# MANAGED DATA STORAGE AND DATA ACCESS SERVICES FOR DATA GRIDS

M. Ernst, P. Fuhrmann, T. Mkrtchyan, DESY, Hamburg, Germany
J. Bakken, I. Fisk, T. Perelmutov, D. Petravick, FNAL, Batavia, IL 60510, USA

*Abstract*

The LHC needs to achieve reliable high performance access to vastly distributed storage resources across the network. USCMS has worked with Fermilab-CD and DESY-IT on a storage service that was deployed at several sites. It provides Grid access to heterogeneous mass storage systems and synchronization between them. It increases resiliency by insulating clients from storage and network failures, and facilitates file sharing and network traffic shaping.

This new storage service is implemented as a Grid Storage Element (SE). It consists of dCache, jointly developed by DESY and Fermilab, as the core storage system and an implementation of the Storage Resource Manager (SRM), that together allow both local and Grid based access to the mass storage facilities. It provides advanced accessing and distributing collaboration data.

USCMS is using this system both as Disk Resource Manager at the Tier-1 center and at multiple Tier-2 sites, and as Hierarchical Resource Manager with Enstore as tape back-end at the Fermilab CMS Tier-1 center. It is used for providing shared managed disk pools at sites for streaming data between the CERN Tier-0, the Fermilab Tier-1 and U.S. Tier-2 centers.

Applications can reserve space for a time period, ensuring space availability when the application runs. Worker nodes without WAN connectivity can trigger file replication from a central repository to the local SE and then access data using POSIX-like file system semantics via the LAN. Moving the SE functionality off the worker nodes reduces load and improves reliability of the compute farm elements significantly.

## INTRODUCTION

While standard Grid infrastructures provide distributed scientific communities with the ability to collaborate and share resources, additional capabilities are needed to cope with the specific challenges associated with scientists accessing and manipulating very large distributed data collections. These collections, ranging from terabytes (TB) to petabytes (PB), comprise raw (measured) and many levels of processed or refined data as well as comprehensive metadata describing, for example, under what conditions the data was generated, how large it is, etc. New protocols and services must facilitate access to significant tertiary (e.g. tape) and secondary (disk) storage repositories to allow efficient and rapid access to primary data stores, while taking advantage of disk caches that buffer very large data flows between sites.

## DATA GRIDS

The computational and data management problems encountered in data intensive research include the following challenging aspects:

**Computation-intensive as well as data-intensive:** Analysis tasks are compute-intensive and data-intensive and can involve hundreds or even thousands of computer, data handling, and network resources. The central problem is coordinated management of computation and data, not just data movement.

**Data and resource sharing:** The data Grid has been introduced as a unifying concept to describe the new technologies required to support next-generation data-intensive applications. Data Grids are typically characterized by the following elements:

1. they layer sophisticated new services on top of existing local mechanisms and interfaces, facilitating coordinated sharing of remote resources, and
2. they provide a new dimension of transparency in how computational and data processing are integrated to provide data products to user applications. This transparency is vitally important for sharing heterogeneous distributed resources in a manageable way.

## THE CMS DATA GRID

CMS faces computing challenges of unprecedented scale in terms of data volume, processing requirements, and the complexity and distributed nature of the analysis and simulation tasks among thousands of physicists worldwide. The data storage rate is expected to grow in response to higher luminosity, new physics triggers and better storage capabilities, leading to data collections of 20-30 PB by 2010 rising to several hundred petabytes over the following decade.

The Data Grid architecture comprises two general classes of components:

1. Resource services that provide Data Grid applications with access to individual Grid resources. Data Grids exploit standard protocols but benefit from specialized behaviours in the Fabric elements to which these services provide access.
2. Higher-level collective services that support the management and coordinated use of multiple

resources. Here, Data Grids introduce specialized requirements, in such areas as catalog services, replica management services, community policy services, coherency control mechanisms for replicated data and replica selection mechanisms.

## DATA SERVICES

We now examine the basic resource-level services required for managing and accessing data sources.

Within the scope of this work issues relating to data access and the rendering of data sources are considered as Grid services. Note that while data may reside physically on a variety of devices, we are concerned here primarily with the interfaces presented to users wishing to access data stored on the device and the performance characteristics advertised for those access methods.

### *GridFTP as a File Access Service*

GridFTP [1] was designed as a fundamental data access and data transport service. Its designers sought to create a protocol that would provide a uniform interface to various storage systems, and storage brokers. Incompatible data access protocols used by these storage systems effectively partition the datasets available on the Grid. GridFTP was designed with the assumption that it would be mutually advantageous to both storage providers and users to have a common but extensible underlying data transfer protocol that provides interoperability between disparate storage systems. To facilitate interoperation with other Grid services, GridFTP uses the widely deployed Grid Security Infrastructure (GSI) for robust and flexible authentication, integrity, and confidentiality.

GridFTP can be used both to access specific data values and to move data blobs from point to point. Its use as a data transport protocol is facilitated by third-party control of data transfer that allows a user or application at one site to initiate, monitor, and control a data transfer operation between two other sites. Third party transfers with GridFTP require the client to establish a control connection to both the source and the destination. However, in many Grid installations today, this is not supported, since worker nodes installed on private networks do not have outbound connectivity to the public Internet.

### *Data Access and Integration*

Data access functionality in GridFTP is primarily directed toward file-oriented structured data with access primitives to return subsections of files. Although FTP's extended data commands do make it possible to perform "get" operations with complex specifications, a higher-level, more direct query interface is desirable and would facilitate more uniform access to data sources.

## MANAGING DATA SOURCES

Moving from data access to resource management functions used to manage the storage systems holding the data collections being accessed, we are interested, in particular, in managing storage space, which requires not only storage quotas, but also storage reservations, which ensures that agreed-upon amounts of storage are available for a specified duration of time. In addition we may also want to manage bandwidth and data throughput. Any data source must incorporate monitoring and auditing services to allow local and remote entities to keep track of the resource's state (e.g. available free space), monitor the progress of individual operations, and track resource consumption of resources by users and services.

### *dCache*

dCache [4] is a software-only Grid storage appliance jointly developed by DESY and Fermilab. To be useful in a Grid environment, storage appliances must provide two features. First, they must be able to make guarantees about storage availability so that wide-area schedulers can move large datasets without fear of resource revocation. Second, they must be self-cleaning to ensure that failed operations or misbehaving clients do not permanently fill the storage appliance and prevent other users from accessing it.

dCache has proven to be capable of managing the storage and exchange of hundreds of terabytes of data, transparently distributed among dozens of disk storage nodes. One of the key design features is that, although the location and multiplicity of data is autonomously determined by the system, based on configuration, CPU load and disk space, the name space is uniquely represented in a single file system tree. The system has shown to significantly improve the efficiency of connected tape storage systems, through caching, i.e. gather & flush, and scheduled staging techniques. Furthermore, it optimizes the throughput to and from data clients as well as smoothening the load of the connected disk storage nodes by dynamically replicating files upon the detection of hot spots. The system is tolerant against failures of its data servers, allowing administrators to go for commodity disk storage components. Access to data is provided by various FTP dialects, including GridFTP, as well as a proprietary protocol, offering POSIX-like file system operations like open/read, write, seek, stat, close.

### *Storage Resource Managers*

While the basic dCache components incorporate management functions into the storage system implementation, storage resource manager (SRM) [2] services manage associated storage via a range of techniques including cache management, pre-allocation and advance reservation of space for data transfers, staging of data from slow to fast storage in a hierarchical storage system, and scheduling of storage system requests. SRMs can also manage the file content of shared temporary space and use replacement policies to maximize file sharing. Since the functionality described here is essential for managed data storage and data access services a v1.1 compliant SRM component has been tightly integrated with the dCache.

A SRM typically manages two types of shared resources, files and space. Its basic function is to allocate space to a client upon demand, check that the client has permission to use that space, assign the space to the client for a period of time according to its policy, and release the space either when the client requests its release or when the lifetime assigned to the space expires. When a file is moved into the space managed by SRM, the file can be pinned for a period of time to ensure that the space occupied by the file is not reclaimed before the client that requested the file can complete its operations on that file. By allowing subsequent requests for the same file from different clients to extend the duration of the pinning, the SRM promotes the sharing of files among clients of the storage system. In such cases, the SRM keeps track of the pin lifetime expiration and space consumed by the file with each user for each file.

SRMs support file replication between Storage Elements (SE) which are carried out as third-party transfers. Unlike with GridFTP, where the client has to be able to connect to both the source and the destination, SRMs offer credential delegation between the client and the local SE. There is no need for a client running on a Worker Node for having full Internet connectivity.

A SRM that manages a disk cache is referred to as a disk resource manager (DRM), to distinguish it from an SRM that manages access to a hierarchical mass storage system, which is called a hierarchical resource manager (HRM). SRMs are designed to communicate with each other using the same interfaces the client uses. DRMs have been designed to manage files on behalf of the client even when they do not support explicit space reservation. In the following scenario a client requests 500 files but can start processing as soon as one file is available. The client submits a single request for the 500 files to a local DRM that manages a shared disk pool manager. The DRM queues the request, allocates a default space to the client, and checks whether any of the files are in its disk cache. As soon as the first file is available the client starts processing and releases each file when it is done, so that the DRM can put another file into the released space. If a requested file is not in the disk cache, the DRM uses the information to get the file from its source location, using data transport protocols such as GridFTP. Once the application is done with the file, space is released either by the client explicitly or automatically with the expiration of the lifetime of the pin. Thus, DRMs perform automatic garbage collection. In order to maximize the sharing benefits files are not removed from the disk cache unless space is needed.

HRMs are basically DRMs associated with a mass storage system (MSS), such as Enstore [3]. HRMs are designed to queue multifile requests for putting and getting files into and out of MSSs. They can make such requests concurrently but usually limit the number of concurrent requests to avoid overloading the MSS and other resources like data servers and the networks. Because they maintain a queue, they can reorder file requests so that files stored on the same tape can be read at the same time. This capability avoids unnecessary dismounts and mounts of tape cartridges. By using information collected by their monitoring capabilities HRMs can recover from temporary MSS failures.

## CMS DATA CHALLENGE DC04

In the following paragraph the CMS Data Challenge DC04 [6] is taken as a representative example of an application that requires storage management as it is described above.

DC04 is one of the milestones of the experiment scoped to ensure the experiment is ready with its global data distribution and analysis systems for the start of data taking at the Large Hadron Collider at CERN in 2007. The performance metrics for DC04 were to provide a baseline to give the experiment input to the Physics and Computing Technical Design Reports in the next two years. The goal of DC04 was to perform at 25% of the throughput needed at the start of data taking in 2007 (5% of the full LHC rate) including distribution of data from CERN to the Tier-1 regional centers for 2 months. Additional goals were to provide the software infrastructure to manage and distribute the data, to provide an end-to-end demonstration of event reconstruction and analysis, including the full chain of data processing, to show the state of the experiment's readiness of the software infrastructure.

A major focus of DC04 was to provide sustained and effective transfer of data between the CERN Tier-0 and distributed Tier-1 facilities across Europe and the US.

The setup used by USCMS comprises an SRM/dCache based DRM at CERN serving as an Export Buffer for data to be sent to Fermilab and an SRM/dCache/Enstore based HRM at Fermilab. Data received by the HRM is transparently migrated off the caching disk to permanent storage (tape).

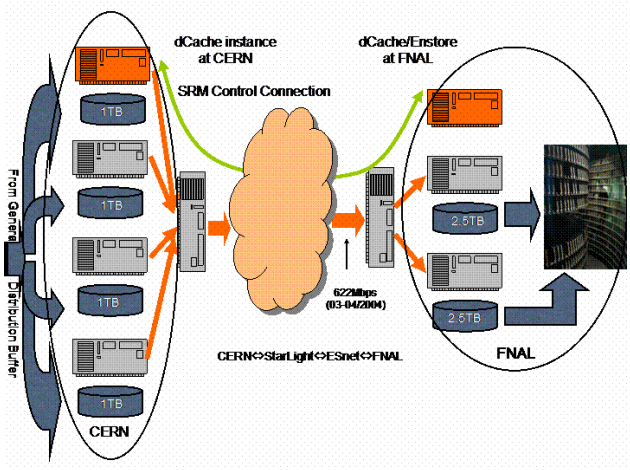


Figure 1: SRM/dCache data distribution system used in DC04

A typical data sample was transferred to Fermilab: 5.2 TB was transferred in 440k files. The average event size for the data challenge was about 100kByte, a factor 5 less than the final event size for the experiment. As a result the

size of the files for distribution was too small to allow efficient management and data transfer. Despite the flood of small files the average transfer rate achieved for long periods was over 10 MBps, peaking at over 20 MBps.

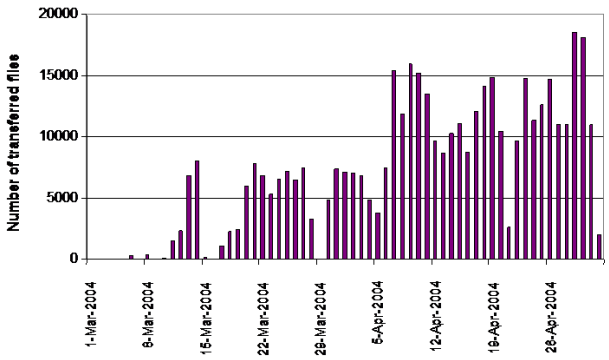


Figure 2: Number of daily transferred files to Fermilab

This proved a challenging application of grid technologies for data management, with DESY/Fermilab SRM/dCache (among others) proving ultimately capable after integration and debugging.

## GRID SCHEDULING AND STORAGE RESOURCE MANAGEMENT

The Grid is supposed to be capable of including various kinds of *devices* or *resources* such as computers, networks, data, software, storage etc. For local usage, most systems have job management systems, typically queuing systems, which are responsible for executing the job. However, in the context of the Grid coordination of multiple remote system entities is required. Therefore a Grid scheduler needs essentially to interact with lower level scheduling systems. Today, most of these local resource management systems just react to requests in a best effort fashion according to some priority list scheduling without additional information or guarantees when the requested resource will actually be available. This needs to be augmented by features like reservations for storage, network and CPU resources, deadlines or estimates about availability of data or the execution schedule. Grid scheduling should include capabilities that allow *planning* of the job execution.

Uniform access to a variety of resource types will ultimately require that Grid scheduling is capable of coordinating the allocation of these resources for Grid jobs.

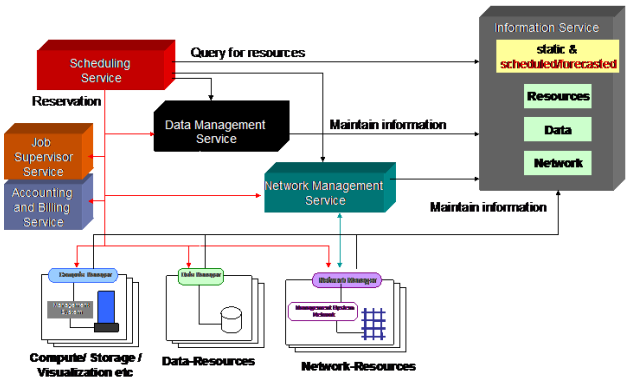


Figure 3: Basic Grid scheduling architecture

A working group under the umbrella of the GGF, called *Grid Scheduling Architecture Research Group (GSA-RG)*, has been approved and is actively pursuing the goals mentioned above. The dCache collaboration is planning to take part in this work in the context of the D-Grid initiative, the latter being an e-science program in Germany.

## REFERENCES


[1] The Globus Project, http://www.globus.org/datagrid/gridftp.html

[2] The SRM Project, http://sdm.lbl.gov/srm-wg/

[3] The Enstore Mass Storage System at Fermilab, http://computing.fnal.gov/docs/products/enstore/

[4] The dCache Project, http://www.dcache.org

[5] The GGF Research Group on Grid Scheduling Architectures, http://www-ds.e-technik.uni-dortmund.de/~yahya/ggf-sched/WG/arch-rg.html

[6] The CMS DC04 Data Challenge http://www.uscms.org/s&c/dc04/