Data Management in an International Data Grid Project

Wolfgang Hoschek[1,3], Javier Jaen-Martinez[1], Asad Samar[1,4], Heinz Stockinger[1,2], and Kurt Stockinger[1,2]

[1] CERN, European Organization for Nuclear Research, Geneva, Switzerland
[2] Inst. for Computer Science and Business Informatics, University of Vienna, Austria
[3] Inst. of Applied Computer Science, University of Linz, Austria
[4] California Institute of Technology, Pasadena, CA, USA

Abstract. In this paper we report on preliminary work and architectural design carried out in the "Data Management" work package in the International Data Grid project. Our aim within a time scale of three years is to provide Grid middleware services supporting the I/O-intensive world-wide distributed next generation experiments in High-Energy Physics, Earth Observation and Bioinformatics. The goal is to specify, develop, integrate and test tools and middleware infrastructure to coherently manage and share Petabyte-range information volumes in high-throughput production-quality Grid environments. The middleware will allow secure access to massive amounts of data in a universal namespace, to move and replicate data at high speed from one geographical site to another, and to manage synchronisation of remote copies. We put much attention on clearly specifying and categorising existing work on the Grid, especially in data management in Grid related projects. Challenging use cases are described and how they map to architectural decisions concerning data access, replication, meta data management, security and query optimisation. [1]

1 Introduction

In the year 2005 a new particle accelerator, the Large Hadron Collider (LHC), is scheduled to be in operation at CERN, the European Organization for Nuclear Research. Four High Energy Physics (HEP) experiments will start to produce several Petabytes of data per year over a life time of 15 to 20 years. Since this amount of data was never produced before, special efforts concerning data management and data storage are required.

One characteristic of these data is that most of it is read-only. In general, data are written by the experiment, stored at very high data rates (from 100 MB/sec to 1GB/sec) and are normally not changed any more afterwards. This is true for about 90% of the total amount of data. Furthermore, since CERN experiments are collaborations of over a thousand physicists from many different

[1] to appear in the IEEE, ACM International Workshop on Grid Computing (Grid'2000), 17-20 Dec. 2000, Bangalore, India

universities and institutes, the experiment's data are not only stored locally at CERN but there is also an intention to store parts of the data at world-wide distributed sites in so-called Regional Centres (RCs) and also in some institutes and universities. The computing model of a typical LHC experiment is shown in Figure 1.

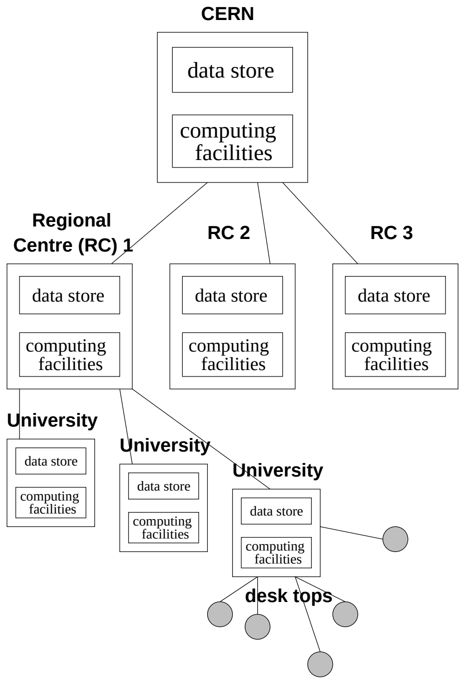

Fig. 1. Example of the network of one experiment's computing model

These RCs are part of the distributed computing model and should complement the functionality of the CERN Centre. The aim is to use computing power and data storage in these Regional Centres and allow physicists to do their analysis work outside of CERN with a reasonable response time rather than accessing all the data at CERN. This should also help the collaboration to have many scientists working spread around the world. Regional Centres will be set up for instance in Italy, France, Great Britain, USA and Japan.

By its nature, this is a typical Grid application which combines two aspects of today's Grid technology: Computational and Data Grids. In order to meet this challenge, the HEP community has established a project called "Research and Technological Development for an International Data Grid". The objectives of this project are the following. Firstly, establish a Research Network which will enable the development of the technology components essential for the implementation of a new world-wide Data Grid on a scale not previously attempted. Secondly, demonstrate the effectiveness of this new technology through the large scale deployment of end-to-end application experiments involving real users. Finally, demonstrate the ability to build, connect and effectively manage large general-purpose, data intensive computer clusters constructed from low-cost commodity components. Furthermore, the project does not only cover HEP but also other scientific communities like Earth Observation and Bioinformatics.

The entire project consists of several work packages for middleware development, computing fabric and mass storage management, testbeds and applications. In this paper we present the data management aspects of the project. The objectives are to implement and compare different distributed data management approaches including caching, file replication and file migration. Such middleware is critical for the success of heterogeneous Data Grids, since they rely on efficient, uniform and transparent access methods. Issues to be tackled within three years include:

- the management of a universal namespace
- efficient data transfer between sites
- synchronisation of remote copies
- wide-area data access/caching
- interfacing to mass storage management systems.

A major aim of the project is to build on existing experience and available software systems. For the startup phase we have chosen the Globus toolkit as the starting point for our middleware research and development. Globus is a promising toolkit and has already proved several times that it is applicable for large Grid projects [12].

The paper is organised in the following way. The section on related work gives an overview of data management in current data Grid projects and discusses related issues of distributed database management systems and distributed file systems. Section 3 emphases the challenging requirements of data-intensive Grid applications. In sections 4 and 5 we present the overall architecture of the data management middleware components and give details on the individual components. Finally, conclusions and future work are presented.

2 Survey and Discussion of Related Work

Traditional distributed file systems like Network File System (NFS) [18] and Andrew File System (AFS) [15] provide a convenient interface for remote I/O with a uniform file name space. However, this approach does not support multisite replication issues and also cannot achieve good performance due to a lack of collective I/O functionalities, i.e. batch I/O and scheduled I/O. In contrast, parallel file systems like Vesta [5] and Galley [16], provide collective I/O but do not address complex configurations, unique performance trade-offs and security problems that arise in wide area environments. Finally, remote execution systems enable location-independent execution of tasks scheduled to remote computers, but do not support parallel I/O interfaces or access to parallel file systems.

In distributed database research replication becomes more and more important. However, the research emphasis is on update synchronisation of single transactions on replicas [1] rather than considering the problem of transferring large amounts of data, which is an issue in our case.

None of these legacy systems are able to satisfy the stringent requirements, posed by both the scientific community and the industry, of having geographically distributed users and resources, accessing Petabyte-scale data and performing computationally intensive analysis of this data.

The notion of the "Grid" has been related to having access to distributed computational resources, resulting in being able to run computation intensive applications. The concept of having a Grid infrastructure which can support data intensive applications is new to the Grid community. There are a few projects, like Globus [4] and Legion [13], which were initially directed towards computational Grids but are now also adding support for distributed data management and integrating this with the computational Grid infrastructure. There is yet another class of on-going projects which have directed their efforts to support the distributed-data intensive applications from the very beginning. These mainly include Particle Physics Data Grid (PPDG)[17], Grid Physics Network (GriPhyN)[8], Storage Request Broker (SRB)[21] and the China Clipper[11] project.

The Global Access to Secondary Storage (GASS) API provided by Globus is the only component in the latest version of the toolkit which performs tasks related to data management. The scope of GASS API, however, is limited to providing remote file I/O operations, management of local file caches and file transfers in a client-server model with support for multiple protocols [3]. The Globus group is currently working on some of the data management issues including replica management and optimising file transfers over wide area networks [4]. The Globus philosophy is not to provide high level functionality, but to develop middleware which can be used as the base for developing a more complicated infrastructure on top.

The Legion project does not have any explicit modules working on data management issues. However, it does provide very basic data management functionality, implicitly, using the backing store "vault" mechanism [13]. High level issues like replica management, optimised file transfers and data load management are not addressed.

The Particle Physics Data Grid (PPDG) project is focussed on developing a Grid infrastructure which can support high speed data transfers and transparent access. This project addresses replica management, high performance networking and interfacing with different storage brokers [17]. This is a one year project and so the intentions are not to have very high level deliverables but to develop a basic infrastructure which can fulfill the needs of physicists.

The Grid Physics Network (GriPhyN) project is a new project whose proposal has been sent to NSF for approval in April 2000. The main goal of the project is to pursue an aggressive programme of fundamental IT research focussed on realising the concept of "virtual data" [8].

Storage Request Broker (SRB) addresses issues related to providing a uniform interface to heterogenous storage systems and accessing replicated data over wide area. SRB also provides ways to access data sets based on their attributes rather than physical location, using the Metadata Catalog (MCAT) [21]. MCAT is a meta data repository system, which provides a mechanism for storing and querying system level and domain independant meta data using a uniform interface [14]. The China Clipper project has its high level goals to support high speed access to, and integrated views of, multiple data archives; resource discovery and automated brokering; comprehensive real-time monitoring of networks and flexible and distributed management of access control and policy enforcement for multi-administrative domain resources [11]. The project goals cover most aspects of a Grid infrastructure and also addresses the middleware development and not only the high level services.

These are the main initiatives which are looking at data management issues in a distributed environment. One of the main goals of our project is to work in collaboration with these on-going efforts, and use the middleware developed by them if it satisfies our requirements. Our final work aims at a system which would integrate or interact with these projects so that end-users can benefit from the efforts being put in, from all over the globe.

3 Use Cases

In our Data Grid initiative three different real-world application areas are included:

- High Energy Physics (HEP)
- Earth Observation
- Bioinformatics

Common to all these areas is the sharing of data in terms of information and databases, which are distributed across Europe and even further afield. The main aim is to improve the efficiency and the speed of the data analysis by integrating widely distributed processing power and data storage systems. However, the applications offer complementary data models, which allow us to assess how well a given solution can be applied to a general-purpose computing environment.

HEP is organised as large collaborations where some 2,000 researchers distributed all over the globe analyse the same data, which are generated by a single, local accelerator. The data access itself is characterised by the generalised dynamic distribution of data across the Grid including replica and cache management. As for Earth Observation, data are collected at distributed stations and are also maintained in geographically distributed databases. In molecular biology and genetics research a large number of independent databases are used, which need to be integrated in one logical system.

In order to get a better understanding of some of the requirements for the Data Grid, let us briefly outline the general characteristics of HEP computing. Experiments are designed to try to understand the physical laws of nature and to test the existing models by studying the results of the collisions of fundamental particles, which are produced after acceleration to very high energies in large particle accelerators. For example, beams of protons are accelerated in opposite directions and are forced to collide in all detectors along the accelerator. Each of these collisions is called an *event*. The detectors track the passage of produced particles. Moreover, the analysis of physical contraints of the produced particles implies computationally intensive reconstruction procedures.

Typical uses in HEP fall into two main categories, namely data production and end-user analysis:

1. Data production
   - central experimental data production at CERN (these data come directly from the on-line data acquisition system of the detector)
   - distributed event simulation
   - reconstruction of event data
   - partial re-reconstruction of event data
2. End-user analysis
   - interactive analysis
   - batch analysis on fully reconstructed data
   - analysis on full event data including "detector studies"

A typical interactive end-user analysis job starts with selecting a large initial collection of independent events. This means that the physics result obtained by processing the event collection is independent of the sequence of processing each single event. During the analysis jobs physicists apply some "cuts" on the data and thereby reduce the number of events in the event collection. In other words, a cut predicate is developed which is applied to the event collection in order to sieve out "interesting" events.

The process of constructing single cut predicates, i.e. optimisations of physics selections, can take several weeks or months, where the current version of the cut predicate is applied to the whole event collection or to subsets of it. One obvious optimisation for such an analysis job is to keep the most frequently used subset of events on the fastest storage (for example, in the disk cache).

An analysis job can possibly compute some very CPU intensive functions of all events, for example, a reconstruction algorithm could create a complex new event object which has to be stored for later analysis. This new object can be regarded as some additional information for this particular event.

Other jobs could apply multiple functions to every event. However, a considerable amount of time is spent on reading the objects, i.e. fetching the objects from the disk cache or from tape. Since all events are independent, a coarse grained parallelism based on the event level allows for a high degree of freedom in I/O scheduling and thus the events can be processed on different CPUs in parallel.

The Data Management tasks are to handle uniform and fast file transfer from one storage system to another. What is more, by studying the access patterns, meta data and file copies need to be managed in a distributed and hierarchical cache. In addition, security issues and access rights of the particular users must be considered.

4 Architecture

The Data Grid is a large and complex project involving many organisations, software engineers and scientists. Its decomposition must meet a number of challenges. The architecture must

- be easy to understand in order to be maintainable over time. Complex and fragile components are discouraged.
- be flexible so that different organisations can plug-in their own packages. A model based on a layered set of interfaces enables multiple implementations to coexist. Each implementation of an interface may focus on different characteristics such as performance or maintainability.
- allow for rapid prototyping. Thus it should leverage previous work as much as possible.
- be scalable to massive high throughput use cases. Careful design and layering is necessary to achieve this.
- respect the nature of distributed development. Effort is split between multiple teams, each working on a substantial component. Therefore components must be well defined and loosely coupled.

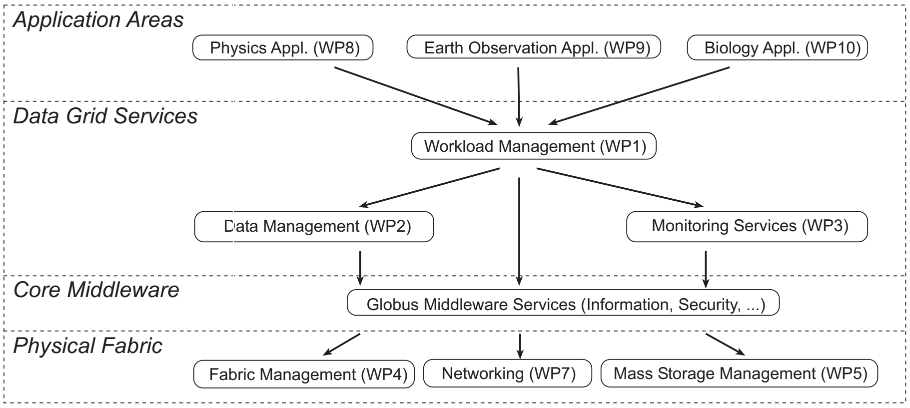

Fig. 2. Overall interaction of project work packages

We now sketch the overall architecture of the Data Grid as depicted in Figure 2. WP indicates the work package within the entire project. *High Energy Physics*, *Earth Observation*, and *Biology* exploit the developments of the project to offer transparent access to distributed data and high performance computing facilities to their respective geographically distributed community. *Workload Management* defines and implements components for distributed scheduling and resource management. *Data Management* develops and integrates tools and middle-ware infrastructure to coherently manage and share Petabyte-scale information volumes in high-throughput production-quality Grid environments. *Monitoring* provides infrastructure to enable end-user and administrator access to status and error information in a Grid environment. *Globus* services form the core middleware. *Fabric Management* delivers all the necessary tools to manage a computing centre providing Grid services on clusters of thousands of nodes. The management functions must uniformly encompass support for everything from the compute and network hardware up through the operating system, workload and application software. The *Networking* work package uses the European and national research network infrastructures to provide a virtual private network between the computational and data resources forming Data Grid testbeds. *Mass Storage Management* interfaces existing Mass Storage Management Systems (MSMS) to the wide area Grid data management system.

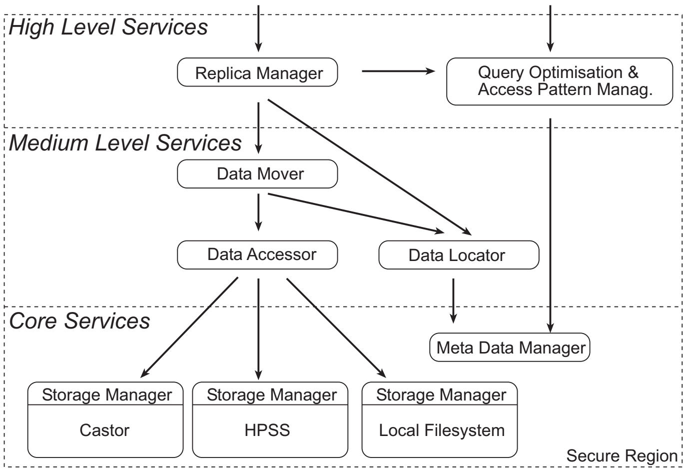

Fig. 3. Overall interaction of data management components

The Data Management work package which is our primary concern in this paper consists of a layered set of services as shown in Figure 3. Arrows indicate "use" relationships. Component A uses component B to accomplish its responsibilities. The *Replica Manager* manages file and meta data copies in a distributed and hierarchical cache. It uses and is driven by plugg-able and customisable replication policies. It further uses the *Data Mover* to accomplish its tasks. The data mover transfers files from one storage system to another one. To implement its functionality, it uses the *Data Accessor* and the *Data Locator*, which maps location independent identifiers to location dependent identifiers. The Data Accessor is an interface encapsulating the details of the local file system and mass storage systems such as Castor [2], HPSS [10] and others. Several implementations of this generic interface may exist, the so-called *Storage Managers*. They typically delegate requests to a particular kind of storage system. Storage Managers are outside the scope of this work package. The Data Locator makes use of the generic *Meta Data Manager*, which is responsible for efficient publishing and management of a distributed and hierarchical set of objects. *Query Optimisation and Access Pattern Management* ensures that for a given query an optimal migration and replication execution plan is produced. Such plans are generated on the basis of published meta data including dynamic monitoring and configuration information. All components provide appropriate *Security* mechanisms that transparently span worldwide independent organisational institutions.

5 Data Management Components

5.1 Data Accessor

One of the core problems that any data management system has to address is the heterogeneity of repositories where data are stored. This is even more of a critical aspect when data management has to be targeted in a wide area network environment. The main problem to be solved is the variety of possible storage systems. These can be either mass storage management systems like HPSS, Castor, UniTree [22], and Enstore [7], multiple disk storage systems like DPSS [6], distributed file systems like AFS, NFS [18], or even databases. This diversity is made explicit in terms of how data sets are named and accessed in all these different systems. For instance, in some cases data are identified through a file name whereas other systems use catalogues where data are identified and selected by iterating over a collection of attributes or by using an object identifier.

To limit the scope of our initial work we will concentrate on data collections that are stored in either Hierarchical Storage Management (HSM) or local file sytems, leaving aside the extremely complex case of homogeneous access to data stored in different database systems. We are targeting specially HSMs because they provide an automatic and transparent way of managing and distributing data across a storage hierarchy that may consist of tapes, compressed disk and high-performance disk. This type of storage system is vital for HEP applications where the volume of data generated requires the use of tapes as a cost-effective media, and where data access requirements range from accessing it many times an hour during analysis to accessing it very infrequently ("cold" data).

Having these assumptions in mind, the problem to be solved within this component of our system is the definition of a single interface that can be used by higher level modules to access data located in different underlying repositories.

Thus, this module will have to make the appropriate conversions for Grid data access requests to be processed by the underlying storage system and to prepare the underlying storage system to be in the best condition to deliver data in a Grid environment. For HSMs, strategies like when data should be staged to a local disk cache before a Grid transfer is triggered, what requests are queued together to get the best performance in terms of tape mounts, and when files in the local cache are released to free space for new incoming requests will be performed by this subsystem in close coordination with the facilities provided by the storage system (existing internal catalogues, mechanisms for data transfer between tapes and local disks, etc).

In summary, this subsystem will hide from higher layers the complexities and specific mechanisms for data access which are particular to every storage system manipulating the performance factors which are proprietary for each system.

5.2 Replication

Replication can, on the one hand, be regarded as the process of managing copies of data. On the other hand, replication is a caching strategy where identical files are available at multiple places in a Grid environment. The main purpose of replication is to gain better response times for user applications by accessing data from locally "cached" data stores rather than to transfer each single requested file over the wide area network to the client application. Fault tolerance, and hence the availability of data, are key items of replication. Replication yields performance gains for read operations since a client application can read data from the closest copy of a file. Update and hence write operations need to be synchronised with other replicas and thus have a worse performance than updates on single copies. The performance loss of replication depends on the update protocols and network parameters of the Grid.

The problem of data replication not only involves the physical transfer of data among sites and the update synchronisation among the different available copies, but is also related to the more complex problem of deciding which are the policies or strategies that should trigger a replica creation. In a Grid environment replication policies are clearly not enforced by a single entity. As an example, system administrators can decide for production requirements to distribute data according to some specific layout, schedulers may require a particular data replication schema to speedup execution of jobs, and even space constraints or local disk allocation policies may force certain replicas to be purged. Therefore, the replication subsystem needs to provide adequate services for task schedulers, Grid administrators, and even local resource managers within clusters to be able to replicate, maintain consistency, and obtain information about replicas to be able to enforce any required policies.

The replication domain includes data and meta data to be replicated. These impose different requirements on the underlying communication system in the Grid. Since we are dealing with Petabytes of data that have to be transferred over the network to Regional Centres, there is an essential requirement for fast point-to-point file replication mechanisms for bulk data transfer. However, in case of limited available bandwidth and limited efficiency (not all the theoretical bandwith is available) a solution that replicates everything everywhere may not be feasible.

The meta data replication requires a client-server communication mechanism at each Grid site. The Globus toolkit offers two possibilities: sockets and a more high level communication library called *Nexus*. The communication subsystem is required to implement different replication protocols like synchronous and asynchronous update methods. An important input factor for the decision of the underlying update mechanism is data consistency. A detailed survey of replica updates can be found in [19].

Once data are in place, the Data Locator is responsible for accessing physical files, mapping location independent to location dependent identifiers. This mapping is required in order to enable transparent access to files within a uniform namespace.

User requests are not directly handed to the Data Accessor, but are routed through the Replica Manager. The Replica Manager provides high level access services and optimises wide-area throughput via a *Grid cache*. It is an "intelligent" service that knows about the wide-area distribution of files. It analyses user access patterns in order to find out where and how files are to be accessed in optimal ways. As a consequence of these access pattern analysis, replicas are created and purged at remote sites.

The Data Accessor simply accesses files which are selected by the Replica Manager. With the Replica Manager taking care of wide-area caching, the mass storage system at each site is responsible for the *local caching* of files.

5.3 Meta Data

The glue for components takes the shape of a Meta Data Management Service. Particularly interesting types of meta data are:

- Catalogues comprising names and locations of unique and replicated files, as well as indexes.
- Monitoring information such as error status, current and historical throughput and query access patterns.
- Grid configuration information describing networks, switches, clusters, nodes and software.
- Policies enabling flexible and dynamic steering.

The key challenge of this service is to integrate diversity, decentralisation and heterogeneity. Meta data from distributed autonomous sites can turn into information only if straightforward mechanisms for using it are in place.

The service manages a large number of objects referred to by identifiers. Respecting the loosely coupled nature of the Grid, it must allow for maintainance of autonomous partitions *and* good performance, both over the LAN and WAN. Thus, the service is build on a fully distributed hierarchical model and a versatile and uniform protocol, such as Lightweight Directory Access Protocol (LDAP)

[23]. Multiple implementations of the protocol will be used as required, each focussing on different trade-offs in the space spanned by write/read/update/search-performance and consistency.

5.4 Security

Certain security aspects of a Grid infrastructure are tightly coupled to Data Management. Identifying these issues, and adapting the Data Management components accordingly, is of vital importance. Some of these issues are discussed here.

An important global security issue is to deal with the Grid cache. The site which owns the data has to make sure that the re mote sites hosting its data caches provide the same level of security as the owner requires for their data. This will be a serious issue when dealing with sensitive data where human or intellectual property rights exist. The fact that different sites will probably be usi ng different security infrastructures will result in more complications. I t is, therefore, required to evaluate strategies and develop tools which c an be used to ensure the same level of security with heterogenous underlyi ng security infrastructures.

Synchronous replication strategies, instead of using an on-demand or time scheduled approach, raise a lot of security concerns for the participating sites. A synchronous solution would involve giving time indefinite write permissions to other nodes in the Grid, so that whenever a replica is updated or deleted, the same operation can be propagated to all the remote replicas. An on-demand or time scheduled solution (asynchronous) is more secure and less consistent, though not as responsive.

The replica selection will depend on many factors, including the security policies of the nodes which contain these replicas. We may want to select a replica from a node which is more "friendly" as compared to one which forces more access restrictions.

The sensitivity levels associated with data and meta data might be different. The actual data might be more sensitive for some sites than their meta data or vice versa. This difference has to be incorporated in the overall design of the Data Management as well as the security system.

Several policy matters which are expected to vary from site to site include

- the usage of a synchronous replication strategy or something more secure,
- the importance of meta data as compared to real data in terms of security
- how much weight to be given to security when selecting a replica.

The intention is not to force all the sites in the Grid to agree on a common policy, but to design a system which is flexible enough to absorb heterogeneity of policies and present a consistent yet easy-to-adapt solution.

5.5 Query Optimisation

Queries are one way for an application user to access a data store. In a distributed and replicated data store a query is optimised by considering multiple copies of a file. A set of application queries is considered to be optimally executed if it minimises a cost model such as a mixture of response time and throughput.

The aim of a query execution plan is to determine which replicated files to access in order to have minimal access costs. We do not want to elaborate much further on a cost model here. However, the optimal query execution plan is based on static and dynamic influences like the following [9]:

- size of the file to be accessed
- load on the data server to serve the requested file
- method/protocol by which files are accessed and transferred
- network bandwidth, distance and traffic in the Grid
- policies governing remote access

The outcome of a query can be either the result of the query itself or a time estimate of how long it takes to satisfy a query. The Meta Data Management service will be used to keep track of what data sets are requested by users, so that the information can be made available for this task.

Query optimisation can be done at different levels of granularity. The method stated above is only based on files and also requires a set of files as an input parameter to the query execution plan. Often files have a certain schema and users query single objects of a file. This requires an additional instance that does the mapping between object identifiers (OID) and files by using a particular index [20] which satisfies the expected access patterns. This introduces another level of complexity for the query optimisation because objects can be available in multiple copies of files and the optimal set of files has to be determined to satisfy the query. Note that the OID file mapping is not an explicit task of the Data Management work package and needs additional information from Workload Management and Application Monitoring.

6 Conclusion and Future Work

In this paper we reported on preliminary work and architectural design which has been carried out in the work package "Data Management" in an International Data Grid project which has been proposed recently. We motivated our Data Grid approach by a detailed discussion and categorisation of existing work on the Grid, especially of data management in Grid related projects. The aim of our three year project is to provide Grid middleware services for the scientific community dealing with huge amounts of data in the Petabyte range, whereas the essential goal is to support world-wide distributed real-world applications for the next generation experiments in High Energy Physics, Earth Observation and Bioinformatics.

Basing the initial work on Globus, we have a Globus test bed running and preliminary promising prototypes are being implemented and tested. First results will be available by the end of the year. Furthermore, we are in close contact with the Globus developers concerning evolving Data Grid ideas and implementations.

Acknowledgements

We would like to thank Les Robertson for bootstrapping this interesting work, and Ben Segal for valuable feedback and contributions to the paper. Thank you to all Data Grid members for their interesting discussions.

References

1. T. Anderson, Y. Breitbart, H. Korth, A. Wool. Replication, Consistency, and Practicality: Are These Mutually Exclusive? Proc. SIGMOD International Conference on the Management of Data, pp. 484-495 1998.
2. O. Barring, J. Baud, J. Durand. CASTOR Project Status, Proc. of Computing in High Energy Physics 2000, Padova, Febr. 2000.
3. J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke. GASS: A Data Movement and Access Service for Wide Area Computing Systems. In Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems, May 1999.
4. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific DataSets. Network Storage Symposium, Seattle 1999.
5. P. Corbett and D. Feitelson. Design and Implementation of the Vesta Parallel File System. In Proceedings of the Scalable High-Performance Computing Conference, pages 63-70, 1994.
6. DPSS: Distributed Parallel Storage System, http://www-itg.lbl.gov/DPSS/
7. Enstore: http://www-isd.fnal.gov/enstore/design.html
8. GriPhyN: Grid Physics Network, http://www.phys.ufl.edu/ avery/mre/
9. K. Holtman, H. Stockinger. Building a Large Location Table to Find Replicas of Physics Objects. Proc. of Computing in High Energy Physics 2000, Padova, Febr. 2000.
10. HPSS: High Performance Storage System, http://hpcf.nersc.gov/storage/hpss/
11. W. Johnston, J. Lee, B. Tierney, C. Tull, D. Millsom. The China Clipper Project: A Data Intensive Grid Support for Dynamically Configured, Adaptive, Distributed, High-Performance Data and Computing Environments. Proc. of Computing in High Energy Physics 1998, Chicago 1998.
12. W. Johnston, D. Gannon, B. Nitzberg. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. Eighth IEEE International Symposium on High Performance Distributed Computing, Redondo 1999.
13. LEGION: http://www.cs.virginia.edu/ legion/
14. MCAT: A Meta Information Catalog, http://www.npaci.edu/DICE/SRB/mcat.html
15. J. Morris, et al. Andrew: A Distributed Personal Computing Evironment. Comms. ACM, vol 29, no. 3, pp. 184-201, 1996.

16. N. Nieuwejaar, D. Kotz. The Galley Parallelle System. In Proceedings of the 10th ACM International Conference on Supercomputing, pages 374-381, Philadelphia, ACM Press, May 1996.
17. PPDG: Particle Physics Data Grid, http://www.cacr.caltech.edu/ppdg/
18. R. Sandberg. The Sun Network File System: Design, Implementation and Experience, Tech. Report, Mountain View CA: Sun Microsystems, 1987.
19. H. Stockinger, Data Replication in Distributed Database Systems, CMS Note 1999/046, Geneva, July 1999.
20. K. Stockinger, D. Duellmann, W. Hoschek, E. Schikuta. Improving the Performance of High Energy Physics Analysis through Bitmap Indices. To appear in 11th International Conference on Database and Expert Systems Applications, London - Greenwich, UK, Springer Verlag, Sept. 2000.
21. SRB: Storage Request Broker, http://www.npaci.edu/DICE/SRB/
22. UniTree: http://www.unitree.com/
23. W. Yeong, T. Howes, S. Kille. Lightweight Directory Access Protocol, RFC 1777. Performance Systems International, University of Michigan, ISODE Consortium, March 1995.