File and metadata management for BESIII distributed computing

C Nicholson1, L Lin2, Z Y Deng3, W D Li3, X M Zhang3 and Y H Zheng1

1 Graduate University of the Chinese Academy of Sciences, 19A Yuquan Road, Beijing 100049, People’s Republic of China
2 Soochow University, No. 1 Shizi Street, Suzhou 215006, People’s Republic of China
3 Institute of High Energy Physics, 19B Yuquan Road, Beijing 100049, People’s Republic of China

E-mail: caitriana@gucas.ac.cn

Abstract. The BESIII experiment at the Institute of High Energy Physics (IHEP), Beijing, uses the high-luminosity BEPCII e+e− collider to study physics in the τ-charm energy region around 3.7 GeV; BEPCII has produced the world’s largest samples of J/ψ and ψ′ events to date. An order of magnitude increase in the data sample size over the 2011-2012 data-taking period demanded a move from a very centralized to a distributed computing environment, as well as the development of an efficient file and metadata management system. While BESIII is on a smaller scale than some other HEP experiments, this poses particular challenges for its distributed computing and data management system. These constraints include limited resources and manpower, and low quality of network connections to IHEP. Drawing on the rich experience of the HEP community, a system has been developed which meets these constraints. The design and development of the BESIII distributed data management system, including its integration with other BESIII distributed computing components, such as job management, are presented here.

1. Introduction
The BESIII experiment at the Institute of High Energy Physics (IHEP) of the Chinese Academy of Sciences uses the high-luminosity BEPCII double-ring electron-positron collider to study physics in the τ-charm energy region around 3.7 GeV. From the start of data-taking in 2009 to the end of 2011, BEPCII produced the world’s largest samples of J/ψ and ψ′ events, about 0.2 billion and 0.1 billion events respectively, giving a total annual raw data volume of about 200 TB. Another 1 billion J/ψ events have been produced in 2012, approximately 5 times the size of the previous dataset.

The currently available computing and storage resources for the BESIII experiment consist of about 3400 CPU cores, with 2.4 PB of disk storage and 4 PB of tape storage, at the central IHEP site. The requirements for handling the forthcoming data are much higher, at 6000 to 10000 CPU cores, 5 PB of disk and 10 PB of tape storage. At peak times in particular, providing this level of resource at IHEP is not feasible and so, following the example of most other High Energy Physics (HEP) experiments, an alternative distributed solution is required. BESIII is a relatively small collaboration, and so a simple, efficient solution is sought.
The BESIII distributed computing environment is proposed to incorporate grid, cloud and possibly volunteer computing resources with the existing IHEP batch farm to meet the data processing requirements of the experiment; in this paper, this distributed environment is referred to generically as the BESIII grid. DIRAC [1] has been adopted as the job management system for the BESIII grid, and Ganga [2] as the user job submission system, with a BESIII-specific plugin, GangaBoss, to handle the BESIII software [3].

In this paper, the file and metadata management components of the distributed data management system are presented. Section 2 describes the requirements of the BESIII experiment, constraints posed by the size and nature of the collaboration, and the design for a system developed in response to these requirements and constraints: BADGER, the BESIII Advanced Data ManaGER. Section 3 presents an evaluation of the technology choices considered for the core catalogs, namely AMGA [4] and the DIRAC File Catalog (DFC) [5]; Section 4 then describes the current status of a BADGER implementation based on DFC and integrated with DIRAC job management. Finally, Section 5 draws some conclusions and presents plans for future work.

2. Design of a File and Metadata Management System
2.1. BESIII Computing Constraints and Requirements

BESIII has until recently maintained a highly centralized, traditional HEP computing environment. There are approximately 350 collaboration members from 52 institutions; several collaborating institutions maintain their own batch farms for local production of Monte Carlo data and analysis activities, but all real data processing and most Monte Carlo production is done at IHEP, with the majority of users performing their analyses using the central batch system. There is little transfer of data between sites due to relatively poor network connectivity. Centrally stored data files are kept in well-defined directories on the local file system; user analysis datasets are defined and their location information propagated in an ad-hoc fashion by physics groups. While there is a clear need to expand computing resources by moving to a more decentralized environment, expertise and manpower available for development and maintenance of the BESIII grid are very limited, and most users have little or no experience of a grid computing environment. The constraints which must be taken into account in developing the BESIII grid, therefore, can be summarized as:

- low network connectivity
- limited manpower
- limited expertise, particularly at remote sites.

The BESIII grid in general must be designed to meet the following requirements:

- minimal data transfer between sites
- easy setup and maintenance for remote sites
- simple, intuitive user interface
- reliable and robust service.

Furthermore, the file and metadata management components have the following specific requirements:

- scalable up to the order of 10 million files and 100 concurrent clients
- searchable file-level metadata
- functionality for datasets, i.e. sets of files defined by the experiment
- user authentication and authorization for data and metadata operations
- good integration with the job management system.

2
It should be noted that only file-level and dataset-level metadata are considered in the current BESIII data management model; event-level metadata may be included in future as the BESIII event model continues to develop.

2.2. System Design

The basic computing model, due to the network restrictions, plans to keep all raw and reconstructed real data (RAW and DST data types respectively) at IHEP. All raw data processing and bulk reconstruction activities, as well as most real data analysis, will take place at IHEP. It is planned to use remote sites for Monte Carlo production, reconstruction and analysis, particularly at peak times when IHEP resources are fully utilized. For the BESIII grid, therefore, the primary use case is Monte Carlo production. The first priority for the data management system is support of this use case, followed by support for user analysis.

BADGER, the BESIII file and metadata management system, must therefore provide the following functionality:

- **File Catalog**: mapping of Logical File Names (LFNs) to Physical File Names (PFNs) of file replicas. Initially, only DST files are considered, as it is not planned to replicate RAW files.
- **Metadata Catalog**: mapping of files to associated metadata. The metadata schema is outlined in Section 2.3.
- **Dataset Catalog**: mapping of dataset names to list of files; a BESIII dataset consists of a set of files for analysis, typically defined by analysis groups.
- **User Interface**: expose relevant methods to users through graphical and command-line interfaces.

In addition, the system must expose a Python API to allow integration with the job management system via Ganga and DIRAC, and with the data transfer tools.

A conceptual diagram of the components and how they should interact with the job management system during a typical job is shown in Figure 1. A user should be able to interact with BADGER directly via its User Interface, to search for files which match metadata conditions, locations of file replicas, get the list of files in a particular dataset, and so on. ‘Power users’, such as production group managers, should be able to use this interface to add or modify datasets, modify metadata attributes, manage authorization groups and so on. Interaction with the job management system depends on the type of job. For Monte Carlo simulation, a job is submitted to Ganga (1), whereon it is split into a number of sub-jobs which are submitted to the grid via DIRAC (3). DIRAC handles the running of the jobs on the worker nodes (5), and registers the output data and associated metadata in the catalogs on job completion (6). Dataset information associated with the new files can then be registered as required (7). For analysis jobs, Ganga should first be able to query BADGER for the input files (2), whether by dataset or by metadata conditions. The job should be split according to the number of input files and sub-jobs submitted to DIRAC (3). DIRAC queries BADGER for the physical locations of input files (4) and schedules sub-jobs to sites accordingly; the rest of the workflow then follows the same steps as the Monte Carlo production case. To reduce data transfer, it is planned to always send jobs to sites where the input files are present.

2.3. Metadata Schema

Table 1 shows the metadata attributes defined for files and Table 2 those defined for datasets. Certain attributes (data type, event type, resonance, experiment number and software version) are assumed to be the same for all files in any given dataset and can therefore also be considered as dataset-level metadata. The attribute data types given in the tables are based on a nominal MySQL implementation and may vary slightly in future implementations.
**Figure 1.** Conceptual diagram of BADGER components and interaction with job management system. Numbers in italics refer to steps in the job workflow, which are described in the text.

| Attribute            | Data type | Description                                                                 |
|----------------------|-----------|-----------------------------------------------------------------------------|
| GUID                 | varchar(32) | Globally unique file ID                                                     |
| LFN                  | varchar(100) | Logical File Name                                                            |
| Data type            | varchar(10) | BESIII data format (DST / RAW / TAG)                                        |
| Event type           | varchar(10) | BESIII event type (real, skim, various types defined by physics groups)     |
| Resonance            | varchar(10) | Resonance at which accelerator was running (J/ψ, ψ', ψ'', etc)              |
| Experiment number    | varchar(10) | Internal BESIII data-taking attribute                                       |
| Software version     | varchar(10) | Version of BOSS software used in reconstruction                            |
| runL                 | int        | Lowest run number in file                                                   |
| runH                 | int        | Highest run number in file                                                  |
| Stream ID            | varchar(10) | Internal BESIII data-taking attribute for MC data                           |
| File size            | int        | Size of data file                                                           |
| Number of events     | int        | Number of events in file                                                    |
| Status               | int        | Data is good / bad / other                                                   |
| Creation time        | timestamp  | Time registered in catalog                                                  |
| Description          | varchar(100) | Extra notes or user-defined metadata                                        |

Such a metadata schema gives, for the file-based metadata, a nominal metadata size per file of 324 bytes. Estimating an upper limit for the number of files BADGER will need to handle
Table 2. BESIII dataset-level metadata attributes

| Attribute name | Data type      | Description                                                                 |
|----------------|----------------|----------------------------------------------------------------------------|
| Dataset ID     | varchar(32)    | Globally unique dataset ID                                                 |
| Dataset Name   | varchar(100)   | User-friendly dataset name                                                 |
| Group ID       | int            | Unique identifier to physics group which owns dataset                       |
| Number of files| int            | Number of files in dataset                                                 |
| Number of events| int           | Total number of events in dataset                                          |
| runL           | int            | Lowest run number in dataset                                               |
| runH           | int            | Highest run number in dataset                                              |
| Data size      | int            | Total size of dataset                                                      |
| Creation time  | timestamp      | Time registered in catalog                                                 |
| Modification time| timestamp    | Time of last modification                                                   |
| Description    | varchar(100)   | Extra notes or user-defined metadata (including BOSS job options used)      |

in the next two years as 10 million gives a total size for the file-based metadata of about 3GB, while the total size of dataset-level metadata is even smaller. It is therefore clear that hosting the metadata catalog will not require a large amount of storage, unless event-level metadata is added in future.

3. Evaluation of Catalog Technologies

In Section 2, the conceptual design for the BADGER system and its metadata schema were outlined. In implementing this system, AMGA [4] and DFC [5] were both identified as possible solutions for the BADGER catalogs, in particular the metadata catalog. A careful evaluation was conducted of their relative strengths and weaknesses with respect to the BESIII use cases, considering both performance and functionality.

Two identical server machines were set up (HP Proliant DL180 with 2.40 GHz 2×4-core CPU, 16GB RAM and 4×450GB hard disk), one running AMGA and one running DFC. Both used a MySQL backend database. The existing BESIII DST data set was loaded into both catalogs in a hierarchical directory structure, reflecting the BESIII file naming scheme, of the format:

/bes/File/<resonance>/<software>/data/<event type>/<expt. no.>/filename
/mc/<event type>/<expt. no.>/<stream ID>/filename

For the AMGA installation, the relevant file metadata were added to each entry. DFC allows metadata to be added to directories as well as files, so where all the files in a directory had the same value of a metadata attribute (e.g. software version, event type, experiment number etc.), the metadata was associated with the directory rather than the file entry. The dataset metadata has not yet been fully implemented.

3.1. Evaluation of AMGA and DFC Functionality

AMGA and DFC both offer metadata catalog functionality. AMGA is designed specifically as a metadata catalog, to be used in a grid environment in conjunction with other data management components, although the catalog structure could also allow it to be used more generally. DFC, on the other hand, is designed as a file catalog with metadata functionality added. Both offer hierarchical directory-like organization of data, user management, authentication and
authorization, Python API, command-line interface (CLI), and so on. The following evaluation reflects the authors’ experience using both catalogs, with respect to the BESIII environment.

3.1.1. AMGA
AMGA is a robust, mature product with powerful metadata functionality. There is a richer set of metadata operations, for example, than is currently available with DFC. There is a choice of database backends, with PostgreSQL being the default, although the authors found that initial installation with MySQL was non-trivial. While running, the service was found to be very stable. Various server parameters, such as the maximum number of processes, are easily configured. The CLI is available in both interactive terminal mode and batch mode, which is very useful for scripting. The Python API, however, was found to be limited with respect to the features required for BADGER. The BESIII file metadata schema was easily implemented.

The main disadvantage of AMGA from the BESIII point of view is the difficulty of integration with the DIRAC-based job management system. DIRAC and Ganga are both Python-based, so the lack of a strong Python API would lead to considerable work being required to achieve good integration. Using AMGA would also require either separate installation and management of file and dataset catalogs, or designing a file and dataset schema which could be implemented within AMGA. The more advanced features of AMGA, such as federated metadata catalogs and replication across sites, are not likely to be required by BESIII.

3.1.2. DFC
DFC has been developed as a DIRAC-native catalog for the management of file replicas, with metadata operations developed on top of that. It is continuing to develop rapidly, with frequent releases providing new functionality. The database used is MySQL. As DFC is provided as part of the DIRAC installation, and does its own database setup, the authors found it straightforward to install. The service was also found to be very stable. Multiple service instances can be run on the same server, allowing efficient use of CPU – for example, on a multicore CPU one can run several DFC instances, utilizing more of the resources. There is a rich Python API. As well as an interactive terminal CLI, there are a number of batch commands available for individual catalog features (such as file replication, getting metadata for a given LFN, etc); however, not all the catalog functionality is available in batch mode, making it harder to use for scripting, for example. The interactive CLI is currently less robust than AMGA’s.

One key feature which makes DFC very attractive from the BESIII point of view is the combination of file and metadata catalog functionality in one service. DFC also provides support for datasets, in the form of dynamic ‘meta-sets’, where a particular metadata query can be saved with some name. If a user requests the list of files in a meta-set, the query is re-run and the list of entries matching that query is returned. While the BESIII collaboration has still to determine the details of its dataset model, dynamic datasets implemented through DFC’s meta-sets seem a promising option. Finally, as DFC is native to DIRAC, integration of BADGER with job management is straightforward.

3.2. Evaluation of AMGA and DFC Performance
For the performance evaluation, the main criterion considered was query time, testing the two catalogs as the catalog size increased, and as the number of concurrent clients increased. For these tests, the existing BESIII data sample of about 240,000 files was multiplied first to 1 million (exact number about 970,000) then to 2 million (exact number about 2,110,000) catalog entries by registering the same files multiple times. The BESIII data sample is estimated to grow to about 2 million files in the next 2 years, so this is a reasonable benchmark, although testing up to 10 million entries would be ideal. Multiple concurrent client processes were run from a set of non-dedicated IHEP machines, running a variety of metadata queries in a continuous loop, while the process for which timing measurements were made was run from a separate, dedicated machine. The AMGA and DFC services were run on separate, identical dedicated servers as
detailed above. AMGA was configured to allow a maximum of 140 concurrent processes. DFC was run with 8 instances of a maximum 50 threads each. The maximum number of concurrent clients tested here was 101, well within the limits of both these configurations. The test query was chosen to return a list of 100 entries matching a particular metadata query. For each data point, the average of 5 measurements was taken.

Figure 2 shows the query time as the number of concurrent clients increases, when the catalog has about 2 million entries. It was found that when the number of clients is below 20 (approx.), AMGA query time is about 10 times faster than DFC, most likely due to overheads in the DFC connection. However, as the number of clients increases, both catalogs show reduced response time, giving approximately equal performance. It is noted that with a high number of clients, query time is highly variable; however, with the average query time still under 1 second, it is acceptable for BESIII.

Figure 3 shows the query time as the size of the catalog is increased, when the number of clients is 1, 11 and 101. It is clear that for a low number of concurrent clients, both AMGA and DFC show a very stable query time as the number of entries is increased. With 101 clients, however, the size of the catalog has a significant impact on the query time, which increases approximately exponentially for DFC. Further investigation is required to examine the behaviour beyond 2 million entries; the current performance is acceptable, but if significant degradation of performance occurs if the catalog is increased to the order of 10 million entries, further optimisation will be required.

The server CPU and memory usage were also monitored as the catalogs were queried. Figure 4 shows the variation of CPU load as the number of clients increases, with about 2 million entries per catalog. The CPU load for user processes was measured with the \texttt{vmstat} utility, and average of 9 measurements taken at 1-second intervals for each data point. It can be seen that CPU usage rises quickly for AMGA and saturates at about 82% when there are 30 or more concurrent clients. DFC CPU usage rises more slowly and saturates at about 89% when the number of clients reaches 60. For both servers, disk utilization (measured with \texttt{sar}) remained low throughout. For a production service, further tuning of both MySQL, DFC service and operating system parameters, and perhaps load-balancing across two servers, will be required to attain an optimal configuration.
3.3. Evaluation Conclusions

Based on the evaluation presented above, it is clear that both types of catalog have advantages and disadvantages for BESIII data management. Both give a satisfactory level of performance for projected BESIII needs. The broader functionality of DFC, however, and its close integration with job management, make it a more suitable fit for BESIII. This solution also eases pressure on manpower, as there are fewer databases and services to manage and maintain. DFC is therefore selected as the core catalog for the BADGER service.

4. Prototype BADGER Implementation

A prototype implementation of BADGER has been developed using DFC as the core catalogue. A BADGER API has been defined and implemented in Python, wrapping some of the DFC API and adding a layer of functionality to register and query metadata, files and datasets.
according to BESIII requirements. BESIII simulation jobs run through Ganga and DIRAC can now successfully register output files and their metadata in BADGER; analysis jobs can query BADGER by metadata or by dataset name, to get a list of required files.

Current work is focused on selection and integration of a data transfer tool to improve data transfer between IHEP and remote sites, continued development of the API, particularly in relation to manipulation of datasets, and refinement of the deployment model towards a production service.

5. Conclusions and Future Work

An outline of the file and metadata management requirements of the BESIII distributed computing environment have been presented, highlighting the need for a solution which is scalable within the projected limits of the experiment, robust, easy to set up and maintain, intuitive for users, with functionality for the querying and management of files and datasets by metadata attributes, and integrated with the job management system. The design of a tool incorporating file, metadata and dataset management, BADGER, was presented, and an evaluation performed of AMGA and DFC as possible options for the core BADGER catalogs. DFC meets the requirements of the experiment, and has been chosen as the central catalog, with a prototype BADGER system now developed around DFC and successfully used in conjunction with the job management system. Further investigation should be done on optimizing DFC performance, especially with large numbers of connections. Work on BADGER is ongoing, with the development of BESIII datasets and integration of data transfer functionality; future work will also include development of a graphical user interface.

Acknowledgments

The authors would like to thank Andrei Tsaregorodtsev and Ricardo Graciani for their help with the DFC, and many valuable discussions; Soonwook Hwang and the AMGA developers for helpful discussions about AMGA; Alexey Zhemchugov for helpful discussions on the BESIII computing model; and colleagues at the IHEP Computing Centre for their support. This work was part funded by the Chinese Academy of Sciences Fellowship for Young International Scientists under grant number 2011Y1JB07.

References

[1] Tsaregorodtsev A et al. 2010 J. Phys: Conference Series 219(6)
[2] Mościcki J et al. 2009 Computer Physics Communications 180 2303 – 2316 ISSN 0010-4655 URL http://www.sciencedirect.com/science/article/pii/S0010465509001970
[3] Antoniev I et al. 2012 Int. Conf. on Computing in High Energy and Nuclear Physics (New York)
[4] Santos N and Koblitz B 2006 Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 559 53 – 56 ISSN 0168-9002 URL http://www.sciencedirect.com/science/article/pii/S0168900206002308
[5] Tsaregorodtsev A and Poss S 2012 Int. Conf. on Computing in High Energy and Nuclear Physics (New York)