

HEP Metadata Schema

Experiments' Experience

Version 1.01: May 2005

Tim Barrass
Peter Kunszt, Gavin McCance, Krzysztof Nienartowicz, Ricardo Rocha
Randolph Herber, Adam Lyon, Julie Trumbo
Mòrag Burgon-Lyon, Tony Doyle, Steven Hanlon, Paul Millar
Solveig Albrand, Jerome Fulachier
James Werner
Carmine Cioffi,
Stephen Burke, Owen Synge

*Bristol
CERN EGEE project
Fermilab
Glasgow
LPSC, Grenoble
Manchester
Oxford
RAL*

Objective

Metadata is “ancillary information about stored data, which aids a user in handling it, describing it and understanding what it contains.”[1] This document surveys metadata schema used in new and mature HEP experiments identifying the commonalities and differences between each approach. Implications, such as schema complexity, user functionality, and extensibility are also examined.

Introduction

Each HEP experiment has adopted a database schema for metadata storage. This document looks at the metadata schema used a number of HEP experiments: ATLAS (AMI), BaBar, CDF & D0 (SAM), CMS and LHCb.

Information from several experiments has been used to compile this document:

- ATLAS (AMI) schema [2];
- BaBar schema [3], [4], [5];
- CDF & D0 (SAM) schema [6];
- CMS TMDB schema [7]; CMS RefDB[8];
- LHCb schema (Bookkeeping chapter) [9],[10];

ATLAS (AMI)

These notes refer to the Atlas Data Challenge 2 schema in AMI.

Dataset Handling

- 1.1 Specify a new dataset
- 1.2 Read metadata for datasets
- 1.3 Update metadata for dataset
- 1.4 Resolve physical data
- 1.5 Download dataset to local disk

Dataset handling use cases are supported mainly through the 'dataset' table, though some metadata is stored in other tables, which are related to the 'dataset' table through the db_model table. In particular, the 'dataset_property' table allows an arbitrary number of named

numeric values to be attached to a dataset, while 'dataset_keyword' allows text tags to be added. There are also 'dataset_extra' and 'dataset_added_comment' which provide for more flexible additions to datasets. These tables allow the implementation of the custom metadata described in 1.3. Reading all metadata for a dataset, as described in 1.2, is possible through the use of the 'db_model' table. This use case is also helped by the 'db_field' table, which allows web interfaces to be built easily.

Use cases 1.4 and 1.5 are supported through the 'metadata' field of the 'logicalFile' table. This field holds XML-formatted metadata that allows the file to be accessed via the POOL File Catalogue.

Analysis

- 2.1 Run a physics simulation program
- 2.2 Select a subset of a dataset
- 2.3 Run an algorithm over an input dataset

The DC2 schema is set up to deal with large scale, organised Monte Carlo production, so does not perfectly support use case 2.1, as written. However, simulation-related metadata is in the database. The 'dataset' table contains fields describing the production time, the detector simulation used and the CPU time consumed.

At present the possibilities for selecting subsets of datasets (use case 2.2) are limited. However, it is possible to select all logical files in a dataset and filter those to determine a subset. Event-level selection is outside the scope of AMI.

The 'task' and 'transformation' tables, with their associated parameter tables, support use case 2.3. Using these tables, the user can fully specify an algorithm to be run on an input dataset. The 'dataset' table also contains a link to the 'task' table, recording how the dataset was created.

Job Handling

- 3.1 Submit a job to the grid
- 3.2 Retrieve/Access output of a job
- 3.3 Estimate resource cost of running a job
- 3.4 Monitor progress of a job
- 3.5 Repeat a previous job

Job handling is outside the scope of AMI. However, AMI does store tables dealing with Tasks and Transformations. A transformation is the general form of a process which can be carried out on a dataset, defining the parameters which must be set. A task is an instance of a transformation and specifies the values of the parameters. Using this information it would be possible to repeat any task, as per use case 3.5.

An interface is available (actually it's still a prototype) which allows physicists to select a transformation, and to provide the parameters to define the task. A prodsys application will read the tasks thus defined, and convert the tasks into a set of jobs. The job details are entered in production system, and should then be executed. The production system itself should take care of resubmitting failed jobs.

Currently AMI monitors the state of files produced by tasks by reading the production system database, but this mechanism is under discussion, as is the complete ATLAS prodsys and data management architecture.

As mentioned previously, the DC2 'dataset' table does include information on the CPU time used in producing datasets. This could potentially be used as an input for resource cost estimates for use case 3.3.

BaBar

BaBar is a SLAC based experiment has been collecting data since 2001 accumulating over a petabyte(PB) of physics data. It has undergone two major design iterations initially using a commercial ODBMS (Object oriented DataBase Management System). This was replaced by an open-source, file based, object persistent system. The data is logically organised in several domains, with no direct cross-references. One of the key domains is the eventstore, which holds the bulk of BaBar data. By contrast, the bookkeeping is done using an RDBMS holding only 3GB of data.

The BABAR database management system is comprised of three basic domains:

The Conditions database manages and tracks the conditions under which experimental data were acquired. It includes such things as electronic calibrations, detector calibrations (e.g. drift time relationships) and detector alignments. It is designed to allow for a complete description of the detector as a function of time, where different components of the detector may exhibit different time variances.

The Event Store manages and tracks the experimental data from the initial raw data produced by the experiment or by simulations through the various reconstruction and physics analysis processing phases and selections that result in the publication of physics papers. This has to deal with the creation of sub-samples of events that meet different criteria, reprocessing of data following updated conditions information and management of experiment-wide event samples as well as other samples that are specific to a particular work group or individual.

Online databases are databases that are specific to the online system that do not fall into the other domains. This includes the resource manager database that is used to allocate resources between concurrent partitions.

BaBar Use Cases

Specify a new Dataset

Datasets are specified at SLAC and produced in tier1. They are catalogued in bookkeeper after a quality assurance process and made available for the users.

Read Metadata for Datasets

Babar is an ongoing experiment, and all necessary information about any given dataset is available using Bookkeeper. Metadata about the different classes of data are available using the internet, hypernews, and header files for the user.

Update Metadata for a Dataset

There are groups responsible for updating information in production. There are several quality assurance process and validation to guarantee the correctness of information and consistency.

Resolve Physical Data

Babar users resolve physical data merging components that are site independent (logical file name) with site dependent (physical file name) to give flexibility to systems managers and cope with heterogeneous environments. This information is stored in Bookkeeper and configuration files, transparent for the users. A user may resolve the information about a dataset by typing `BbkDatasetTcl datasetname`. The resolved information is stored in tcl files.

Download a Dataset to Local Disk

Bookkeeper has tools to mirror the central database at SLAC, mark datasets and import data to local disk. The process manages updates to the relational database with dataset information, and stores data in the correct place following configuration files.

Run a Physics Simulation Program

Babar framework is an integration of several packages that allow users to run monte carlo production using packages installed in AFS. Parameters are defined in tcl files, using environment variables to configure the environment.

Select a Subset of a Dataset

Every dataset is stored as a sequence of files of 2GB. When users require Bookkeeper to resolve physical data for the dataset, several sequential tcl files are generated depending the number of events the user wants in each tcl file. Users can run several jobs (one for each tcl file) in parallel using a grid.

Run an Algorithm over an Input Dataset

Babar software was structured as a framework where users have to code the low level routines to perform the necessary analysis for each event stored in the dataset. The software, previously installed in the users computer, is stored in AFS providing all necessary packages and tools. To run the analysis software the user provides the tcl file with necessary parameters and file names, obtained from Bookkeeper. The framework executes all necessary logical relation between the condition and configuration databases and datafiles.

Submit a Job to the Grid

The EasyGrid Job Submission software is an intermediate layer between the Grid, where the resources can be found, and the user, who has a pre-developed software analysis (written in C++) which they run on a chosen dataset. EasyGrid's first task is to find what event files are in the dataset. For each dataset there is a metadata file containing the names of the event files. These physical files are registered with the RLS, with several logical file names in the format `datsetname_CEJobQueue` assigned to them as aliases, showing the CEs which contain copies of that dataset. NB: If a CE holds any of the files in a dataset it holds all of them. Searching all the aliases for a dataset name provides a list of CEs to which jobs can be submitted. The next stage is generation of all necessary information to submit the jobs on the LCG Grid. This is done by the GGenerator of Resources Available (Gera) which produces the Job Description Language (JDL) files, the script with all necessary tasks to run the analysis remotely at a WN, and some grid dependent analysis parameters. The JDL files define the input sandbox with all necessary files to be transferred, and a WN balance load algorithm matches requirements to perform the task optimally. When the task is delivered in the WN, scripts start running to initialise the specific Babar environment, and the analysis software is downloaded. NB: The analysis executable is allocated in the SE and its' logical file name (LFN) is also catalogued in the RLS so any WN need download it only once.

Retrieve/Access Output of a Job

Everytime users require easygrid and the jobs already have been finished, easygrid recovers results from LCG grid into users directory. Users do not have to deal with tokens.

Monitor the Progress of a Job

If the job have been submitted and did not finished, easygrid performs the status query to LCG grid and provide a formatted report.

Repeat a Previous Job

If the software crashes, LCG grid diagnostic is performed and the report is generated for the user. We do not consider automatic tools for resubmission to avoid waist of resources. There are tools to submit in a very easy way only datafiles that have crashed.

CDF & D0 (SAM)

The SAM (Sequential Access via Metadata) schema is complex, comprising of 130 tables. This data handling system is used by multiple experiments, including CDF and D0 in production. The SAM database design organises the majority of these tables into four areas of general functionality: Core; Job Handling; Experiment Specific and Metadata Query. A further 14 tables

are used to interface different sections of the schema. Not every table is used by every experiment, and where a table is not required it exists but is empty. This allows a single database team to support the SAM schema for all experiments.

The Core area is the largest comprising 57 tables. Of these the majority are devoted to File Handling. Other major areas include Processes, the mechanism by which SAM allows files to be consumed by an application and the SAM Station which oversees the setting up of Processes to consume Files. Caching is also a Core function.

Job Handling tables are divided into Monte Carlo for storing request details, Batch Processing which although implemented in the database as yet has no API and General Support tables for authentication, and authorisation of the various users, groups and administrators.

Experiment Specific tables are grouped into Events tables for tracking physics events. These are used by D0 but not CDF. The Luminosity tables holds luminosity block, version and version types. The schema then ties this luminosity data to the Runs and Data Files. SAM touches on data Streams and Triggers. Most of the streaming and trigger data is kept in other applications. SAM records the physical data stream, the trigger stream and the trigger list(s).

In SAM, files are imagined to exist in a multi dimensional space specified by their metadata. The metadata query service manipulates constraints on these dimensions, defining a portion of metadata space containing files of interest. The tables used for these functions are stored in the Query Metadata and Dimensions areas.

The schema used in SAM is fixed and any additions to the database require thorough testing before release into production. To allow users to add dimensions to a running production system a number of extra tables have been incorporated. Code development to utilise these tables fully is underway.

The SAM schema has evolved over a period of 8 years. It caters to the vast majority of functionality requested by its users. This continued bootstrapping approach of adding in extra tables where required has led to some inefficiencies. Although redesign would likely result in a better system, the time to redesign SAM completely is prohibitive. However this mature schema can provide new experiments with useful information about the functional areas to cover, and also where metadata rulings such as “not null” on pertinent information can ensure data stays useful.

CDF Use Cases

Specify a new Dataset

The Dataset *Definition* Editor web-browser tool can be used to specify new datasets. A user simply points their web browser to http://fcdfcaf550.fnal.gov:19655/sam_dataset_editor/. Once logged in, a user can view the files used to make up any dataset. Dialog boxes may be filled with information about the dataset such as which group it will belong to. A straight forward syntax is used to list the names of the files to be included in the new dataset. e.g.

FILE_NAME = jh02237c.00e3bot0 or FILE_NAME = jh02237d.0103bot0

Definitions for existing datasets may be cloned, and the clone altered before saving. Alternatively the *sam define dataset* command can be used, e.g.

sam define dataset -group=b-physics -defname=jbot-2files -filename=jh02237c.00e3bot0

The tables used for this task are mainly in the Files section of the Schema.

Read Metadata for Datasets

A user can complete searches on file metadata using *sam translate constraints* command line queries which queries mainly the Files section of the schema.

Update Metadata for a Dataset

CDF do not allow metadata to be changed, however, a dataset can be cloned and the name and

metadata changed, as detailed above.

Resolve Physical Data

The *sam locate* command line command can be used to find the physical locations of either an individual file, or the files in a dataset.

e.g. *sam locate jh02237c.00e3bot0*

A path for every instance of the file will be returned at the command prompt.

Download a Dataset to Local Disk

A simple SAM project can be used to download a dataset. A short *tcl* file is prepared that uses *DHInput* to specify the name of the required dataset. The script is called by a second script which is run from a command prompt.

e.g. *./RunTest.sh*.

This starts a sam project which oversees the transfer of the dataset to the local sam cache. The interface to *DHInput* allows the requested dataset to be downloaded. Examples of both files are shown in Appendix A. This uses mainly the Station and Files sections of the Schema.

Run a Physics Simulation Program

Monte Carlo Simulation are generally run by submitting a tarball of scripts to a DCAF (Distributed CDF Analysis Farm). This uses SAM to fetch any data if required, and to store the output.

Select a Subset of a Dataset

The easiest way to select a subset of an existing dataset is to use the Dataset Editor and create a new dataset by cloning the original and removing the unwanted files.

Run an Algorithm over an Input Dataset

A physics simulation programme can be run using the method shown in **Download a Dataset to Local Disk**, with an expanded *tcl* file.

Submit a Job to the Grid

A job may be submitted to the grid using *jim_client*. A *jdl* (job description language) file is prepared specifying the input, log and error report file names. Members of the CDF collaboration are issued with a kerberos principal for secure access to the CDF computing systems. This principal can be used to generate a certificate for submitting grid jobs. The user may log into a kerberised computer with *jim_client* installation, prepare their kerberos principle and *jdl* file (e.g. *testfile.jdl*) then simply type:

```
samg submit testfile.jdl
```

Retrieve/Access Output of a Job

JIM jobs can be accessed, and the output and log files downloaded by pointing a web browser to <http://samgrid.fnal.gov:8080/>. First select the site the job was submitted to, and then the job itself. Job information is displayed, and all output and log files can be downloaded as a single *tar* file.

Monitor the Progress of a Job

As with the last use case, a user can view the state of a job (submitted/running/completed) using a web browser. For SAM jobs, further information, such as which files are currently being transferred, can be viewed using SAMTV (<http://ncdf151.fnal.gov:8520/samTV/current/samTV.html>).

Repeat a Previous Job

Although SAM has facilities to deal with repeating parts of a failed job, at the grid level a repeat job must be submitted manually at present.

CMS

PhEDEx (Physics Experiment Data Export) is used by the CMS experiment for data transfer and is comprised of:

- TMDB (Transfer management database)
- Transfer agents that manage the movement of files
- Management agents which assigns files to destinations based on site data subscriptions
- Transfer request management tools.
- Local agents for local file management.
- Web monitoring tools

The TMDB is made up of ten tables concerned with file transfer, lookup, routing, file bookkeeping and transfer state. In CMS, metadata is conceptually divided into CORBA metadata and Catalogue metadata, adhering to the basic assumption that catalogue metadata follows the files. It is copied into RefDB as part of production. Each data publishing site will insert metadata into the catalogue.

RefDB is the CMS Monte Carlo Reference Database. It is used for recording and managing all details of physics simulation, reconstruction and analysis requests, for coordinating task assignments to world-wide distributed Regional Centers, Grid-enabled or not, and tracing their progress rate. RefDB is also the central database that the workflow-planner contacts in order to get task instructions. It is automatically and asynchronously updated with book-keeping run summaries. Finally it is the end-user interface to data catalogues. RefDB is accessed via a web-server by the workflow planner tool (e.g. MCRunjob).

RefDB uses MySQL RDBMS. The tables are organised into the following functional areas: Software & Executional; Monitoring; Input Parameters; Dataset & Collection; Pile Up Conditions; Request & Assignment.

CMS Use Cases

The CMS experiment has yet to make a decision on which systems will be used. As such, the Use cases can not yet be applied for CMS.

LHCb

Metadata in LHCb is stored using a two schema strategy. The data is stored as key-value pairs in a Warehouse DataBase (WDB). A View of the data, using a different schema specialised to a specific task, is used to access the data for different purposes. LHCb uses two services to access these databases. A servlet service allows dataset selection based on job provenance using a web browser. An XML-RPC service allows the data in the WDB to be changed whilst also allowing access by GANGA to the bookkeeping database.

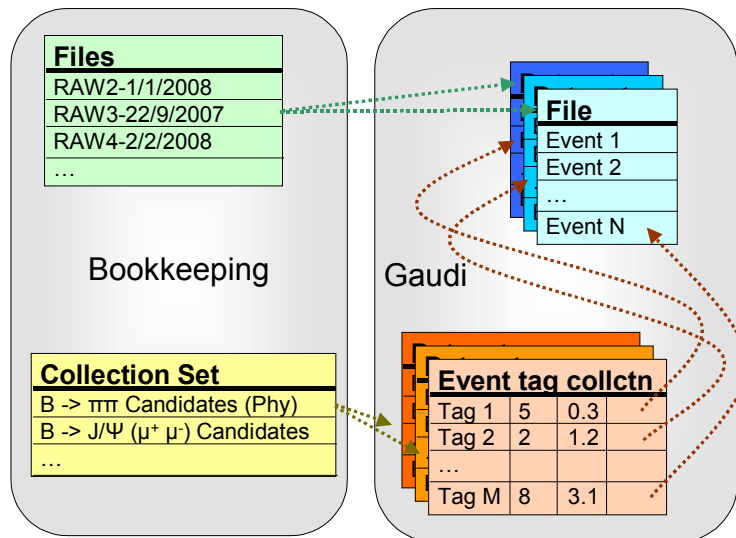
A bookkeeping domain is used by the LHCb experiment. Data is stored in three core areas: File Information; Job Information; and Quality Information. On top of the file and job centric system, a system called Gaudi allows users to select on an event basis using specialized Ntuples called Event Tag Collections.

Bookkeeping is accessible to any Gaudi application using the standard Gaudi mechanism:

- A Gaudi service encapsulates the access to the database. Accessible interfaces implement all the required functionality to edit jobs, files, qualities, types and all sorts of parameters.
- No database internal details are exposed.

LHCb view a dataset as a collection of files with common properties. An atomic dataset contains only one file. Links between objects are not filled at creation time. The service resolves on request object dependencies such as the associations between files

and jobs.
The figure below shows this relationship:



The service must offer access to the data as well as edition capabilities. These two functionalities are reflected in two separated interfaces, both implemented by the service. This allows access to the data to be restricted independently of the database's access rights. The logical data model is shown below:

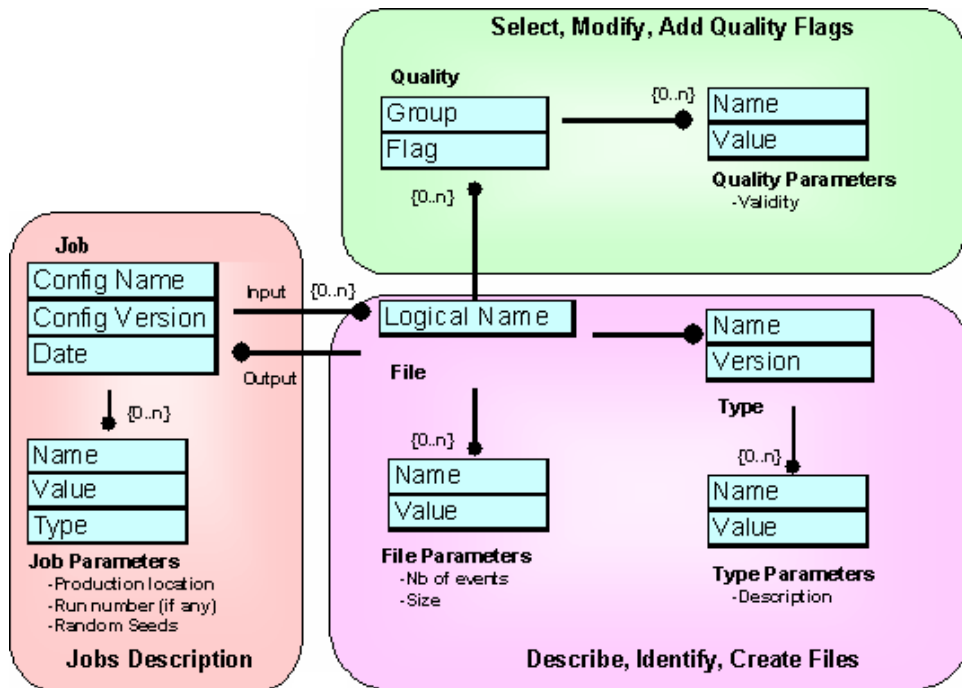


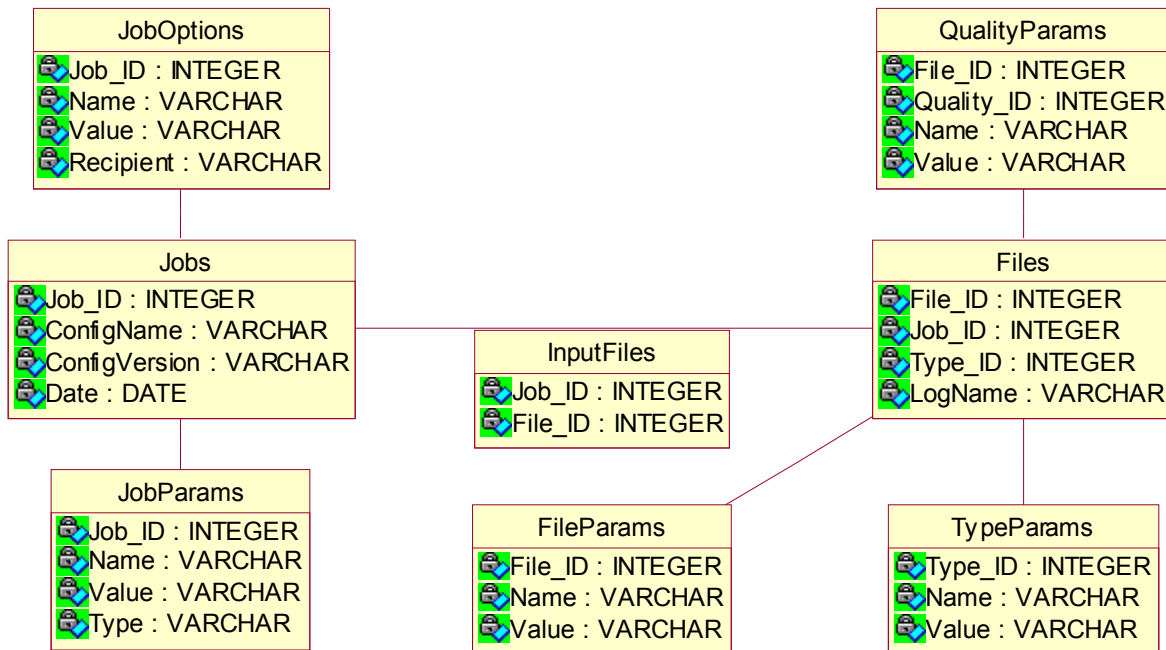
Figure 1. Logical metadata model.

The persistent data model describes the design of the database used to store bookkeeping data. This database is relational giving descriptions of the tables used and of their columns. Each bubble of the figure below is a table and the list of columns is given inside each one. The associations drawn as lines show where IDs are supposed to match. As an example, the Job_ID

in `_Job` and in `_JobOptions` are supposed to match.

The whole design was intended to build thin tables on the database side by removing all columns that would not be absolutely necessary for all cases. Lines in one of the parameter tables replace these columns. As an example, the number of events of a given file is only relevant in the case of a file containing data (not in the case of a log file). Thus, it was not added in the `_Files` table but will be a line in the `_FileParams` table with name "NbEvent" and value the actual number of event of the associated file.

This architecture takes advantage of the indexing capabilities of the underlying database to search for objects. As an example, indexes should be created in the `_InputFile` table, based on the `file_ID` and in the `_Files` table based on the `Job_ID`. This allows an efficient browse of the hierarchy of jobs and files.



LHCb Use Cases

Specify a new Dataset

LHCb does not use the concept of a Dataset as collection of files. For the purpose of the use cases below, a dataset is a file. The datasets are selected using the servlet service. The selection is done using a web browser based on the job provenance information.

Read Metadata for Datasets

The web service allows the user to go deep inside the metadata information of the dataset at selection time. Alternatively the XML-RPC service can be used to access the information from the application. LHCb also has a java implementation (BookkeepingSvc) with it's own API to manipulate (read/ write) the bookkeeping information.

Update Metadata for a Dataset

For a single private dataset the XML-RPC service or the BookkeepingSvc can be used. In the case of production, Dirac is used to send an XML file with all the metadata of the datasets created to the production manager which, after checking that the datasets are stored and accessible correctly, will move the information from the XML files to the bookkeeping.

Resolve Physical Data

The bookkeeping for the time being keeps information about the location and the Physical Filename of each file. Usually, for submission to Dirac, this information is copied into an XML

file catalog very similar to the POOL XML file catalog, which is sent to Dirac during the submission of the job.

Download a Dataset to Local Disk

When a user wants to download a file to his own pc he simply gets the Physical Filename from the bookkeeping system and copies the file to his HD. In the case of submission to Dirac ... (to be completed) .

Run a Physics Simulation Program

The user has to prepare the Job Option File for each of the Applications involved in the simulation process and then run the application from the command prompt. This is what a user has to do to run a private simulation program, in the case of data production the production manager will prepare all the jobs and submit to the Dirac system as below.

Select a Subset of a Dataset

Since the dataset is a file this use case does not apply.

Run an Algorithm over an Input Dataset

The Gaudi framework is used to create the application that implements the algorithm and the Datasets (files) to be read are specified into the Job Option File. The JOF will contain all the parameters to configure the job plus the list of datasets that the job has to read.

Job Handling Use Cases

Implimentation for the set of use cases dealing with grid submissions and retrievals have not yet been decided by LHCb.

Conclusions

New developments, such as the Global Grid Forum move to a web services model as the preferred basis for grid solutions are apparent in the systems presently under development by new HEP experiments. This in turn has effected the schemas used to store metadata. SAM, the oldest data handling system shown uses a single large schema, with functional areas to address each aspect of metadata. By comparison AMI allows users access to any database, caring little about the schema within. A further example is the approach adopted by CMS where two separate databases are used with distinct tasks, one for file transfer and the other for recording and managing all details of physics requests.

Appendix A – Code examples for CDF

Test.tcl

```
talk DHInput  
include dataset jbot0h  
cache set SAM  
show  
exit  
begin -new 10  
exit
```

RunTest.sh

```
source ~cdfsoft/cdf2.cshr  
setup cdfsoft2 5.0.0int2  
setup sam  
setenv SAM_STATION cdf-oxford  
setenv SAM_PROJECT Unique_Name120405  
AC++Dump Test.tcl
```

References

- 1: S. Hanlon et al., Unlucky For Some, The Thirteen Core HEP Metadata Use Cases, 2005
- 2: S. Hanlon et al., The AMI Database Schema, 2005
- 3: The BaBar Database Web pages:
<http://www.slac.stanford.edu/BFROOT/www/Public/Computing/Databases/index.shtml>
- 4: Jacek Becla, David Wang, Lessons Learned Managing a Petabyte, 2005
- 5: J. Werner, <http://www.physics.gla.ac.uk/metadata/index.php/BaBar>
- 6: M. Burgon-Lyon, J. Trumbo, The SAM Database Schema, 2005
- 7: T. Barrass, S. Metson, L. Tuura, CMS Data Handling: TMDB Schema and Database Interactions, 2004
- 8: V. Lefébure, J. Andreeva, RefDB: The Reference Database for CMS Monte Carlo Production, 2003
- 9: C. Cioffi, File Metadata Management System for the LHCb Experiment, CHEP04
- 10: C. Cioffi et al., Bookkeeping Architecture