A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids

Srikumar Venugopal, Rajkumar Buyya
GRIDS Laboratory and NICTA Victoria Laboratory
Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia
{srikumar,raj}@cs.mu.oz.au

Lyle Winton
Experimental Particle Physics Group
School of Physics
The University of Melbourne, Australia
winton@ph.unimelb.edu.au

ABSTRACT

Large communities of researchers distributed around the world are engaged in analyzing huge collections of data generated by scientific instruments and replicated on distributed resources. In such an environment, scientists need to have the ability to carry out their studies by transparently accessing distributed data and computational resources. In this paper, we propose and develop a Grid broker that mediates access to distributed resources by (a) discovering suitable data sources for a given analysis scenario, (b) suitable computational resources, (c) optimally mapping analysis jobs to resources, (d) deploying and monitoring job execution on selected resources, (e) accessing data from local or remote data source during job execution and (f) collating and presenting results. The broker supports a declarative and dynamic parametric programming model for creating grid applications. We have used this model in grid-enabling a high energy physics analysis application (Belle Analysis Software Framework) on a grid testbed having resources distributed across Australia.

1. INTRODUCTION

The next generation of scientific experiments and studies, popularly called as *e-Science* [14], will be carried out by communities of researchers from different organizations that span national and international boundaries. Some well-known scientific experiments of this nature include the CERN-led ATLAS and CMS experiments and the KEK-led Belle experiment. These experiments involve geographically distributed and heterogeneous resources such as computational resources, scientific instruments, databases and applications.

Grid computing [11] enables aggregation and sharing of these resources through by bringing together communities with common objectives and creating virtual organizations [12]. Data Grids [9] have evolved to tackle the twin challenges of large datasets and multiple data repositories at distributed locations in data-intensive computing environments [16]. However, the harnessing the power of grids remains to be a challenging problem for users due to the complexity involved in the creation and composition of applications and their deployment on distributed resources.

2nd International Workshop on Middleware in Grid Computing, October 18, 2004, Toronto, Ontario, Canada.
Copyright 2004 ACM 1-58113-950-0/00/0004…$5.00.

Resource brokers hide the complexity of grids by transforming user requirements into a set of jobs that are scheduled on the appropriate resources, managing them and collecting results when they are finished. A resource broker in a data grid must have the capability to locate and retrieve the required data from multiple data sources and to redirect the output to storage where it can be retrieved by processes downstream. It must also have the ability to select the best data repositories from multiple sites based on availability of files and quality of data transfer. In this paper, one such broker called the Gridbus Broker providing services relevant to data-intensive environments is presented. Its application to the high-energy physics domain is discussed by illustrating its use within the Belle Analysis Data Grid and the results of experiments that have been conducted on it are presented.

2. RELATED WORK

While the scheduling of independent jobs on distributed heterogeneous resources is a well-studied problem [15], the discussion here is restricted to those efforts which deal with Grids.

The Gridbus broker extends the Nimrod-G [1] computational Grid resource broker model to distributed data-oriented grids. Nimrod-G specializes in parameter-sweep computation and its model has been proven successful for several applications [4][18]. However, the scheduling approach within Nimrod-G aims at optimizing user-supplied parameters such as deadline and budget [5] for computational jobs only. It has no functions for accessing remote data repositories and for optimizing on data transfer. The Gridbus broker also extends Nimrod-G's parametric modeling language by supporting dynamic parameters, i.e. parameters whose values are determined at runtime.

Like Nimrod-G, the AppLeS PST [7] [8] supports deployment of parameter-sweep applications on computational grids, but its adaptive scheduling algorithm emphasizes on data-reuse. The users can identify common data files required by all jobs and the scheduling algorithm replicates these data files from the user node to computational nodes. It tries to re-use the replicated data to minimize the data transmission when multiple jobs are assigned to the same resource. However, multiple repositories of data are not considered within this system and therefore, this scheduling algorithm is not applicable to Data Grids.

Ranganathan and Foster [17] have conducted simulation studies for various scheduling scenarios within a data grid. Their work recommends decoupling of data replication from computation while scheduling jobs on the Grid. It concludes that it is best to schedule jobs to computational resources that are closest to the data required for that job, but the scheduling and simulation studies are restricted to homogeneous nodes with a simplified First-In-First-Out (FIFO) strategy within local schedulers.

Similar to [17], our work focuses on a resource scheduling strategy within a Data Grid but we concentrate on adaptive scheduling algorithms and brokering for heterogeneous resources that are shared by multiple user jobs. In addition, the scheduling strategy has been implemented within the Gridbus broker and its feasibility to support the deployment of distributed data-intensive applications (e.g. KEK Belle high-energy physics experiment data analysis) within a real Grid testbed (e.g., Australian Belle Analysis Data Grid) has been evaluated.

3. ARCHITECTURE

3.1 Data Grid Overview and Brokering

A data-intensive computing environment can be perceived as a real-world economic system wherein there are producers and consumers of data. Producers are entities which generate the data and control its distribution via mirroring at various replica locations around the globe. Information about the data replicas is assumed to be available through a data catalogue mechanism such as the Globus Replica Catalog [19]. The consumers in this system would be the users who may need to investigate specific datasets out of a set of hundreds and thousands. A sample scenario for such a data-intensive computing environment and the role of the broker in it is discussed in [6].

3.2 Gridbus Data Grid Service Broker

The architecture of the Gridbus broker is shown in Figure 1. The inputs to the broker are the tasks and the associated parameters with their values. These can be specified within a "plan" file that specifies the tasks and the types of the parameters and their values for these tasks.

A task is a sequence of commands that describe the user's requirements. For example, the user may specify an application to be executed at the remote site, an input file to be copied over before execution and the results to be returned back. A task encapsulates this information within its description. A task is accompanied by parameters which can either be static or dynamic. A static parameter is a variable whose domain is well-defined either as a range of values, as a single static value or as one among a set of values. A dynamic parameter has either an undefined or an unbounded domain whose definition or boundary conditions respectively, have to be established at runtime. As an example, in the current implementation, a parameter type has been defined which describes a set of files over which the application has to be executed. This set can be described as a wildcard search within a physical or a logical directory, to be resolved at runtime, thus creating a dynamic parameter.

The task requirements drive the discovery of resources such as computational nodes and data resources. The resource discovery module gathers information from remote information services such as the Grid Index Information Service (GIIS) [10] for availability of compute resources. Optionally, the list of available compute resources can be provided by the user to the broker. The broker also interacts with the information service on each computational node to obtain its properties. Data files can be organised as Logical File Names (LFNs) within a virtual directory structure using a Replica/Data Service Catalog. Each LFN maps to one or many Physical File Names (PFNs) somewhere on the Grid, usually specified via URLs. The broker will resolve the LFNs to the appropriate physical file location(s) by querying the catalog.

The task description, i.e. the task along with its associated parameters, is resolved or "decomposed" into jobs. A job is an instantiation of the task with a unique combination of parameter values. It is also the unit of work that is sent to a Grid node. The set of jobs along with the set of service nodes are an input to the scheduler. For jobs requiring remote data, the scheduler interacts with a network monitoring service to obtain the information about current available bandwidth between the data sources and the compute resources. In the current implementation, the Network Weather Service (NWS) [21] has been used to obtain this information. The scheduling algorithm is described in more detail in the next section.

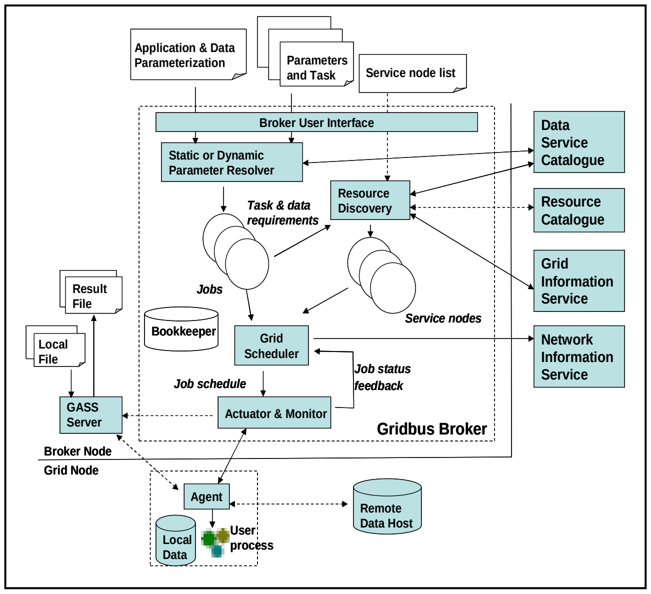


Figure 1: Gridbus broker architecture.

The jobs are dispatched to the remote node through the Actuator component. The Actuator submits the job to the remote node using the functionality provided by the middleware running on it. The Actuator has been designed to operate with different Grid middleware frameworks and toolkits such as Globus 2.4 [13]. The task commands are encapsulated within an Agent which is dispatched to and executed on the remote machine. If a data file has been associated with the job and a suitable data host identified for that file, then the Agent obtains the file through a remote data transfer from the data host. Additionally, it may require some configuration or input parameter files that it obtains from the broker through a mechanism such as a GASS [2] (Globus Access to Secondary Storage) Server. These files are assumed to be small and in tens or hundreds of kilobytes which impact the overall execution time of a job negligibly whereas the data files are in the range of megabytes or larger. On the completion of execution, the Agent returns any results to the broker and provides debugging information. The Monitoring component keeps track of job status – whether the jobs are queued, executing, finished successfully or failed. The Bookkeeper keeps a persistent record of job and resource states throughout the entire execution.

The design and implementation of the broker could not be described here due to paucity of space. The interested reader can refer to the related technical report [20] for details.

3.3 Scheduling

The scheduler within the broker looks at a data grid from the point of view of the data. It perceives a data-intensive computing environment as a collection on data hosts, or resources hosting the data, surrounded by compute nodes. Some of the data resources may have computation facilities too, in which case there is assumed to be nearly infinite bandwidth between the data host and the compute resource at the same site. The scheduling heuristic is listed in Figure 2.

The scheduler minimizes the amount of data transfer involved while executing a job by dispatching jobs to compute servers which are close to the source of data. A naïve way of achieving this would be to run the jobs only on those machines that contain their data. But, the data hosts may not have the best computational resources. Hence, the scheduler selects an appropriate compute resource to execute a job based on factors such as capability and performance of the resource, bandwidth available from the compute resource to the data host that contains the data file required for the job and the cost of data transfer. A detailed analysis and performance evaluation of this scheduling algorithm is out of scope of this paper. However, an evaluation using a high energy physics analysis application within a Data Grid environment is presented in the next section.

4. EXPERIMENTAL EVALUATION

High-Energy Physics (HEP) Experiments are large and technically sophisticated and necessarily involve international collaboration between many institutes over very long time scales. Computing resource requirements for HEP are increasing exponentially because of advancements in particle accelerators and increasing size of collaborations. Therefore, data grids are important to ensure continued availability of computational and data resources in experimental high energy physics. A survey of the data grid efforts in this domain is presented in [3].

The Belle experiment, built and operated by a collaboration of 400 researchers across 50 institutes from 10 countries, provides the state-of-the-art instruments to explore the effects of Charge-Parity (CP) violation within B-mesons produced at the the KEKB accelerator at the Japanese High Energy Accelerator Research Organization (KEK) in Tsukuba. The current experiment and simulation data set is increasing rapidly and has begun to pose problems for the processing and access of data at geographically remote institutions, such as those within Australia. Hence, it is important for Data Grid techniques to be applied in this experiment [21].

4.1 The Testbed

The Belle Analysis Data Grid (BADG) testbed has been set up in Australia in collaboration with IBM. The location, configuration and capabilities of the testbed resources are shown in Figure 3. Each of the nodes have 4 CPUs (2 Intel Xeons), except for the PC in the School of Physics, University of Melbourne which has only one. However, two of these resources (Adelaide and Sydney) were effectively functioning as single processor machines as the Symmetric Multi-Processing (SMP) Linux kernel was not running on them. All the nodes in this testbed were running Globus 2.4.2

<u>Initialisation</u>

1. Identify characteristics, configuration, capability, and suitability of compute resources using the Grid information services (GIS).
2. From the task definition, obtain the data query parameters (if present), such as the logical file name
   a. Resolve the data query parameter to obtain the list of Logical Data Files (LDFs) from the Data Catalog
   b. For each LDF, get the data sources or Data Hosts that store that file by querying the Data Catalog.

<u>Scheduling Loop</u>

Repeat while there exist *unprocessed jobs*. [This step is triggered for each scheduling event. The event period is a function of job processing time, rescheduling overhead, resource share variation, etc.]:

1. For each compute resource, predict and establish the *job consumption rate* or the available resource share through the measure and extrapolation strategy taking into account the time taken to process previous jobs. Use this estimate along with its current commitment to determine expected job completion time.
2. If any of the compute resource has jobs that are yet to be dispatched for execution and there is variation in resource availability in the Grid, then move such jobs to the Unassigned-Jobs-List.
3. Repeat until all unassigned jobs are scheduled or all compute resources have reached their maximum job limit.
   a. Select the next job from the Unassigned-Jobs-List.
   b. Identify all Data Hosts that contain the LDF associated with the job.
   c. Create a Data-ComputeResource-List for the selected job:
   d. For each data host, identify a compute resource that can complete the job earliest given its current commitment, job completion rate, and data transfer time using current available bandwidth estimates.
   e. Select a data host and compute resource pair with the earliest job completion time from the Data-ComputeResource-List.
   f. If there exists such a resource pair, then assign the job to the compute resource and remove it from the Unassigned-Jobs-List.

End of scheduling loop.

Figure 2: Adaptive scheduling algorithm for Data Grid.

and NWS sensors and except for the Adelaide node, are connected via GrangeNet, a Gigabit wide-area network within Australia. The broker was deployed on the Melbourne Computer Science machine and broker agents were dispatched at runtime to the other resources for executing jobs and initiating data transfers.

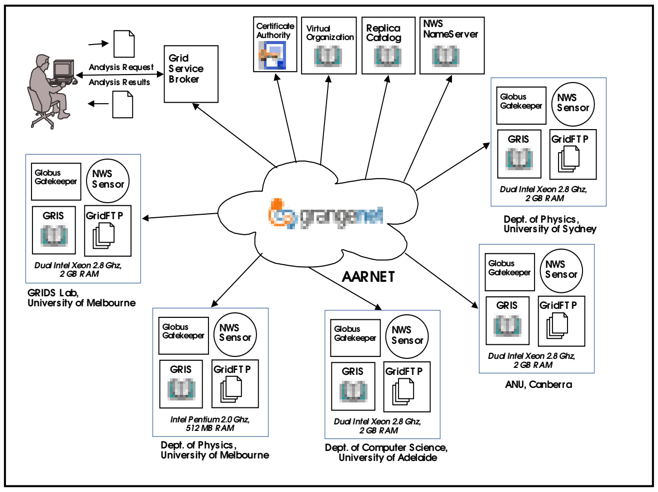

Analysis Request
Analysis Results
Grid Service Broker
Certificate Authority
Virtual Organization
Replica Catalog
NWS NameServer
Globus Gatekeeper
NWS Sensor
GRIS
GridFTP
Dual Intel Xeon 2.8 Ghz, 2 GB RAM
Dept. of Physics, University of Sydney
GRIDS Lab, University of Melbourne
grangenet
AARNET
ANU, Canberra
Intel Pentium 2.0 Ghz, 512 MB RAM
Dept. of Physics, University of Melbourne
Dept. of Computer Science, University of Adelaide

Figure 3: Australian Belle Analysis Data Grid testbed.

Data that was produced on one site in BADG had to be shared with the other sites. For this purpose, a Data Catalog was set up for BADG using the Globus Replica Catalog (RC) mechanism. The primary application for the Belle experiment is the Belle Analysis Software Framework (BASF). This application is used for simulation, filtering of events, and analysis. It is also a legacy application that consists of about a Gigabyte of code. Therefore, this application was installed on all the resources beforehand.

4.2 Application Parameterisation and Experimental Setup

The experiment consists of 2 parts, both of which involve execution over the Grid using the Gridbus broker. In the first part, 100,000 events of the “decay chain” of particles, B0->D*+D*-Ks as shown in Figure 4, are simulated via distributed generation and this data is entered into the replica catalog. In the analysis part, the replica catalog is queried for the generated data and this is analysed over the Belle Data Grid to generate histograms. Here only the results of the analysis are discussed as it involved accessing remote data.

A plan file for the composing analysis of Belle data as a parameter sweep application is shown in Figure 5. The plan file follows Nimrod-G’s declarative parametric programming language which has been extended in this work by introducing a new type of parameter called “Gridfile”. This dynamic parameter describes a logical file location, either a directory or a collection of files and the broker resolves it to the actual file names and their physical locations. The plan file also instructs copying of user defined analysis modules and configuration files to the remote sites before any execution is started. The main task involves executing a user-defined shell script at the remote site which has 2 input parameters: the full network path to the data file and the name of the job itself. The shell script invokes BASF at the remote site to conduct the analysis over the data file and produce histograms. The histograms are then copied over to the broker host machine.

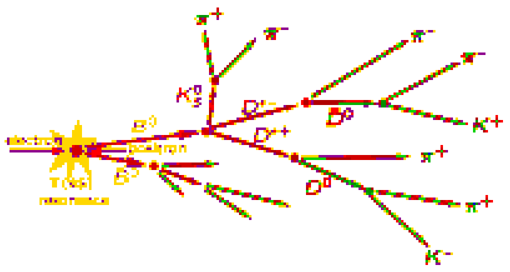
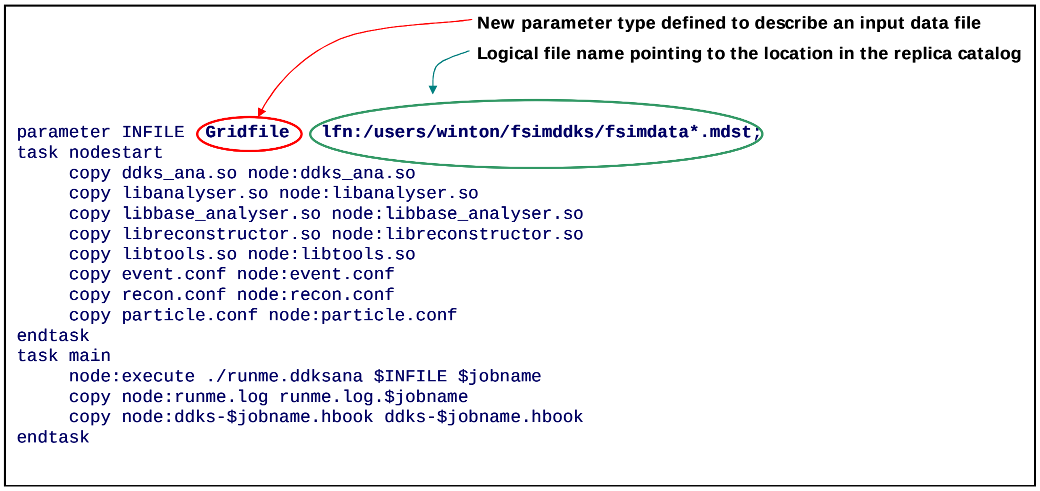


Figure 4: The B^0->D*+D*-Ks decay chain.

The Logical file name in this particular experiment resolved to 100 data files. Therefore, the experiment set consisted of 100 jobs, each dealing with the analysis of one data file using BASF. Each of these input data files was 30 MB in size. The entire data set was equally distributed among the five data hosts i.e. each of them has 20 data files each. The data was also not replicated between the resources, therefore, the dataset on each resource remained unique to it. The histograms generated were 968 KB in size.

For monitoring the bandwidth between the resources, an NWS sensor was started on each of the resources which reports to the NWS name server located in Melbourne. An NWS activity for


New parameter type defined to describe an input data file
Logical file name pointing to the location in the replica catalog

```
parameter INFILE Gridfile lfn:/users/winton/fsimddks/fsimdata*.mdst;
task nodestart
     copy ddks_ana.so node:ddks_ana.so
     copy libanalyser.so node:libanalyser.so
     copy libbase_analyser.so node:libbase_analyser.so
     copy libreconstructor.so node:libreconstructor.so
     copy libtools.so node:libtools.so
     copy event.conf node:event.conf
     copy recon.conf node:recon.conf
     copy particle.conf node:particle.conf
endtask
task main
     node:execute ./runme.ddksana $INFILE $jobname
     copy node:runme.log runme.log.$jobname
     copy node:ddks-$jobname.hbook ddks-$jobname.hbook
endtask
```

Figure 5: Plan file for Data Analysis

monitoring bandwidth was defined at the name server within which a clique containing all the resources on the testbed was created. Members of the clique conduct experiments one at a time to determine network conditions between them. Querying the name server at any point provided the bandwidth and latency between any 2 members of the clique.

4.3 Results of Evaluation

Three scheduling scenarios were evaluated: (1) scheduling with computation limited to only those resources with data, (2) scheduling without considering location of data, and (3) our adaptive scheduling (shown in) that optimizes computation based on the location of data. The experiments were carried out on April 19th, 2004 between 18:00 and 23:00 AEST. At that time, the Globus gatekeeper service on the Adelaide machine was down and so, it could not be used as a computational resource. However, it was possible to obtain data from it through GridFTP. Hence, jobs that depended on data hosted on the Adelaide server were able to be executed on other machines in the second and third strategies. A graph depicting the comparison of the total time taken for each strategy to execute all the jobs is shown in Figure 6 and another comparing resource performance for different scheduling strategies is shown in Figure 7.

In the first strategy (scheduling limited to resources with the data for the job), jobs were executed *only* on those resources which hosted the data files related to those jobs. No data transfers were involved in this scenario. As is displayed in the graph in Figure 7, all of the resources except the one in Adelaide were able to execute 20 jobs each. The jobs that were scheduled on that resource failed, as its computational service was unavailable. Hence, Figure 6 shows the total time taken for only 80 successful jobs out of 100. However, this time also includes the time taken by the scheduler to analyse the remaining 20 jobs as failed. In this setup, the related data was *exclusively* located on that resource and hence, these jobs were not reassigned to other compute resources. Thus, a major limitation of this scheduling strategy was exposed.

In the second strategy (scheduling without any data optimization), the jobs were executed on those nodes that have the most available computational resources. That is, there was no optimization based on location of data within this policy. The Adelaide server was considered a failed resource and was not given any jobs. However, the jobs that utilized data files hosted on this machine were able to be executed on other resources. This strategy involves the maximum amount of data transfer which makes it unsuitable for applications involving large data transfers and utilising resources connected by slow networks.

The last evaluation (scheduling with data optimization) was carried out by scheduling jobs to the compute resources that satisfied the algorithm given in Section 3.3. In this case, as there were no multiple data hosts for the same data, the policy was reduced to dispatching jobs to the best available compute resource that had the best available bandwidth to the host for the related data. It can be seen from Figure 7 that most of the jobs that accessed data present on the Adelaide resource were scheduled on the Melbourne Physics and CS resources because the latter had consistently higher available bandwidth to the former. This is shown in the plot of the available bandwidth from the University of Adelaide to other resources within the testbed measured during the execution, given in Figure 8. The NWS name server was polled every scheduling interval for the bandwidth measurements. As can be seen from Figure 6, this strategy took the least time of all three.

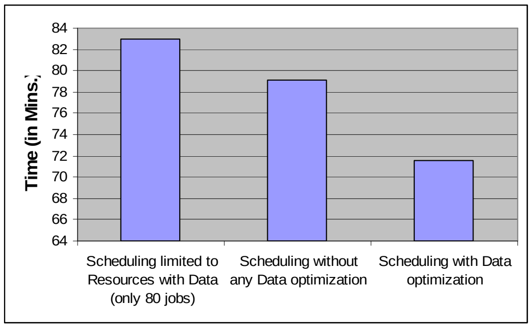


Figure 6: Total time taken for each scheduling strategy.

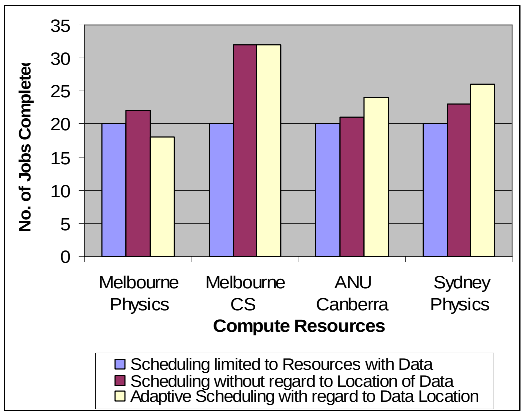


Figure 7: Comparison of resource performance under different scheduling strategies.

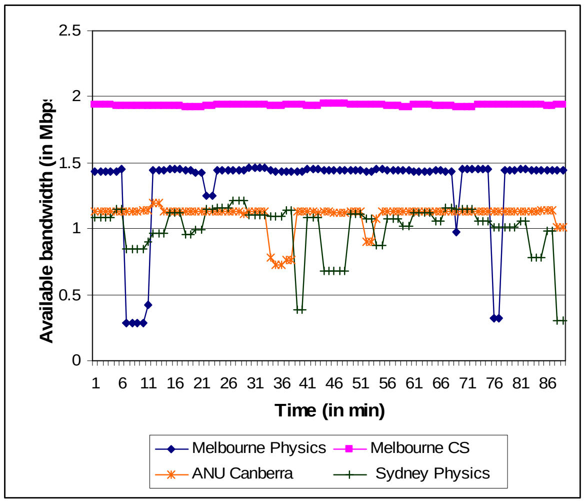


Figure 8: Available bandwidth from University of Adelaide to other resources in the testbed.

5. SUMMARY AND CONCLUSION

We have presented a grid broker for executing distributed data-oriented jobs on a grid. The broker discovers computational and data resources, schedules jobs based on optimization of data transfer and returns results back to the user. We have applied this broker to a data-intensive environment, which is the analysis of the Belle high-energy physics experiment data and have presented the results of our evaluation with different scheduling strategies. The proposed scheduling strategy took into consideration the network conditions and has produced the best possible outcome by executing the jobs within the least amount of time.

We plan to conduct further evaluations with larger file sizes and multiple repositories for the same datasets. This will ensure that the data transfer time becomes more significant while making scheduling decisions and that the scheduler will be able to choose between different data hosts.

ACKNOWLEDGEMENTS

This work is partially supported by Australian Research Council Discovery Project. We thank members of GRIDS Lab for their contribution towards the implementation.

REFERENCES

[1] D. Abramson, J. Giddy, and L. Kotler, *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?*, Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2000), May 1-5, 2000, Cancun, Mexico, IEEE CS Press, USA, 2000.

[2] J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke, *GASS: A Data Movement and Access Service for Wide Area Computing Systems,* Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems, Atlanta, GA, May 1999. ACM Press.

[3] J. Bunn and H. Newman, Data-intensive grids for high-energy physics, In *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and T. Hey, Eds. John Wiley & Sons, Inc., New York, 2003.

[4] R. Buyya, K. Branson, J. Giddy, and D. Abramson, The Virtual Laboratory: Enabling Molecular Modeling for Drug Design on the World Wide Grid, *Concurrency and Computation: Practice and Experience*, Volume 15, Issue 1, Pages: 1-25, Wiley Press, USA, January 2003

[5] R. Buyya, D. Abramson, and J. Giddy, *An Economy Driven Resource Management Architecture for Global Computational Power Grids*, Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications , June 26-29, 2000, Las Vegas, USA, CSREA Press, USA, 2000.

[6] R. Buyya and S. Venugopal, *The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report*, Proceedings of the First IEEE International Workshop on Grid Economics and Business Models (GECON 2004), April 23, 2004, Seoul, IEEE Press, USA

[7] H. Casanova, G. Obertelli, F. Berman, and R. Wolski, *The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid*, Proceedings of the IEEE SC 2000, International Conference Networking and Computing, Nov. 2000, Dallas, Texas, IEEE CS Press, USA.

[8] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, *Heuristics for scheduling parameter sweep applications in grid environments*, Proceedings of the 9th Heterogeneous Computing Workshop, 2000. (HCW 2000), Cancun, Mexico. IEEE CS Press, USA.

[9] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, *Journal of Network and Computer Applications*, vol. 23, no. 3, pp. 187–200, 2000.

[10] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, *Grid Information Services for Distributed Resource Sharing*, Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), IEEE CS Press, USA, 2001.

[11] I. Foster and C. Kesselman (editors), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999.

[12] I. Foster, C. Kesselman, and S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, *International Journal of High Performance Computing Applications*, vol. 15, pp. 200-222, Sage Publishers, London, UK, 2001.

[13] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Applications*, 11(2): 115-128, 1997.

[14] T. Hey and A. E. Trefethen, The UK e-Science Core Programme and the Grid, *Future Generation Computer Systems*, Volume 18, Issue 8, October 2002, Pages 1017-1031.

[15] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. F. Freund, Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems, *Journal of Parallel and Distributed Computing*, Volume 59, Issue 2, November 1999, Pages 107-131

[16] R. Moore, C. Baru, R. Marciano, A. Rajasekar, and M.Wan, The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1998, ch.5,"Data Intensive Computing".

[17] K. Ranganathan and I. Foster, *Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications*, Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002, IEEE CS Press, USA.

[18] W. Sudholt, K. Baldridge, D. Abramson, C. Enticott, and S. Garic, Parameter Scan of an Effective Group Difference Pseudopotential Using Grid Computing, *New Generation Computing,* Volume 22 , Pages:125-135, 2004.

[19] S. Vazhkudai, S. Tuecke, I. Foster, *Replica Selection in the Globus Data Grid*, Proceedings of the First IEEE/ACM International Conference on Cluster Computing and the Grid (CCGRID 2001), pp. 106-113, IEEE CS Press, May 2001.

[20] S. Venugopal, R. Buyya and L. Winton, A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids, *Technical Report*, GRIDS-TR-2004-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, February 2004.

[21] Lyle Winton, Data Grids and High Energy Physics: A Melbourne Perspective, *Space Science Reviews*, 107 (1-2): 523-540, Kluwer Academic Publishers, Netherlands, 2003

[22] R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing", *Journal of Future Generation Computing Systems*,Volume 15, Numbers 5-6, pp. 757-768.