

Computing and Data Management for CMS in the LHC Era

Koen Holtman¹, Frank van Lingen^{2,3,5}, Heinz Stockinger^{4,5} and Ian Willers⁵

¹California Institute of Technology, Pasadena, USA;

²University of West England,

³Eindhoven University of Technology, The Netherlands,

⁴Institute for Computer Science and Business Informatics, University of Vienna, Austria;

⁵European Laboratory for Particle Physics, CERN, Geneva Switzerland

Abstract

In 2006 when the CMS experiment will commence taking data, the CMS Offline computing will consist of a tiered hierarchical set of large computing facilities spread around the world. There will be a large, tier-0, central computing site near the experiment in CERN, five tier-1 regional centres in major research institutions and about 25 tier-2 centres at universities around the world.

Around the year 2003 major elements of this structure will already be in place taking part in the enormous simulation effort that precedes the running experiment. Already CMS has computing and data management needs equivalent to that of one of today's experiments. This paper will describe the CMS requirements and activities for Grid computing, the work, which has already taken place and is in use in the present simulation efforts, and techniques that deal with the problems of heterogeneous data sources.

1 Introduction

In many next generation HEP experiments, object-oriented software engineering tools and languages are used to develop the software infrastructure for the final physics analysis. To store the experiment's data, currently an object-oriented database management system or an object data store is assumed as the data persistency solution. At the highest level of abstraction in the experiment's data models, all data are persistent objects and can be accessed through an object-oriented navigation mechanism.

The experiment's physics detector makes observations of high energy physics collisions. Each observation is called an "event" and has a unique event number. For each event, a number of objects are present. There are raw data objects that hold the data directly taken from the detector, and reconstructed objects that hold processed versions of this raw data.

The high level experiment's data view contains neither the concept of files nor the concept of data replication: all objects are supposed to simply "exist" without regard to how they are stored and how many replicas exist. Files and replication appear only at lower layers of abstraction as implementation mechanisms for the experiment's object view.

A single file will generally contain many objects. This is necessary because the number of objects is so large (in the order of 10^7 to 10^{10} for a modern physics experiment) that storing each object in a single file would lead to scalability problems in the file systems and tertiary storage systems used. Moreover, the object persistency solutions used only work efficiently if there are many objects per file.

As most objects are read-only after creation, access patterns show considerable repetitiveness and locality, and both the user community and the hardware resources are highly distributed, support for replication is clearly desirable. Replication is also desirable because the current production versions of the object persistency layers at each site do not have the native ability to efficiently access objects on remote sites, as they were built under the assumption that a low latency exists when accessing storage.

2 Grid Requirements

CMS physicists need to seamlessly access their experimental data and results, independent of location and storage medium, in order to focus on the exploration for the new physics signals rather than the complexities of worldwide data management. In order to achieve this goal, CMS has adopted a tiered worldwide computing model that will incorporate emerging Grid technology.

CMS has started to use Grid tools for data processing, replication and migration. Important Grid components are expected to be delivered by the Data Grid projects, like EU DataGrid, PPDG and GriPhyN. As part of the activity of interfacing with these projects, CMS has created a set of long-term requirements to the Grid projects. These requirements are presented and briefly discussed.

In the period December 2000 - July 2001, CMS [1] conducted a major requirements and consensus building effort that resulted in a series of documents with concrete requirements for the Grid projects (GriPhyN [2], PPDG [3], and the EU DataGrid [4]) that CMS is involved in as a ‘customer’. At the highest level, the requirement is simply that the Grid projects should deliver software components to CMS that can be used by CMS in the construction of the CMS Data Grid system. At a more detailed level, the requirements give a comprehensive overview of the Data Grid system that CMS intends to operate around December 2003. This CMS Data Grid system contains a large number of software components from different sources, including the Grid projects. Final selection, integration, and operation of these components will remain the responsibility of CMS.

The requirements effort focused in particular on the architectural constraints, which the Grid components to be delivered to CMS need to take into account, on the scalability requirements for these components, and on the level of complexity of the workload that needs to be supported.

To support its current wide-area distributed production effort, CMS has developed, and is already operating, a ‘proto-Grid’ system at a scale of about 10 sites, 500 CPUs, and a few TB of storage space. This proto-Grid system already satisfies some of the requirements outlined below, and uses several components created by the Grid community. The goal is to incorporate additional Grid components when they become available, evolving the system towards greater capabilities and greater scalability.

2.1 Requirements documents

The central requirements document is [5]. This document contains a snapshot, taken in 2001, of the vision that CMS has of the intended software capabilities of its production Data Grid system in 2003, and the expected scaling towards 2007. To capture the expected level of complexity, the vision is sometimes worked out to considerable detail, even though some of these details are likely to be adjusted in future. The document focuses on the relation between the CMS software components and the Grid software components operating inside the 2003 CMS Data Grid system, and contains the architectural constraints for the Grid components.

Tier	CPU capacity	Nr of CPUs	Active tape	Archival tape	Disk
0 (CERN)	455,000 SI95	3000	1540 TB	2632 TB	796 TB
1 (5 all over the world)	105,000 SI95	750	590 TB	433 TB	313 TB
2 (25 all over the world)	26,000 SI95	180	none	50 TB	70 TB

Table 1: Estimates for 2007 CMS hardware capacity needs

With respect to scalability, [6] provides comprehensive estimates of the hardware capacity needs. Grid component scalability requirements for the 2007 timeframe can be directly derived from [6]. Table 1 gives a compact indication. Intermediate scalability requirements for 2003 are driven by the CMS ‘20% data challenge’ milestone, for which the work starts in January 2004 with milestone completion in December

2004. In connection to this challenge, the Grid components delivered to CMS by the end of 2003 need to scale to supporting a hardware configuration that has 20% of the projected 2007 capacity.

With respect to the complexity of the workload to be supported, [5] is an important source, but the most detailed source to date, especially for the ‘chaotic’ user analysis workload, is the HEPGRID2001 model [7], which gives a baseline for the workload that needs to be handled efficiently by the CMS Data Grid system around 2006. The HEPGRID2001 workload model is a model in terms of ‘virtual’ data products, that is in terms of objects that can be created when requested. A workload model at the level of files can be generated by extending the HEPGRID2001 model with a simulation of the CMS mapping of data products to files sets. The creation of such an extension is a planned activity.

2.2 Division of labour

The CMS requirements for the Grid projects [5] define the following high-level division of labour. Tasks in the 2003 CMS Data Grid system for components provided by the Grid community and Grid projects are as follows:

- Basic management and access interfaces for Grid resources such as storage systems, CPUs, and the network.
- Queuing of Grid jobs, Grid job execution management, and integration with local site job submission systems.
- Distributed job scheduling: optimize job execution by efficiently allocating subjobs to sites, moving code to data if possible, by taking into account factors like data location and site loads, and by generating efficient data replication actions to pre-stage data when necessary.
- Error recovery services during job execution, which are configured with CMS-provided error recovery rules and scripts.
- Resource management, monitoring, accounting tools and services.
- Query estimation based on a decomposed job description and the current state of the Grid.
- Efficient wide-area data transfer in terms of files.
- Access by CMS executables to physical Grid files on site-local disk systems via POSIX calls on the regular UNIX file system interface.
- File catalog services mapping logical to physical files.
- File set catalog services.
- File replication services in terms of file sets with the ability to implement CMS-configured consistency management policies.
- Data management services: services to configure and maintain backups or mirrors to ensure the long-term availability and integrity of precious data.
- Resource optimization: longer term data migration (often based on user hints or initiated by system operator commands) to balance the use of resources in different Grid sites.
- Grid wide authentication and authorization services, security infrastructure.

Tasks for CMS components and commercial components selected by CMS are as follows:

- Physics analysis tools that provide user interfaces to the Grid services for end-user physicists, interfaces in terms of the high-level physics application semantics.
- Persistency layer that maintains data product values in files.
- Optimizing the strategy for mapping data product values into files.

- Local and remote extraction and packaging of data products to/from files.
- Configuration management for CMS data products and metadata.
- Generation and maintenance of configuration metadata for each file set, creation of services which allow CMS physicists to find the data they need using application-level queries.
- Efficient job decomposition into subjobs.
- Mapping of the application-level job description to a Grid-level description containing the names of input, work, and output file sets.
- Configuration of resource usage and access policies.
- Creation of error recovery rules and scripts.
- Making the tradeoff between pre-staging and dynamic staging of files, initiation of dynamic file staging operations in CMS executables.

2.3 Grid Components

Concerning the Grid components that are to be created by the Grid projects, and delivered to CMS from now until the end of 2003, [5] defines many requirements, and, even more importantly, many architectural constraints which these components need to take into account. An exact specification of the components to be delivered was however beyond the scope of the requirements activity that CMS conducted. The creation of such exact Grid component specifications is considered to be joint future work between the Grid projects and their customers, with the Grid projects taking the lead in this effort.

3 Grid Data Management Pilot, GDMP

Data replication is an optimization technique well known in the distributed systems and database communities as a means of achieving better access times to data (data locality) and/or fault tolerance (data availability); see [8], [9] and [10]. This technique appears clearly applicable to data distribution problems in large-scale scientific collaborations, due to their globally distributed user communities and distributed data sites. As an example of such an environment, we consider the High Energy Physics community where several thousand physicists want to access the Terabytes and even Petabytes of data that will be produced by large particle detectors around 2006 at CERN, the European Organization for Nuclear Research.

The computing model of a typical next generation experiment at CERN foresees the use of a distributed network of regional centers, each equipped with computing and data storage facilities and linked with wide area network connections [11]. Since these sites are intended to be used in a coordinated fashion, there is a natural mapping to a Grid environment [12] and the High Energy Physics community is building a Data Grid [13] to support the distributed management and analysis of its data. Recently, the European Data Grid Project (“EU DataGrid” [4]) project has been initiated and a prototype project GDMP (Grid Data Management Pilot) [14] has been used in a production environment in an experiment involving the secure replication of database files between several sites in Europe and the United States. GDMP provides file replication services and some preliminary storage management functionality. Although it is not yet a fully functional replication manager (e.g., see [15]), it does provide useful services and is extensible to meet future needs.

GDMP uses services provided by the Globus Toolkit [16] for security and other purposes. An initial version, GDMP version 1.2 [17], was limited to transferring Objectivity [18] database files. In more recent work, we have significantly extended GDMP capabilities by integrating two new Globus Data Grid tools [19], available as an alpha release as of early 2001: the Globus Replica Catalog, which we use to store replica location metadata, and the GridFTP high-performance wide area transport library, which we use as our transport engine.

Data replication is a key issue in a Data Grid and can be managed in different ways and at different levels of granularity: for example, at the file level or object level. In the High Energy Physics community, Data Grids are being developed to support the distributed analysis of experimental data. We have produced a prototype data replication tool, the Grid Data Management Pilot (GDMP) that is in production use within CMS, with middleware provided by the Globus Toolkit used for authentication, data movement, and other purposes. We present here a new, enhanced GDMP prototype implementation that uses Globus Data Grid tools for efficient file replication. We also explain how this architecture can address object replication issues in an object-oriented database management system. File transfer over wide-area networks requires specific performance tuning in order to gain optimal data transfer rates.

3.1 Globus Data Grid Tools

Since GDMP is using new Globus Data Grid tools [19], we describe their features and functionality with respect to file replication.

3.1.1 Replica Catalog

The Globus replica catalog is intended as a fundamental building block in Data Grid systems. It addresses the common need to keep track of multiple physical copies of a single logical file by maintaining a mapping from logical file names to physical locations. The catalog contains three types of object. The highest-level object is the collection, a group of logical file names. Discussions with various user groups show that datasets are normally manipulated as a whole and the collection abstraction provides a convenient mechanism for doing this. A location object contains the information required to map between a logical filename (a globally unique identifier for a file: not a physical location) and the (possibly multiple) physical locations of the associated replicas. The final object is a logical file entry. This optional entry can be used to store attribute-value pair information for individual logical files. We believe that much of this type of data will be stored in a separate metadata catalog [20], but the facility is available.

The operations that can be performed on the catalog are as one might expect: creation and deletion of collection, location, and logical file entries; insertion and removal of logical file names into collections and locations; listing of the contents of collections and locations; and the heart of the system, a function to return all physical locations of a logical file. Further documentation for the replica catalog can be found at: <http://www.globus.org/datagrid/replica-catalog.html>

Replica catalog functions can be used directly in applications, but also form the basis (with GridFTP) for a replica management system that provides functions for the reliable creation, deletion, and management of replicas. Replica management documentation can be found at: <http://www.globus.org/datagrid/replica-management.html>

3.1.2 GridFTP

GridFTP is a data transfer and access protocol that provides secure, efficient data movement in Grid environments. The GridFTP protocol extends the standard FTP protocol, providing a superset of the features offered by the various Grid storage systems currently in use. We choose to work with the FTP protocol because it is the most commonly used protocol for data transfer on the Internet; of the existing candidates from which to start, we believe it comes closest to meeting the Grids needs. The GridFTP protocol includes the following features:

- Public-key-based Grid Security Infrastructure (GSI) [21] or Kerberos support (both accessible via GSS-API [22])
- Third-party control of data transfer
- Parallel data transfer (one host to one host, using multiple TCP streams)
- Striped data transfer (m hosts to n hosts, possibly using multiple TCP streams if also parallel)
- Partial file transfer

- Automatic negotiation of TCP buffer/window sizes
- Support for reliable and restartable data transfer
- Integrated instrumentation, for monitoring ongoing transfer performance

Programmatic access to this functionality is provided via two primary libraries, `globus_ftp_control` and `globus_ftp_client`. These libraries have been used to develop a server, based on the Washington University FTP Daemon (`wuftpd`) that implements the GridFTP features listed above. A full-featured command line tool appropriate for scripting called `globus_url_copy` is provided. A version of the interactive `ncftp` client has also been developed that has GSI support and hence can communicate with GridFTP servers; however, this client does not incorporate the other features listed above. Further documentation for GridFTP and these libraries is available at: <http://www.globus.org/datagrid/gridftp.html>

3.2 General Architectural Issues

In this section, we briefly describe the entire GDMP architecture, focusing on the new features of our second-generation architecture, which concern namespace and file catalog management, efficient file transfer, and preliminary mass storage management. See [14] for a description of GDMP version 1.2, which is in production use in CMS experiment.

GDMP is a file replication software system that was initially designed to replicate Objectivity database files from one site (storage location) to one or more other remote sites. A storage location is considered to be a disk space on a single machine or on several machines connected via a local-area network and a network file system. Remote sites are connected to each other via long latency (as compared to local-area network) wide-area network connections. GDMP works as follows. A site produces a set of files locally and another site wants to obtain replicas of these files. In the case of Objectivity files, each site is running the Objectivity database management system locally that has a catalog of database files internally. However, the local Objectivity database management system does not know about other sites and a replication mechanism is required that can notify other sites about new files, efficiently transfer the files to the remote site, and integrate the filenames into the Objectivity internal file catalog. An additional server needs to be available at each site to handle replication requests and to trigger file transfers, notification messages, and updates of local catalog information. Simply put, this is done by a GDMP server running at each site where files are produced and possibly replicated.

With the new architecture and newly added components, GDMP has been extended to handle file replication independent of the file format. Note that we do not address replica synchronization issues; hence this work is useful mainly for read-only files. In GDMP 1.2, the file replication process was too tightly connected to Objectivity-specific features for naming conventions of logical and physical files and for obtaining information about the files from the Objectivity's catalog. This dependency is removed in the new version (the official release will be called GDMP 2.0) by splitting the data replication process into several steps. Other possible file types are Oracle files and flat files with particular internal structure. Thus, successfully replicating a file from one storage location to another one consists of the following steps:

- pre-processing: This step is specific to the file formats and might even be skipped in certain cases. This step prepares the destination site for replication, for example by creating an Objectivity federation at the destination site or introducing new schema in a database management system so that the files that are to be replicated can be integrated easily into the existing Objectivity federation.
- actual file transfer: This has to be done in a secure and efficient fashion; fast file transfer mechanisms are required.
- post-processing. The post-processing step is again file type specific and might not be needed for all file types. In the case of Objectivity, one post-processing step is to attach a database file to a local federation and thus insert it to an internal file catalog

- insert the file entry into a replica catalog: This step also includes the assignment of logical and physical filenames to a file (replica). This step makes the file (replica) visible to the Grid.

The GDMP replication process is based on the producer-consumer model: each data production site publishes a set of newly created files to a set of one or more consumer sites, and GDMP ensures that the necessary data transfer operations (including all the steps mentioned above) complete successfully. These services are implemented by a set of interacting servers, one per site participating in the data replication process.

Let us consider a Data Grid where data is produced and replicated (consumed) at a number of sites. Each of these sites deploys a GDMP server to interact with other sites and provides GDMP client commands for publishing file information to other sites (notifying other sites that new data is available) and initiating file replication requests for a set of files. In more detail, a high-level file get request is issued by a GDMP client application at one site to get files from another site and create replicas locally.

To sum up, GDMP client APIs provide four main services to the end user [14]:

- subscribing to a remote site for getting informed when new files are created and made public,
- publishing new files and thus making them available and accessible to the Grid,
- obtaining a remote sites file catalog for failure recovery, and
- transferring files from a remote location to the local site.

Every client request to a GDMP server is authenticated and authorized by a security service. GDMP uses the Globus Security Infrastructure (GSI) [21], which provides single sign capabilities for Grid resources.

Client requests are sent to the GDMP server through the Request Manager. The Request Manager is the client-server communication module, which is used to generate client requests and implement server functions for serving these requests. Using the Globus IO and Globus Data Conversion libraries, the Request Manager provides a limited Remote Procedure Call functionality.

File transfer requests are served by the GDMP Data Mover service that uses a local file transfer server such as FTP. Since file transfers must be both secure and fast, the Data Mover service has to use a file transfer mechanism that provides both features (more in Section 3.4). Once files are successfully transferred, they have to be inserted into a replica catalog. The Replica Catalog Service provides this functionality (see Section 3.3).

In an early version, GDMP was restricted to disk-to-disk file replication and it was assumed that all files are permanently available on disk. Since Data Grids deal with large amounts of data, files are permanently stored in Mass Storage Systems (MSS) such as HPSS and moved between disk and tape on demand. Thus, a disk pool is considered as a cache. GDMP provides a plug-in for initiating file stage requests on demand between a disk pool and a Mass Storage System (see Section 3.5).

In the next subsections, we describe the replica catalog, data mover, and the storage management service in detail.

3.3 Replica Catalog Service

The GDMP replication service uses a Replica Catalog to maintain a global file name space of replicas (see Section 3.1.1). GDMP provides a high level replica catalog interface and currently uses the Globus Replica Catalog as the underlying implementation. An end user who produces new files uses GDMP to publish information into the replica catalog. This information includes the logical file names, meta-information about the file (such as file size and modify time-stamps) and the physical location of the file. In detail, when a site publishes its files:

- These files (and the corresponding meta-information) are added to the replica catalog.
- The subscribers are notified of the existence of new files.

The Replica Catalog service also ensures a global name space by making sure that all logical file names are unique in the catalog. GDMP supports both the automatic generation and user selection of new logical file names. User selected logical file names are verified to be unique before adding them to the replica catalog. Race conditions on the replica catalog are currently not dealt with.

Client sites interested in a new file can query the Replica Catalog Service to obtain the information required to replicate the file. Users can specify filters to obtain the exact information that they require; information is returned only about those logical files that satisfy the filter criteria. The information returned contains the meta-information about the logical file and all the physical instances of the logical file. This information can then be used as a basis for replica selection based on cost functions, which is part of planned future work. (See [23] for some early ideas.)

The current Globus Replica Catalog implementation uses the LDAP protocol to interface with the database backend. We do not currently distribute or replicate the replica catalog but instead, for simplicity, use a central replica catalog and a single LDAP server for the Replica Catalog service. In the future, we will explore both distribution and replication of the replica catalog.

The GDMP Replica Catalog service is a higher-level object-oriented wrapper to the underlying Globus Replica Catalog library. This wrapper hides some Globus API details and also introduces additional functionality such as search filters, sanity checks on input parameters, and automatic creation of required entries if they do not already exist. The high level API is also easier to use and requires fewer method calls to add, delete, or search files in the catalog.

We have already tested the new API successfully on two independent test beds involved LDAP servers at CERN (Switzerland), Caltech (California, USA) and SLAC (California). Note that each test bed only used a single replica catalog.

3.4 Data Mover Service

In a Data Grid where large amounts of data have to be transferred from one site to another (“point-to-point replication”) we require high-performance data transfer tools. This is one of the major performance issues for an “efficient” Data Grid and is the target of the Globus Data Grid Toolkits GridFTP system

The GDMP Data Mover service, like the GDMP Replica Catalog service, has a layered, modular architecture so that its high level functions are implemented via calls to lower level services that perform the actual data manipulation operations. In this case, the lower level services in question are the data transfer services available at each site for movement of data to other Grid sites.

It seemed to us that the GridFTP design addressed the principle requirements for a Data Grid data transfer primitive, in particular security, performance, and robustness. Hence, we have explored the use of GridFTP as GDMP’s underlying file transfer mechanism.

The large size of many data transfers makes it essential that the Data Mover service be able to handle network failures and perform additional checks for corruption, beyond those supported by TCP’s 16 bit checksums. Hence, we use the built-in error correction in GridFTP plus an additional CRC error check to guarantee correct and uncorrupted file transfer, and use GridFTP’s error detection and restart capabilities to restart interrupted and corrupted file transfers. In the future, we will exploit GridFTP’s support for “pluggable” error handling modules to incorporate a variety of specialized error recovery strategies.

3.5 Storage Management Service

In order to interface to Mass Storage Systems (MSS), the GDMP service uses external tools for staging files. For each type of Mass Storage System, tools for staging files to and from a local disk pool have to be provided. We assume that each site has a disk pool that can be regarded as a data transfer cache for the Grid and that, in addition, a Mass Storage System is available at the same site but does not manage the local disk pool directly. The staging to local cache is necessary because the MSS is mostly shared with other administrative domains, which makes it difficult to manage the MSS's internal cache with any efficiency. Thus, GDMP needs to trigger file staging requests explicitly. This is our current environment, which might change slightly in the future.

A file staging facility is necessary if disk space is limited and many users request files concurrently. If a remote site requests a replica from another remote site where the file is not available in the disk pool, GDMP initializes the staging process from tape to disk. The GDMP server then informs the remote site when the file is present locally on disk and at that time performs automatically the disk-to-disk file transfer.

In the replica catalog, physical file locations are stored and contain file locations on disk. Thus, by default a file is first looked for on its disk location and if it is not there, it is assumed to be available in the Mass Storage System. Consequently, a file state request is issued and the MMS transfers the file to the disk location stored in the replica catalog. Note that Objectivity has an interface to HPSS [24] and the file naming convention is the same: the default location is a disk location. Some other storage management systems have a tape location as a default file location.

Note that more sophisticated space management mechanisms such as reservation of disk space are currently not available. In particular, the underlying storage system needs to provide an API for storage allocation, e.g. allocate storage (data size). In this case, the file replication transfer might be started only if the requested storage space can be allocated.

GDMP has a plug-in for the Hierarchical Storage Manager (HRM) [25] APIs, which provide a common interface to be used to access different Mass Storage Systems. The implementation of HRM is based on CORBA communication mechanisms. Some initial integration tests have been performed, with promising results. Integration with HRM will provide GDMP with a flexible approach to deal with the different MSSs being used at the different regional centers where GDMP has been installed. It also provides a cleaner interface as compared to the staging script solution, which we had employed previously.

4 Data Integration

This section discusses several strategies to perform data integration, and mentions several research projects in this area. This section serves as an introduction to data integration. It shows that the problem is not unique for CERN, and that literature describes solutions for similar problems. Furthermore, it introduces XML as a format that can be used within data integration approaches.

Three levels of data can be identified within a data integration system [26]:

- Data. Numbers and Strings stored in either databases, flat files or other sources
- Information. This describes the data (Meta data or database schemes), and relates to information modeling (e.g. on top of several data sources). Furthermore it deals with data retrieval and fusion through a certain model.
- Knowledge. Knowledge provides an added value to information by using the data attached to the information in for example, simulations, visualization or problem solving methods. Knowledge can be either consumed by humans (analyzing a graph), or stored as data with information attached to it. This data can then be used in other knowledge creation.

Notice that every level deals with data (information and knowledge are data too). We will focus on the second level. The first level is well covered by literature (see for example [27]). The knowledge level is in many cases very domain specific (specific algorithms, simulations, etc), or relates to very general problem solving environments (e.g. expert systems).

4.1 Introduction

Scientific data sets are not only getting larger and larger but the complexity is also increasing, meaning that the extraction of meaningful knowledge requires more and more computing resources. Furthermore, scientific collaborations are getting larger and more geographically dispersed. This leads to the need of catalogues and indexes of data archives and the ability to select and integrate data objects, and to define complex processing. [28] identified several areas that are important: information modeling, standard scientific data objects/models and database interoperability. Large-scale data manipulation requires the tight integration of data and computing sources. Some domains that deal with large amounts data stored in different repositories are:

- Geographic Information Systems. Different repositories for geographic information systems contain data of different aspects of a landscape [29], [30].
- Biology. Data about RNA and DNA strings is stored in different databases [31].
- Libraries [32]
- Astrophysics [33]
- Gravitational wave observatory [28].

The previous list shows that data integration is an issue in different scientific areas. What do these areas have in common? They deal with large amounts of data. Data are not located in one database but are distributed over many data sources. To analyze these data, often data from different sources and different formats need to be correlated. Applications need to use the same data in order to have consistent and reliable outputs. These data can be located in different data sources. To achieve this, methods, models, and middleware are developed for seamless integration of the data. If we look at these characteristics we see that they also apply to the high energy physics environment [28].

Besides science, data integration also takes place in industry. Companies want to integrate data from the complete production process. An example of data integration in industry is an ERP (Enterprise Resource Planning) system. The costs for such systems can be considerable. Furthermore, it will change the organization with respect to how people deal with data and information. [34] gives an overview of this. There are several reasons not to choose an ERP system for CERN:

- Complexity of collaboration. Geographically dispersed. Numerous small groups in research institutes
- Terms of commitment of groups within CERN. The autonomous way of working within the different domains at CERN.
- Heterogeneous environment. Many groups choose their own technology and data models for various reasons. Because of the autonomous way of working this will not be easy to change.

4.2 Data integration strategies

There are several strategies for data integration. We make a categorization into three groups: data warehouse, distributed database, and mediators. The next sections discuss these three approaches. Data integration strategies can also be classified by comparing the degree of autonomy, distribution, and heterogeneity they exhibit. However this classification leads to a multitude of categories and does not give a clear field of data integration strategies

4.2.1 Data warehouse

If data do not change frequently, a data warehouse approach can be followed. A data warehouse is a collection of materialized views derived from relations that may not reside at the warehouse. Using these stored views, user queries, can often be evaluated much more cheaply than using the base relations. Keeping the views consistent with updates to the base relations, however, can be expensive. A data warehouse builds cross-references of information between these different data sources to enable data analysis. It groups data into subject areas so that users can find data specific for their domain faster and more efficiently. Furthermore, it can maintain historical data for trend analysis.

A data mart has similar functions as a data warehouse except that a data mart is a lot smaller in size and has a smaller group of users. A data mart is a materialized view on a data warehouse. For example, a department can design a data mart that is tailored to the specific department needs. The data mart can contain additional domain specific information for the department. Furthermore, a data mart gives more performance to the local domains, because the data are local. Depending on the business requirements and the types of data, the data loading frequency can be just once, once a day, once a week, or once a month. There will be on-going data and system administrative work required to maintain the data warehouse.

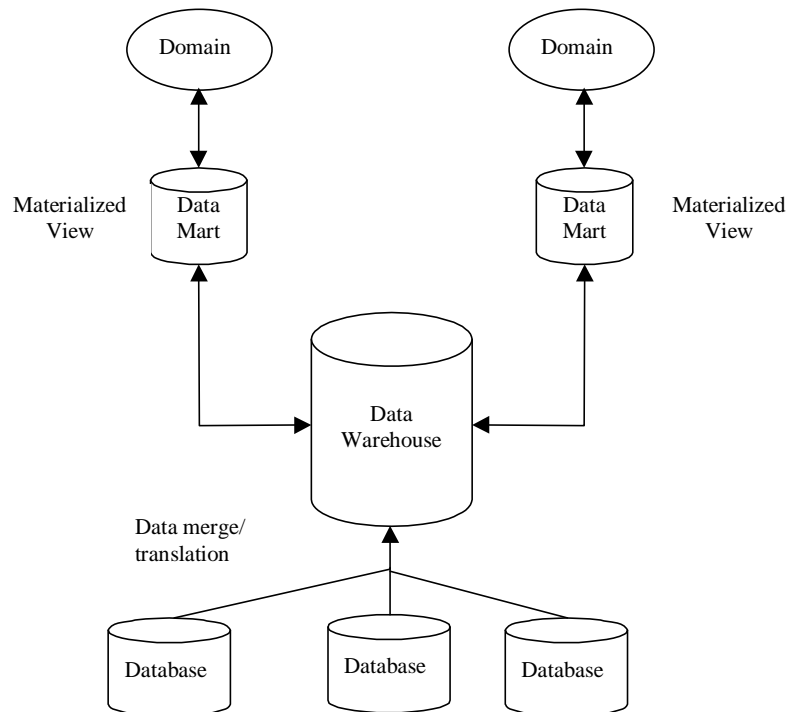


Figure 1 Data warehouse and data marts

Figure 1 shows a diagram of a data warehouse and data marts. The main drawback of the centralized data warehouse is that it is difficult to develop a global data model for most organizations. It is also difficult to agree on a corporate wide level of detail and naming conventions. Data marts can provide users with a more domain specific “schema” for data. However, if users have unpredictable needs, a data warehouse might not be the appropriate approach.

4.2.2 Distributed databases

A distributed database is a collection of logically interrelated databases distributed over a computer network [35]. Both the storage of data and the processing of queries and transactions may occur across several nodes (also referred to as sites). Two types can be identified within distributed databases:

- Every site runs the same type of DBMS (DataBase Management System) (homogeneous).
- Heterogeneous database systems or multi databases provide integrated access to multiple data sources.

Users are shielded from the different schemas of the underlying data. A heterogeneous distributed database is one where the local databases need not be managed by the same DBMS. For example, one DBMS can be a relational system while another can be a hierarchical system or OODBMS (Object Oriented Database Management System). In addition to the main functions, the distributed DBMS must provide interfaces between the different DBMS.

An advantage of a distributed database is that it enables direct access to the sources. Within a data warehouse data could be outdated if it is not updated frequently. The problem with a distributed database is to keep it consistent. In order to keep consistency distributed transactions can use a two-phase commit protocol in order to synchronize related pieces of work taking place in different processes or on different data sources. Furthermore, the concept of “ACID” is used. ACID stands for: (1) Atomic - the transaction will either complete successfully in all the nodes participating in the distributed database or in none. (2) Consistent - the transaction will always produce the same results if applied more than once (i.e. must be consistently reproducible). (3) Isolated - the view of the data across the entire distributed database at the moment of the transaction must be protected from changes until the transaction has completed across all the nodes. (4) Durable - once committed, the data updated or appended through the transaction become secure.

Not every application needs data to be consistent all the time. For example, it can be sufficient to update data every night or every week. This creates weaker constraints on consistency in a distributed database. Furthermore, different databases within a distributed database can be stored in different places. This influences the performance for data access. As a result, applications can have a local copy of the data. Weak consistency and local data access are characteristics of a data warehouse with data marts.

4.2.3 Mediator

The problem of heterogeneous data sources can be tackled by the use of mediators. A mediator is an intermediate between the data sources and the applications/users or between different data sources. A mediator has knowledge about where data are located, and decouples queries from applications/users, into sub queries suited for different data sources. [36] uses the following definition: "mediators are software agents which act as translators for data, encapsulating all the routine work of converting data from one format to another". [37] gives a classification of mediated query systems. Mediators in general have (amongst others) the following characteristics:

- Mediators are small and focused on a particular type of data within data sources. This keeps the structure of mediators simple and makes it less sensitive to changes of the data source.
- Mediators can be based on other mediators.

By defining multiple mediators and building a hierarchy, the scope of mediators can be extended in terms of other mediators. A problem within mediator technology is coping with change of data sources. It cannot be predicted how a source will change. Therefore, there will always be a human component in building or changing mediators. Efforts have been made to semi-automate this work by constructing general frameworks for mediators (see [38] and [39]). Mediators exploit the structure of the data sources. It is very difficult to integrate data while not assuming anything about its structure. Exploiting the structure of sources makes it easier to integrate sources. A drawback of this coupling is the changing of structures of data sources. This is partly solved by designing mediators that are focused on specific structures within a data source.

Within many information integration projects, mediator technology has been used ([40], [38], [41], [42], [39]). In order to translate and integrate the data into a uniform view it is convenient that the data describes its own structure. The TSIMMIS project [40] used an OEM (Object Exchange Model). This language is a predecessor of XML. MIX [42] used XML for this. Information integration becomes more important

nowadays when more and more data sources become available via the Web. These sources have different structures (relational DB, OODB, or flat file).

Mediator technology becomes important if we want to query these sources without having knowledge of all the different structures of these sources, and when the content of these sources can change frequently. Even when all these sources have an XML representation, names and attributes can be different. As an example one can imagine it will be possible to query multiple databases of films. Films can be stored on video or DVD. Within one database titles are sorted without words like “the”, “on” or “it”. Names of actors can be stored as one name, or as first and last name, depending on the database.

5 Conclusions

We have looked at CMS computing from the point of view of CMS’s Grid requirements, CMS’s use of data replication already used in simulation, and techniques of accessing heterogeneous and distributed data sources.

We believe that our input into the European Union project DataGrid reflects the present needs of a large database user. The timeliness of the input was especially important since the architecture of the DataGrid is taking shape this year (2001). The present demands rely on the presence of a file system although the user sees only an individual object or set of objects. The mapping of the objects to the file system is one of the many functions that must be done by the CMS application programmers. We depend heavily on the Grid tools being provided by Globus, industry and the DataGrid project.

CMS has taken an active role within the DataGrid project implementing GDMP for data replication. This was initially for Objectivity database files but has been extended to more general files using a replica catalog. We are actively using this software for our production jobs that simulate the CMS detector. Simulated data is generated in many sites and replicated to other sites when needed. The use of Globus tools to transfer the data has resulted in a highly reliable system that fully exploits the network.

Finally, we gave an overview of access to distributed heterogeneous data sources using data warehousing, distributed databases or mediators. The autonomous way that CMS subdetector groups are working results in a variety of data sources each with its own model and technology. When the subdetectors are assembled we will need to also assemble the data describing each subdetector.

6 Acknowledgements

The number of people who should be acknowledged is large. The Grid requirements were the result of many discussions involving almost everyone from the CMS Computing and Core Software project, the GDMP software was created in collaboration with the DataGrid project, especially workpackage 2, and the Globus team, and the data integration involved people from CMS subdetectors especially ECAL where people from CERN and the University of West England were actively involved.

7 References

- [1] CMS: <http://cmsdoc.cern.ch/>
- [2] GriPhyN: <http://www.griphyn.org/>
- [3] PPDG: <http://www.ppdg.net/>
- [4] European Data Grid project: <http://www.eu-datagrid.org/>
- [5] Koen Holtman, on behalf of the CMS collaboration. CMS Data Grid System Overview and Requirements. CMS Note 2001/037. <http://kholtman.home.cern.ch/kholtman/cmsreqs.ps> , .pdf
- [6] Ian Willers. CMS Interim Memorandum of Understanding: The Costs and How They are Calculated. CMS Note 2001/035

- [7] Koen Holtman. HEPGRID2001: A Model of a Virtual Data Grid Application. Proc. of HPCN Europe 2001, Amsterdam, p.711-720, Springer LNCS 2110. CMS Conference Report 2001/006. Web site: <http://kholtman.home.cern.ch/kholtman/hepgrid2001/>
- [8] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. Proceedings of IEEE Infocom, 1999
- [9] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, Y. Yerushalmi. Web Caching with Consistent Hashing. 8th International World Wide Web Conference, 1999
- [10] R. Tewari, M. Dahlin, H. Vin, J. Kay. Design Considerations for Distributed Caching on the Internet, 19th IEEE International Conference on Distributed Computing Systems, 1999
- [11] H. Newman. Worldwide Distributed Analysis for the Next Generations of HENP Experiments. Computing in High Energy Physics, February 2000
- [12] I. Foster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufmann, 1999
- [13] 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 Data Sets, J. Network and Computer Applications, 2000
- [14] A. Samar, H. Stockinger. Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication, IASTED International Conference on Applied Informatics (AI2001), Innsbruck, Austria, February 2001
- [15] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, K. Stockinger, Data Management in an International Data Grid Project, 1st IEEE/ACM International Workshop on Grid Computing (Grid'2000), Bangalore, India, 17-20 Dec. 2000
- [16] I. Foster, C. Kesselman, The Globus Toolkit, The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufmann, 259-278, 1999
- [17] GDMP web page: <http://cmsdoc.cern.ch/cms/grid>, February 2001
- [18] Objectivity, Inc. <http://www.objectivity.com>, February 2001
- [19] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. 18th IEEE Symposium on Mass Storage Systems and 9th NASA Goddard Conference on Mass Storage Systems and Technologies, San Diego, April 2001
- [20] C. Baru, R. Moore, A. Rajasekar, M. Wan. The SDSC Storage Resource Broker. CASCON'98 Conference, 1998
- [21] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids, ACM Conference on Computers and Security, 83-91, 1998
- [22] J. Linn. Generic Security Service Application Program Interface Version 2, Update 1, IETF RFC 2743, 2000. <http://www.ietf.org/rfc/rfc2743>
- [23] S. Vazhkudai, S. Tuecke, I. Foster. Replica Selection in the Globus Data Grid, IEEE International Symposium on Cluster Computing and the Grid (CCGrid2001), Brisbane, Australia, May 2001
- [24] A. Hanushevsky. Obejectivity/DB Advanced Multi-threaded Server (AMS) www.slac.stanford.edu/~abh/objy.html, April 2000
- [25] L. M. Bernardo, A. Shoshani, A. Sim, H. Nordberg, Access Coordination of Tertiary Storage for High Energy Physics Application, 17th IEEE Symposium on Mass Storage Systems and 8th NASA Goddard Conference on Mass Storage Systems and Technologies, Maryland, USA, March 27-30, 2000

- [26] Reagan W. Moore "Knowledge-based Grids" 18th IEEE symposium on Mass Storage Systems and Technologies, 2001
- [27] W. Kim "Modern Database Systems", 1995, Addison-Wesley. ISBN 0-201-59098-0
- [28] Roy Williams, Paul Messina, Fabrizio Gagliardi, John Darlington, Giovanni Aloisio, "Large Scientific Databases" workshop Annapolis, Maryland, USA, 1999 September 8-10
- [29] P.C.H. Wariyapola, S.L.Abrams, A.R.Robinson, K.Streitlien, N.M.Patrikalakis, P.Eliseeff, H.Schmidt, "Ontology and Meta-data Creation for the Poseidon Distributed Coastal Zone Management System", 1998
- [30] Stefan Göbel Karen Lutze, "Development of meta- databases for geospatial data in the WWW", ACM GIS vol 11, 1998, pp94-99
- [31] T. Critchlow, M. Ganesh, R. Musick, "Meta-Data Based Mediator Generation", LLNL, Proceedings of the Third International Conference of Cooperative Information Systems, 1998
- [32] Bill Birmingham et al. "EU-NSF Digital Library Working Group on Interoperability between Digital Libraries Position Paper", road map report from the DELOS project, 1998
- [33] Cynthia Cheung, Dave Leisawitz, Nick Roussopoulos, Stephen Kelly, Jane Wang, Gail Reichert, David Silberberg, "AMASE: An Object-Oriented Meta-database Catalog for Accessing Multi-Mission Astrophysics Data", SISIC, November 6-9, 1995
- [34] Christopher Koch, Derek Slater and E. Baatz, "The ABCs of ERP", Enterprise Resource Planning Research Center, 1999, <http://www.cio.com/forums/erp/>
- [35] M. Tamer Oszu, Patrick alduriez. "Distributed database systems: Where are we now?" IEEE Computer, pages 68-78, August 1991
- [36] Gio Wiederhold, "Mediators in the Architecture of Future Information Systems", IEEE computer magazine, 1992
- [37] Domenig, R. and Dittrich, K.R. (1999)." An Overview and Classification of Mediated Query Systems". SIGMOD Record 28(3)
- [38] V.S. Subrahmanian, Sibel Adali, Anne Brink, Ross Emery, James J. Lu, Adil Rajput, Timothy J. Rogers, Robert Ross, Charles Ward" HERMES: Heterogeneous Reasoning and Mediator System " Fourth Annual Electrical Engineering, Computing and Systems Research Review Day Friday, April 30, 1999
- [39] Matthew Morgenstern, "An Integration Platform for Heterogenous Databases: The DAISy System" a DARPA funded project, 1996
- [40] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and Jennifer Widom. "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS". In Proceedings of the AAAI Symposium on Information Gathering, pp. 61-64, Stanford, California, March 1995
- [41] R. J. Bayardo Jr., W. Bohrer, and et. Al., " InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments", ACM SIGMOD, 1997
- [42] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, V. Chu, "XML-Based Information Mediation with MIX" exhibition program, ACM Conf.on Management of Data SIGMOD99, 1999