, it will not be pre-empted until it completes its execution (and therefore it is not time dependant). The same is the case for the data transfer cost which should remain the same whether a longer job is being executed or a shorter job is being scheduled. The only variable which can change with time is the network cost. Although the network cost can influence the data transfer cost it does not affect the execution time since jobs do not communicate with each other during execution. The data transfer cost is the replication cost and is equally important for the longer and shorter jobs as far as their execution times are concerned.

Firstly we submitted 25 jobs on the GILDA Testbed and observed their queue and execution times. The GILDA Testbed employed the gLite workload management system (WMS) as a meta-scheduler and therefore the submitted jobs followed either eager or lazy scheduling with resources being allocated on a FCFS basis. Figure 4 and 5 show the queue and execution times of WMS against which we are comparing the DIANA meta-scheduler. Secondly, we submitted the same number of jobs three times and re-measured the queue and execution times. Then we increased the number of jobs to 500 and then gradually to 1000, so that we could check the capability of the existing scheduling system. We can see that with an increasing number of jobs the execution performance increases which

indicates the effect of the DIANA scheduling approach. Here we note that the effect of DIANA became more significant as the number of jobs increased since DIANA identifies only those sites for job executions which are least loaded and which preferably have the required data, since this will reduce any transfer times. As shown in Figures 5, the DIANA Scheduler with its multi-queue priority mechanism had an improved execution. Multi-queuing not only enabled the short job first execution but also managed the queues on a priority basis and significantly reduced the total execution times.

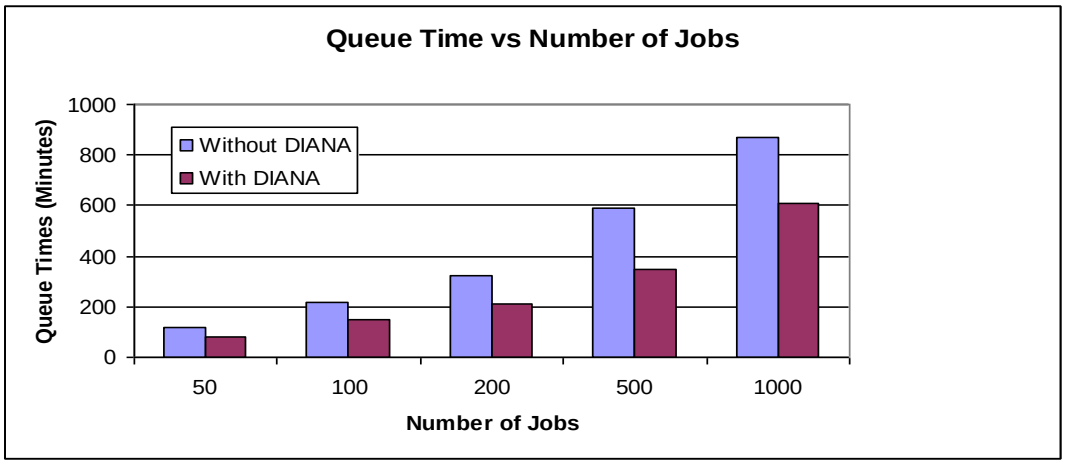

Figure 4: Queue time versus number of jobs

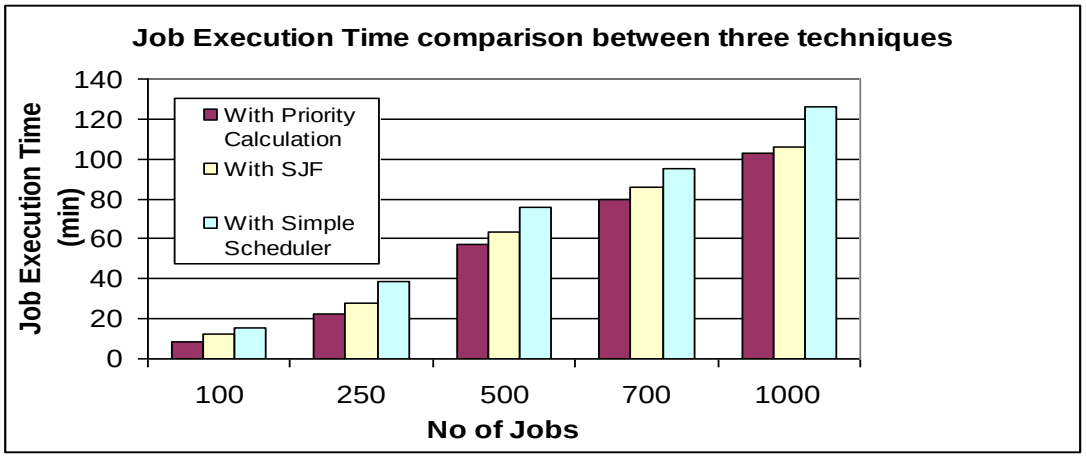

Figure 5: Execution time comparison

Figure 6 demonstrates the results related to network issues which have a high impact on the execution of the data intensive jobs. In this experiment, we submitted the same number of jobs to different sites with different network conditions. The bandwidth varied from 10MB/s to 1000MB/s to enable us to gauge its effect on the job execution time. We used Iperf [7] to generate the extra network traffic and to saturate the network so that available bandwidth could vary from 10 to 1000 Mbps. In these tests we showed the effect of bandwidth on the execution time of the jobs. The data size was the same for all the jobs. Here the execution time included the time required to schedule and execute the job to one of the 'best sites' plus the time required in sending the data and job to that remote site and the time elapsed in queuing on that remote site. We used different networks to check the influence of the network parameters on the data transfer cost. From the comparison graph in Figure 6, we note that the network plays a vital role in scheduling decisions.

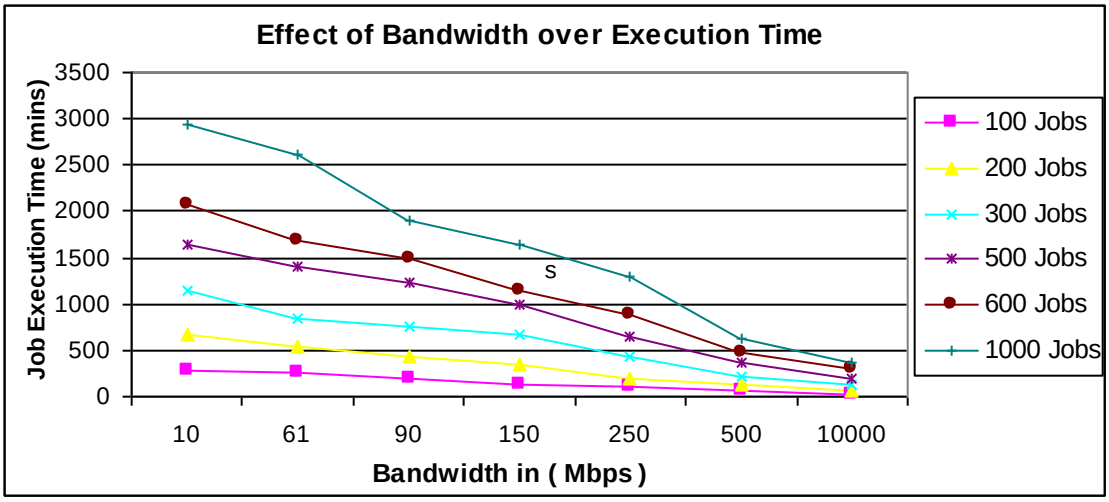

Figure 6: Execution times vs. bandwidth.

3.3 Queue related results

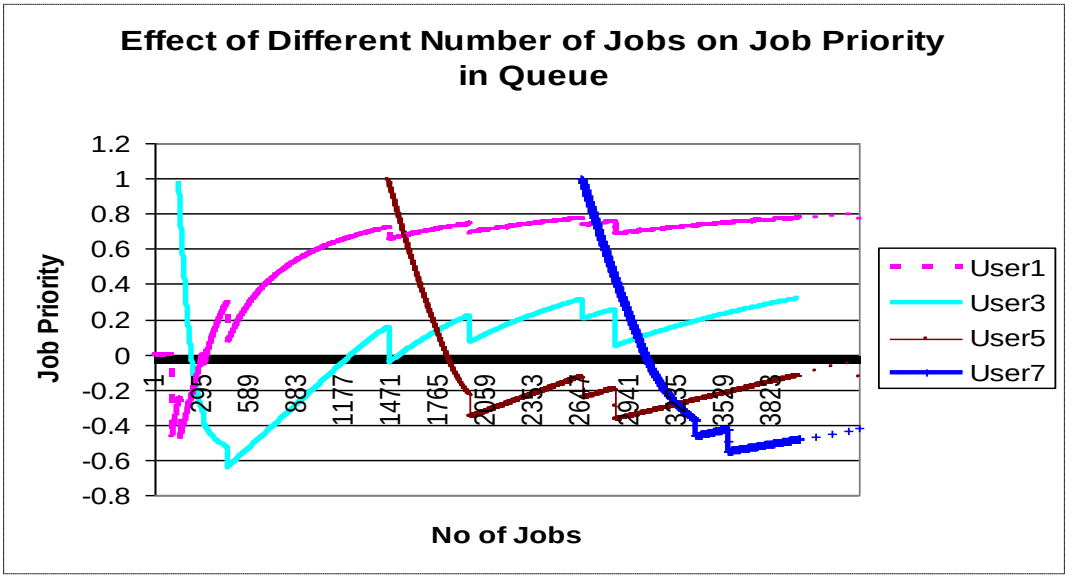

Figure 7. Number of Jobs and Priority

The results suggest that job priority decreases gradually when a user keeps sending jobs in the scheduler queue. Thus, in the case when a user bombards a site with hundreds of jobs, the scheduler decreases the priority of the jobs of that particular user. Figure 7 shows that at the outset, when a user starts sending his jobs in the queue its priority was high however, if he bombard the site with jobs, the priority decreased gradually with an increasing number of jobs. The results suggest that a large number of jobs from a user lead to a lower job priority. Thus the algorithm makes best use of the available resources to schedule the short jobs before the resources are allocated to the large jobs. It helps the scheduler to prevent resource starvation since shorter jobs will have to wait for a very long time if long jobs are scheduled first. Similarly, a higher number of jobs from a user will lead to a lower priority for his/her jobs so that no user can prevent other users in using resources by submitting

bursts of jobs. The priority mechanism will prevent the users and jobs from manipulating Grid resources and will provide a fair-share utilization of resources for all users and applications.

4. Conclusions and Future Direction

We have demonstrated that a cost-based approach can improve the scheduling process if each job is submitted and executed after taking into consideration certain associated costs. Our results demonstrate that a P2P meta-scheduler is better suited to Data Intensive And Network Aware (DIANA) scheduling than a single, centralized meta-scheduler. A multi-queue, priority-driven feedback based bulk scheduling algorithm is proposed and the results suggest that it can significantly improve the Grid scheduling and execution process.

In future, we plan to explore the data characteristics and train the algorithms which can minimize the analysis and discovery time by running them on distributed resources and using new matchmaking and knowledge discovery techniques. There are a number of questions which need to be investigated in this research area. For example, how can we scale the algorithms as the number of records or observations, attributes and the demand for interactivity and real-time response increases? How can we automatically (instead of manually) find the patterns and clusters of particles in the (physics) data distributed across the world using scalable algorithms by mining large, massive and high dimensional data sets and how can we predict the future trends using the past behaviour of the particles? Similarly, we also need to find how we can minimize the replication and data transfer using some abnormality and outlier detection techniques.

5. References

[1] Tony Hey and Anne Trefethen, Grid Computing – Making the Global Infrastructure a Reality, Wiley, January 2003

[2] Kavitha Ranganathan ,Ian Foster, Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications , HPDC 2002.

[3] Jeff Mausolf, IBM, Grid Computing Initiative Grid in action: Managing the resource managers, developerWorks, 2005 http://www-128.ibm.com/developerworks/Grid/library/gr- metasched/

[4] Mathis, Semke, Mahdavi & Ott, The macroscopic behaviour of the TCP congestion avoidance algorithm, Computer Communication Review, 27(3), July 1997.

[5] H. Jin, X. Shi et al. An adaptive Meta-Scheduler for data-intensive applications, International Journal of Grid and Utility Computing 2005 - Vol. 1, No.1 pp. 32 - 37

[6] S. Park, and J. Kim, Chameleon: a resource scheduler in a data Grid environment, Proceedings of the 3rd IEEE/ACM Symposium on Cluster Computing and the Grid, Tokyo, 2003

[7] Iperf Bandwidth measurement tool, http://dast.nlanr.net/Projects/Iperf/