The CMS Dataset Bookkeeping Service

Anzar Afaq[1],Andrew Dolgert[2], Yuyi Guo[1],Chris Jones[2], Sergey Kosyakov[1],Valentin Kuznetsov[2], Lee Lueking[1],Dan Riley[2], Vijay Sekhri[1]

[1]Fermilab, Batavia, Illinois 60510
[2]Cornell University, Ithaca, New York 14850

E-mail: lueking@fnal.gov

Abstract. The CMS Dataset Bookkeeping Service (DBS) has been developed to catalog all CMS event data from Monte Carlo and Detector sources. It provides the ability to identify MC or trigger source, track data provenance, construct datasets for analysis, and discover interesting data. CMS requires processing and analysis activities at various service levels and the DBS system provides support for localized processing or private analysis, as well as global access for CMS users at large. Catalog entries can be moved among the various service levels with a simple set of migration tools, thus forming a loose federation of databases. DBS is available to CMS users via a Python API, Command Line, and a Discovery web page interfaces. The system is built as a multi-tier web application with Java servlets running under Tomcat, with connections via JDBC to Oracle or MySQL database backends. Clients connect to the service through HTTP or HTTPS with authentication provided by GRID certificates and authorization through VOMS. DBS is an integral part of the overall CMS Data Management and Workflow Management systems.

1. Introduction

The CMS Dataset Bookkeeping Service (DBS) [1] provides access to a catalog of all event data used by the Experiment and records the files and their processing parentage. It includes the data definitions such as Run number, Luminosity Section number, the algorithms and configurations used to process the data, and details of the information within each file. With the information in the catalog the provenance of any data can be tracked back to the original RAW data or Monte Carlo generation. Data files are mapped to File Blocks that aggregate related files for data placement purposes, and the locations of these blocks are also tracked within DBS. Users and production teams can discover what data exists in the catalog using a Web browser, Command Line Interface, or through a DBS API. The service is part of the overall CMS Data Management and Workflow Management System [2] and is used for distributed analysis using tools such as CRAB [3], production data processing activities using ProdAgent [4], and correlated with the data locations in PhEDEx [5]. For additional overview of the CMS system refer to [6]. DBS does not contain any event data itself, but only pointers to physical data that is managed by other parts of the over all CMS Data Management system.

2. Typical Use Cases

Use cases for DBS include MC generation, detector data, large scale production processing and user data analysis. The level at which the activity is undertaken is referred to as the "scope"

(see [7]). Fundamentally there are two scopes: Global and Local. The Global scope is a single instance shared by all CMS and is the authoritative source for official data information. Many local scopes are established to do processing and analysis at distributed sites for which the details will eventually be migrated to the global scope. In a typical production scenario, the steps are as follows:

(i) Identify the desired input data via queries to the global DBS
(ii) Find the physical locations of the input data
(iii) Copy the DBS information for the input data to the appropriate local scope DBS
(iv) Configure and submit the EDM framework production jobs
(v) As jobs finish, collect the framework job reports and register the intermediate output files into the local scope DBS
(vi) Merge the intermediate output files into final output files registered in the local scope DBS
(vii) Publish the final output files to the global scope DBS
(viii) Copy the final output files to the appropriate Tier1 and register locations with the global DBS

A typical flow of data through the processing steps for a recent computing and software challenge are shown in Fig. 1. In this picture Tier 0,1,2 refer to processing tiers of the computing model, and RAW, FEVT, AOD refer to data tiers in the data model. The important steps included in this picture are 1) Adding new data, 2) Processing existing data, 3) Merging data for example by combining, and 4) Skimming data, also referred to as filtering or streaming. In all cases, the data provenance and complete file and dataset parentage are recorded.

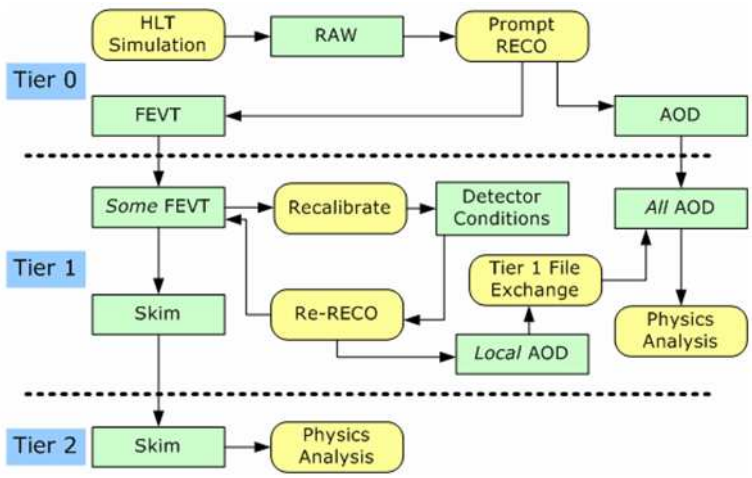

Figure 1. Typical use-cases considered for DBS design.

3. CMS Data Management Concepts

The concepts of the CMS Data Management and Workflow Management models are reflected in the design of the DBS schema. Some of the more important relationships of the schema are illustrated in Fig. 2 and include the following concepts:

Dataset: A set of files representing a coherent sample. Datasets are defined primarily by processing history and event selection criteria.

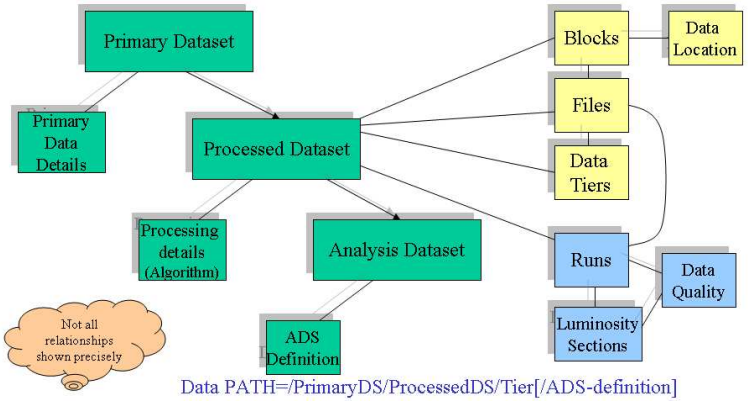

Figure 2. Relationships of the DBS schema.

Primary Dataset: Determined by High Level Trigger (HLT) trigger classification or MC production parameters

Processed Dataset: Slice of a primary dataset with a consistent processing history. Note: May include multiple copies of some events with slight differences in processing.

Analysis Dataset: Snapshot of a subset of processed dataset representing a coherent sample for physics analysis

Luminosity Section: Sub-section of a run during which time the instantaneous Luminosity is unchanging. (2^{20} orbits $\sim$ 93 Seconds). Unit of accounting for integrated luminosity. Production data files will contain one or many whole luminosity sections.

Run: Period of data-taking over which conditions are stable.

Data Quality: DQ flags set for Run or specific Lumi Sections of a Run

Data Tier: A set of objects grouped together for each event defined by the software release configuration files.

Files: Parentage relationships between files is recorded in addition to Dataset lineage, Files can be marked as "unavailable" if they are lost or corrupted.

File Blocks: Files grouped into blocks of reasonable size or logical content. Tracking many files grouped into blocks has advantages in data transfers. Physical storage locations is recorded at the block level.

Block Location: Location of File Blocks at Site Storage Elements (SE).

4. System Architecture and Design

The DBS system is a multi-tier web application illustrated in Fig. 3. The servlet design is modular and makes it easily adaptable to multiple database technologies and various payload formats. Currently there are three types of database supported, ORACLE, MySQL, and SQLite. This enables DBS to be deployed in a range of environments from general CMS at large installations to specific personal installations.An XML format is used for the format of the HTTP payload exchanged with the client.

DBS has been designed to support a hierarchical deployment model as shown in Fig. 4. This model makes it convenient for workgroups and personal users to work within the same environment as that used for CMS at large, while allowing the system to scale and provide testing and scratch instances of the database. One global service provides the overall CMS community accesses to the official information about CMS datasets, and various workgroup

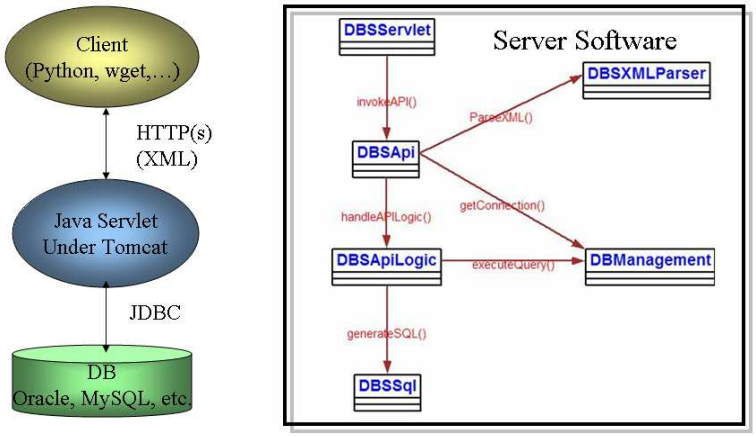

Figure 3. DBS system architecture and server software design.

and personal level DBS instances are deployed to provide datasets respectively at sites or individually controlled nodes. A migration API is provided enabling the transfer of datasets from one instance to another. In addition to making the system extremely flexible this design allows the system to scale to the needs of CMS with many dozens of sites, hundreds of users and a multitude of data processing and analysis activities. The migration among DBS instances is

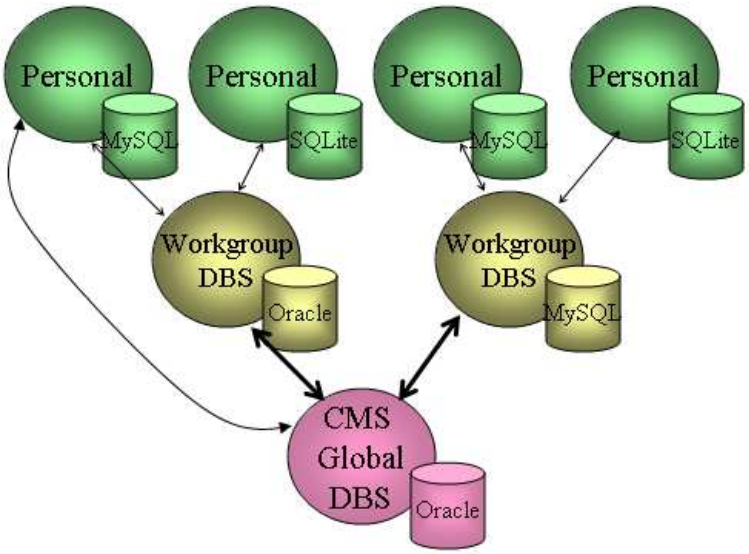

Figure 4. DBS Instance hierarchy.

enabled through a straightforward API that has several options to enable transfers of the desired granularity and parentage depth. It is possible to transfer an entire Processed Dataset with its complete parentage, or to specify a single file block. It is also possible to transfer the dataset with no parentage at all from the global instance to any local instance which can significantly reduce the amount of information transferred and stored in the local instance. Several rules are automatically enforced to ensure consistency of the data across all instances, and especially between all local instances and the global instance.

Authentication and authorization are provided using X.509 standard Grid Certificates with VOMS and GUMS. Fig. 5 shows how the grid authentication is achieved in the system. A Part of the Globus GSI is used on the server for the authentication. Authorization is required for inserting new data, modifying certain status fields, and modifying existing data. Read only access does not require authentication.

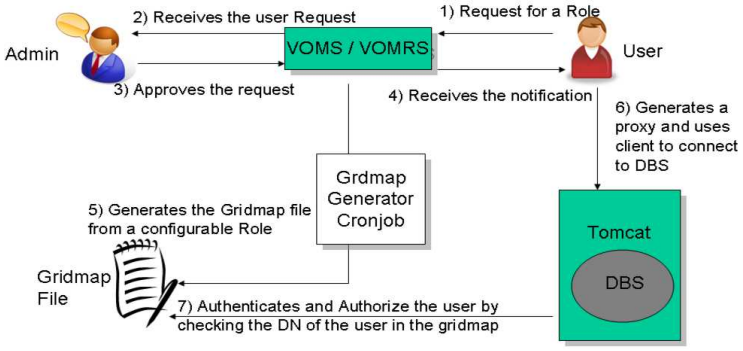

Figure 5. The authentication and authorization model used by DBS.

One important part of the system is the web-based discovery interface that gives users the ability to easily find the data they are looking for. The design and implementation details are described in [8] and example screen shots are shown in Fig. 6. The interface enables the user to locate datasets, files, file block locations based on drop-down menus or wildcard searches. It also provides instantaneous access to additional information related to the data once the user has accomplished his search goal.

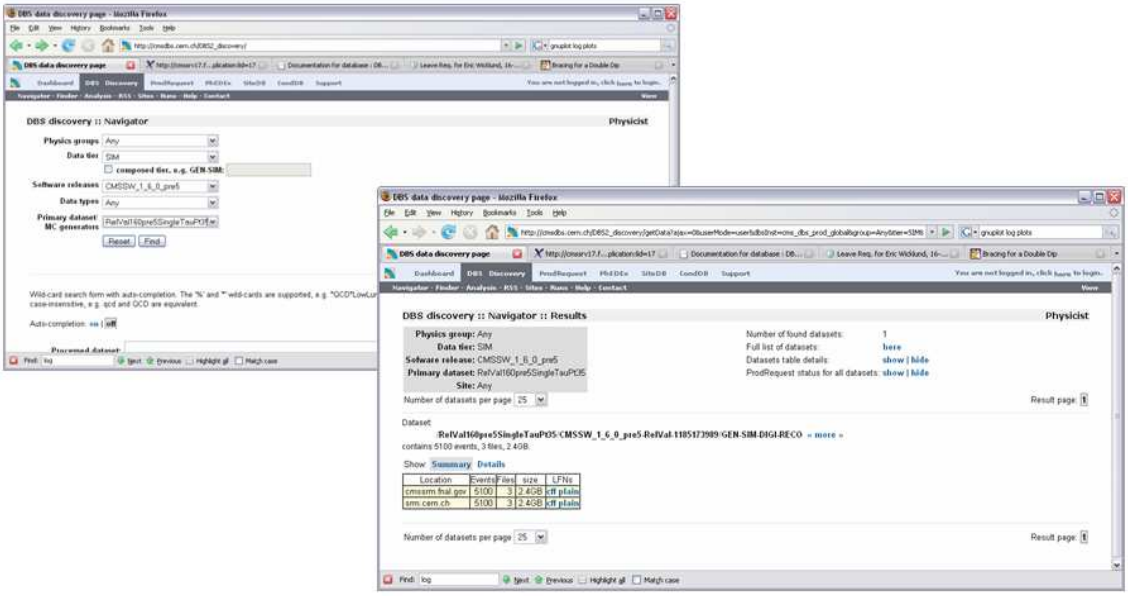

Figure 6. The DBS Discovery page Web interface enables users to locate the data in which they are interested.

5. Operational Experience

The current DBS deployment has been in operation since April of 2007 and used to record all MC production and analysis steps. Prior to this, there was a DBS prototype in operation from the summer of 2006 to April 2007 and nearly all of its data were migrated to the current production system. CMS operates a DBS service at CERN comprising 1) a Global DBS instance that is the authoritative source of official CMS dataset information, 2) Six "local" instances used for MC production, 3) one Tier 0 instance and, 4) several test instances. Overall, the current implementation and deployment of DBS has proved to be performant, highly reliable and stable. A general overview of this is shown in the diagram in Fig. 7.

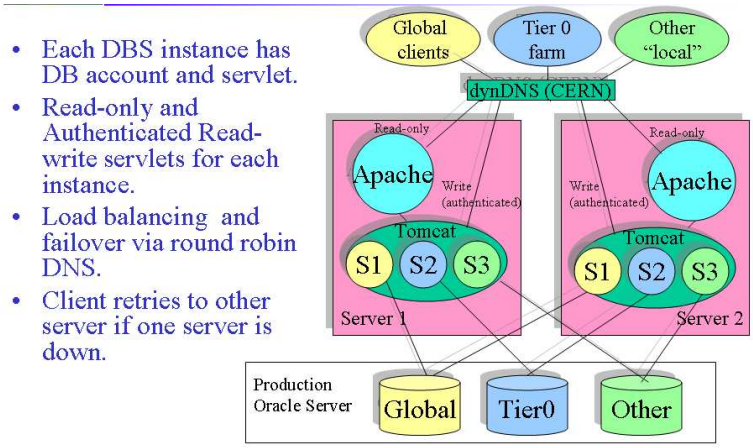

Figure 7. Hardware and server deployment at for the central DBS service at CERN.

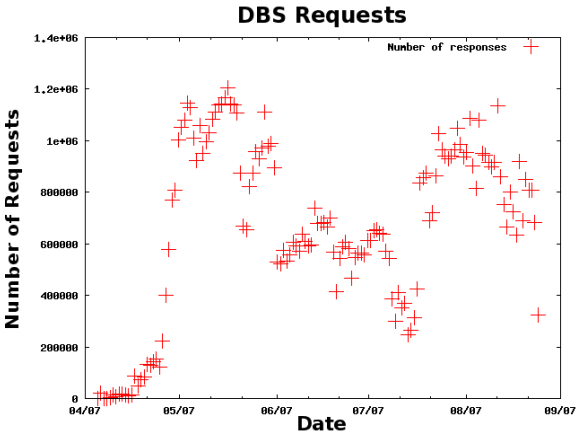

Figure 8. Requests per day made to the DBS server at CERN.

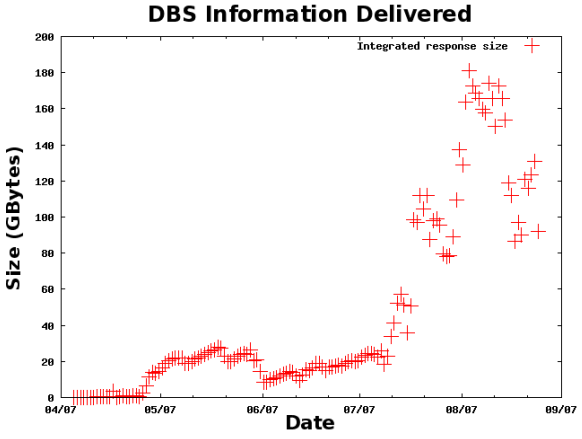

Figure 9. Delivered information per day made by the DBS server at CERN.

The operation of the system has given the DBS Team valuable experience and understanding of the usage patterns of, and bottlenecks in, the system. A sense of the magnitude of the use of DBS within CMS can be gleaned from statistics for the Global instance. As of the end of August there were 740M Events, 321834 Files, 12063 File Blocks, 1431 Primary Datasets, and 2638 Processed Datasets. The Global instance uses almost 2 GB storage in Oracle divided equally into data and index space. Several of the local instances are much larger with the largest

nearly 13 GB in size. The daily number of requests and response sizes are charted in Fig. 8 and Fig. 9. It is worthy to note that on more than one occasion the system has served more than 1.2M daily requests. As additional files and datasets are added to the catalog the size of the responses tend to increase as the majority of requests are for lists of these items and in August several days were marked by the delivery of around 180 GB of information. It is also noteworthy that a significant increase in the size of requests began after a version upgrade in July. This increased throughput is understood and possible improvements to reduce the size of payloads are being considered.

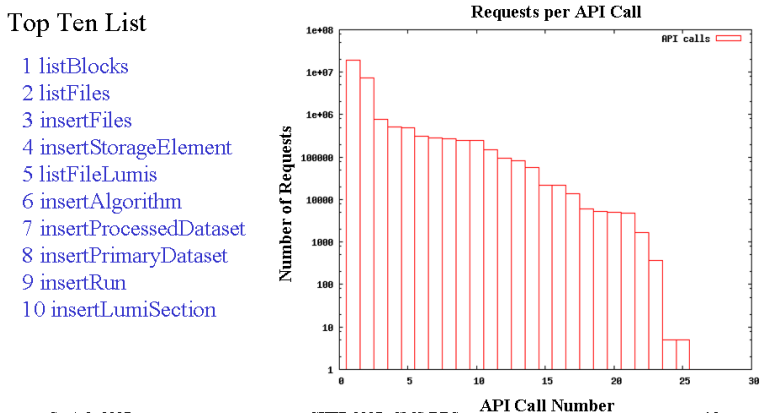

Figure 10. Frequency of api calls made to the DBS server at CERN.

It is interesting to study the distribution of the various API Calls made. Fig. 10 is the frequency of calls made to each api and shows that, by an order of magnitude, the majority of the requests are for lists blocks and files. The complete list of DBS API Calls is shown in Fig. 11. Targeted improvements to the system are being planned based on this and additional information.

(i) listBlocks
(ii) listFiles
(iii) insertFiles
(iv) listFileLumis
(v) insertStorageElement
(vi) insertAlgorithm
(vii) insertProcessedDataset
(viii) insertPrimaryDataset
(ix) insertRun
(x) insertLumiSection
(xi) listProcessedDatasets
(xii) updateFileMetaData
(xiii) insertRunInPD
(xiv) updateFileStatus
(xv) listRuns
(xvi) listDatasetParents
(xvii) listLFNs
(xviii) insertBlock
(xix) closeBlock
(xx) listDatasetContents
(xxi) insertDatasetContents
(xxii) listPrimaryDatasets
(xxiii) deleteSEFromBlock
(xxiv) updateProcDSStatus
(xxv) listStorageElements
(xxvi) listAlgorithms

Figure 11. List of API calls in descending frequency order as shown in Fig. 10

6. Summary

DBS is used by CMS to record and track the history of all event data. It is designed to accommodate the data processing and event data models of CMS, and integrate with the workflow tools employed by the experiment. The current deployment has demonstrated good

functionality, scalability, stability and performance. Examination of usage patterns, performance metrics and additional use cases are being used to plan future enhancements.

7. Acknowledgments

We would like to thank DOE and NSF for support of the project under which this work was conducted. This effort is the result of many within CMS working together to anticipate the needs of the Experiment. The CMS Computing and Offline Software organization, and specifically the Data Management group has provided the overall data model into which DBS was crafted.

References

[1] The CMS Dataset Bookkeeping System DBS web site: https://twiki.cern.ch/twiki/bin/view/CMS/DBS-TDR
[2] P. Elmer, "The CMS Data and Workflow Management System," CHEP 2007.
[3] D. Spiga, "CRAB (CMS Remote Analysis Builder)," CHEP 2007.
[4] D. Evans, et. al.,"CMS MC Production System Development & Design," CHEP 2007.
[5] L. Tuura, et. al.,"Scaling CMS data transfer system for LHC start-up," CHEP 2007.
[6] V. Kuznetsov, A. Dolgert, C. Jones, D. Rily,"A multi-dimensional view on information retrieval of CMS data," CHEP 2007.
[7] C.D. Jones, V. Kuznetsov, D. Riley, G.J. Sharp, "Eventstore: Managing Event Versioning and Data Partitioning Using Legacy Data Formats," CHEP 2004.
[8] V. Kuznetsov, A. Dolgert, "Searching for CMS Data: Rapid web development using Python and AJAX," CHEP 2007.