# The High-Level Dataset-based Data Transfer System in BESDIRAC

**T Lin[1,2], X M Zhang[1], W D Li[1] and Z Y Deng[1]**

[1] Institute of High Energy Physics, 19B Yuquan Road, Beijing 100049, People's Republic of China

[2] University of Chinese Academy of Sciences, 19A Yuquan Road, Beijing 100049, People's Republic of China

E-mail: `lintao@ihep.ac.cn`

**Abstract.** Data transfer is an essential part in grid. In the BESIII experiment, the result of Monte Carlo Simulation should be transfered back from other sites to IHEP and the DST files for physics analysis should be transfered from IHEP to other sites. A robust transfer system should make sure all data are transfered correctly.

`DIRAC` consists of cooperation distributed services and light-weight agents delivering the workload to the Grid Resources. We reuse the most functionalities supplied by `DIRAC`. `BESDIRAC` is an extension to `DIRAC` for BES specified. In `BESDIRAC`, a Dataset-based Data Transfer System is developed. In this paper, we present the design of this system and its implementation details.

## 1. Introduction

The BESIII experiment [1] at the Institute of High Energy Physics (IHEP) of the Chinese Academy of Sciences (Beijing, China) uses the high luminosity BEPCII double-ring electron-positron collider to study physics in tau-charm energy region around 3.7GeV. It has accumulated the world's largest datasets at charm threshold and the integrated luminosities of the data taken at $\sqrt{s}$ = 3.650 and 3.773 GeV are measured to be $(44.49 \pm 0.02 \pm 0.44)$ pb$^{-1}$ and $(2916.94 \pm 0.18 \pm 29.17)$ pb$^{-1}$ [2]. The current computing and storage resources for the BESIII experiment consist of about 4500 CPU cores, with 3 PB of disk storage and 4 PB of tape storage at the central IHEP site. The simulation and reconstruction are mostly performed at IHEP batch farm. The details about mass production can be found in Table 1, which shows both simulation and reconstruction jobs.

When more and more processing data is accumulated, the resource at IHEP is not able to support the data production of the experiment any more. An alternative distributed solution has been considered. The basic computing model is proposed to incorporate grid and cloud computing resources with the existing IHEP batch farm. In this model, most computing tasks including the raw data processing, MC production and analysis are performed at the central site, IHEP. However, a fraction of MC generation will be done at remote sites to meet the peak demand of CPU resources. The result of MC simulation should be transferred back from other sites to IHEP for reconstruction, while the DST files for physics analysis should be transferred to other sites. In the future, reconstruction can also take place in remote sites, so the random trigger files also need to be transferred.

**Table 1.** Jobs in BESIII mass production

| | Simulation | Reconstruction |
|---|---|---|
| Events | 50000 | 50000 |
| Input | a few parameters | 255-350MB rtraw + 1.5-2GB Random Trigger |
| Output | 255-300MB rtraw | 900MB dst |
| Time | 14-19 hours | 4-6 hours |

In this paper, the design and implementation of the Dataset-based Data Transfer System are presented. In Section 2 we describe why we use `DIRAC` to create this system and what the architecture of the transfer system is. The transfer system kernel, the accounting and the user interface are described in Section 3. Then in Section 4, we show the result of our test. Section 5 is devoted to conclusion and outlook.

## 2. Architecture Overview

### *2.1. DIRAC Framework*

`DIRAC` [3, 4] has been adopted as the job management system and data management system for the BESIII grid [5], since it is a complete grid solution for a community of users such as the LHCb collaboration. It consists of many cooperating distributed services and light-weight agents within the same `DIRAC SEcure Transport` (`DISET`) framework following the grid security standards. Modular design applied in various system components allowed to quickly introduce new functionality and make component modifications without affecting other components.

### *2.2. The BESDIRAC Project*

Like LHCb, the BESIII experiment also needs experiment-specific functionalities in its grid. The `BESDIRAC` [7] is developed to provide these functionalities based on the underlying framework of `DIRAC`. The first level is the system name. The second level contains `Client`, `Service`, `Agent`, `DB` and auxiliary classes. Also, there are `Web` directories if web portal is needed. The `DIRAC` will automatically load these extensions when `BESDIRAC` is enabled by the `DIRAC`'s configuration service.

The `BADGER` (`BESIII Advanced Data ManaGER`) is the first sub-system in `BESDIRAC`, which is implemented using `DIRAC File Catalog` (`DFC`) and integrated with the `DIRAC` job management. It provides file catalog, metadata catalog and dataset catalog. The transfer system is the second sub-system implemented in the `DIRAC` framework. New services and agents can be easily deployed in the server due to `DIRAC`'s good scalability and flexibility.

### *2.3. The architecture of the Transfer System*

The main components of the transfer system are as follows:

- `Transfer Agent` is the scheduler to manage transfer workers.
- `Transfer Request Service` manages the transfer requests created by users.
- `Transfer DB` is the shared memory between the agent and service.
- `Dataset Service` is the dataset manager.
- `Accounting` keeps the transfer history. It follows the structure in `DIRAC` Accounting System [8]. It can be easily integrated with the web portal.
- `Web Portal` and `command line scripts` are two types of User Interface.

The workflow is described in the following:

(i) Administrator creates datasets in `DFC` first.
(ii) User creates a transfer request of a dataset using the command line or the web portal.
(iii) The `transfer request service` saves the transfer request in the `transfer DB`.
(iv) `Transfer agent` will transfer these files in the dataset.

## 3. Design and Implementation of the Transfer System

### *3.1. Transfer Request Service*

The transfer request service supports a basic management which includes:

- create a new transfer request with dataset name, source SE, destination SE and protocol.
- show the status of the specific transfer request.
- kill or re-transfer one transfer process.

It derives from class `RequestHandler` in `DISET` which implements `export_XYZ` methods. Then we use `RPCClient` in `DISET` to call these `XYZ` methods. The users use this `RPC` call mechanism to get useful information. The users can also use web portal to dispatch these `RPC` calls.

### *3.2. Transfer Agent*

The transfer agent implements a non-blocking scheduler managing the transfer workers. A transfer worker is an object whose status is saved in the database while it also has a process being executed with the low-level transfer tools. The python's standard library `subprocess` is used for communication between the worker and the real transfer process. The workflow of the transfer agent is shown in Figure 1 and described as follows.

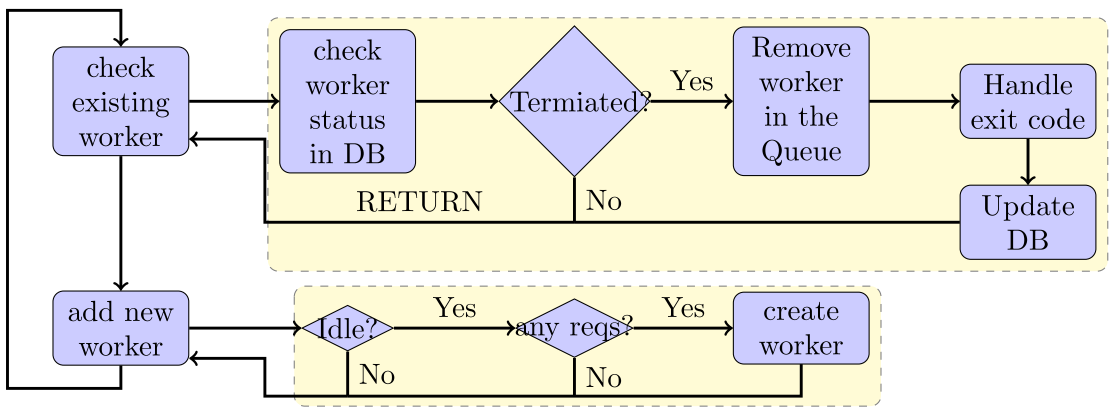


**Figure 1.** Workflow of the transfer agent

(i) The scheduler checks the existing worker. It can be divided into several steps:
   (a) Check the status of the worker in the database. For example, if user kills a transfer worker, the status in database will become "`kill`". The scheduler will send a signal to terminate the transfer process when "`kill`" is detected.
   (b) Check if the process of the worker is terminated or not.
   (c) Remove the worker from the queue.
   (d) Handle the exit code to check if the transfer is OK or not.
   (e) Update the status on the database and commit the entry in the accounting system.

(ii) The scheduler adds new worker if possible. If the system is idle and there are new requests, the scheduler will create a new worker.

For flexibility, a `TransferFactory` is used to create transfer workers with different protocols. An interface `ITransferWorker` is defined to be used by the transfer agent. The derived class should implement the interface and construct the command line which will be called when the worker begins to run. The derived class can also define how to handle the output of the process. In our views, building a high-level transfer system is simplified by this design.

## *3.3. Transfer Database*

In the transfer system, the transfer request service and the transfer agent share the data using the transfer database. In the database, we maintain two tables. One is for the transfer status of dataset and another is for the transfer status of worker. The schemas of these tables are defined in `BESDIRAC/TransferSystem/DB/TransferDB.sql` in the `BESDIRAC` project [7].

In `DIRAC`, a class called `TransferDB` derived from `DB` is used to manipulate the tables in transfer database. It can be used by transfer request service and transfer agent. A factory function called `namedtuple` for tuples with name fields in python's standard library is used to represent a record in tables. Several useful functions are also defined in `TransferDB` to simplify the operations.

## *3.4. Accounting*

Even though there is a type called `DataOperation` in `DIRAC` accounting system, a new type called `DataTransfer` is introduced to avoid ambiguousness between transfer system and other systems in `DIRAC`. The accounting system in `DIRAC` is flexible. Referring to Figure 2, what a data type is and how this data is shown are needed, therefore two classes derived from `BaseAccountingType` and `BaseReporter` are defined respectively. When a user sends a plotting request in the web portal, DIRAC will invoke the reporter automatically.

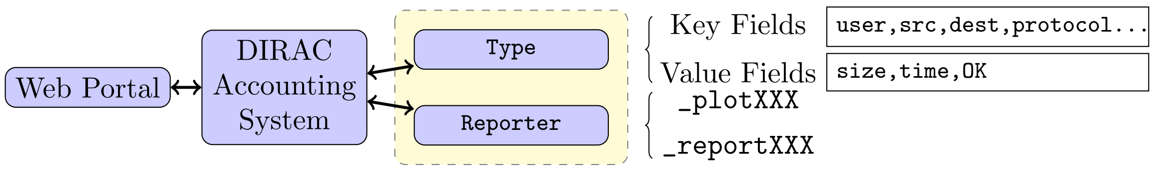


**Figure 2.** DIRAC Accounting System

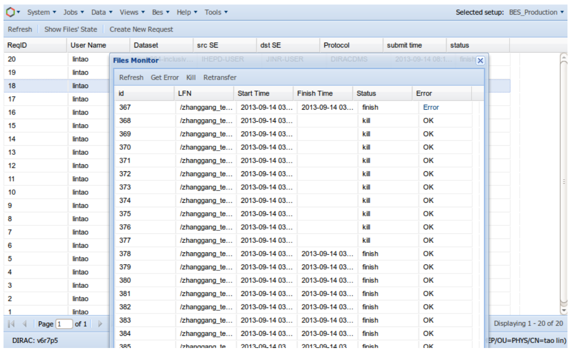
## *3.5. User Interface*

There are two types of User Interface: one is the command line scripts and another is the web portal. They are all simple wrappers for the RPC Calls which interact with the `BESDIRAC`. The web portal for `BESDIRAC` uses `Pylons` and `ExtJS2` in server side and client side respectively [9]. The `BESDIRAC` web portal supports transfer request management, dataset management and accounting of the transfer system. An example is shown in Figure 3.

In `BESDIRAC`, the web portal related code is organized into the directory called `Web`. There is a file called `web.cfg` which is used to load the menu entries into the menu bar in `DIRAC` web portal. The `DIRAC` web portal supports a uniform interface for the systems in both `DIRAC` and extensions. Even though the `DIRAC` project and `DIRACWeb` are divided into two different projects, there are relations between the two. Therefore, the `Web` module of `BESDIRAC` is distributed together with `BESDIRAC` to gain consistency.

## 4. Usage and limitation

We tested the transfer tool with 900 million $J/\psi$ event data which takes about 5TB of disk space. These data are divided into 9 datasets. It took 5 days to transfer these data from IHEP (Institute of High Energy Physics, Beijing, China) to JINR (Joint Institute for Nuclear Research, Dubna, Russia). The throughput is shown in Figure 4. The transfer speed is not high, about 13MB/s, due to the poor network connectivity. Currently, the consistency check does not take place in the transfer agent because it spends a lot of time to compute the Adler-32 checksum. In the future version, another agent will check the transfered files are the same between the remote sites.



| ReqID | User Name | Dataset | src SE | dst SE | Protocol | submit time | status |
|---|---|---|---|---|---|---|---|
| 20 | lintao |  | IHEPD-USER | JINR-USER | DIRACDMS | 2013-09-14 08:1... | finish |
| 19 | lintao |  |  |  |  |  |  |
| 18 | lintao |  |  |  |  |  |  |
| 17 | lintao |  |  |  |  |  |  |
| 16 | lintao |  |  |  |  |  |  |
| 15 | lintao |  |  |  |  |  |  |
| 14 | lintao |  |  |  |  |  |  |
| 13 | lintao |  |  |  |  |  |  |
| 12 | lintao |  |  |  |  |  |  |
| 11 | lintao |  |  |  |  |  |  |
| 10 | lintao |  |  |  |  |  |  |
| 9 | lintao |  |  |  |  |  |  |
| 8 | lintao |  |  |  |  |  |  |
| 7 | lintao |  |  |  |  |  |  |
| 6 | lintao |  |  |  |  |  |  |
| 5 | lintao |  |  |  |  |  |  |
| 4 | lintao |  |  |  |  |  |  |
| 3 | lintao |  |  |  |  |  |  |
| 2 | lintao |  |  |  |  |  |  |
| 1 | lintao |  |  |  |  |  |  |

| id | LFN | Start Time | Finish Time | Status | Error |
|---|---|---|---|---|---|
| 367 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | Error |
| 368 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 369 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 370 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 371 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 372 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 373 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 374 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 375 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 376 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 377 | /zhanggang_te... | 2013-09-14 03... |  | kill | OK |
| 378 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | OK |
| 379 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | OK |
| 380 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | OK |
| 381 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | OK |
| 382 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | OK |
| 383 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | OK |
| 384 | /zhanggang_te... | 2013-09-14 03... | 2013-09-14 03... | finish | OK |
| 385 | /zhanggang_te | 2013-09-14 03 | 2013-09-14 03 | finish | OK |

**Figure 3.** Transfer Request Management. It shows the status of the dataset and the files in dataset. It is in Bes→Transfer→Transfer Request in BESDIRAC web portal.

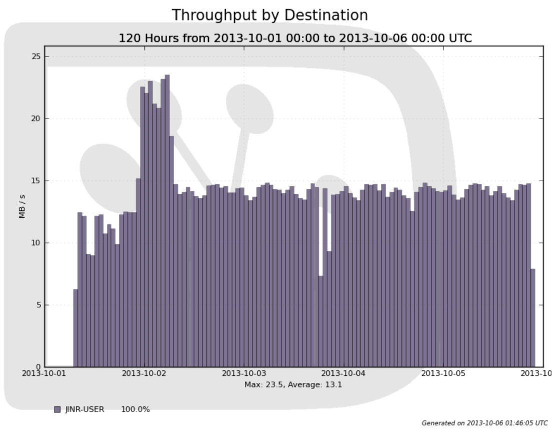


**Figure 4.** Throughput from Oct.1 to Oct.6, average throughput is 13MB/s. This figure is generated in DIRAC web portal automatically.

Now the main bottleneck of this system is the inefficiency because the transfer agent is deployed in one server and the maximum number of the workers is limited by the network bandwidth. But this system can be easily extended. One solution is to deploy multiple agents in different sites so that the agents can transfer these datasets parallelly. In order to keep up transfer status, those transfer agents only use one database. We will test this after several DIRAC instances are installed.

## 5. Conclusion and outlook

We have presented the design and implementation of BESDIRAC transfer system and had test with large datasets in this system. There is also a web portal for users to manage their own transfer requests and view the accounting more easily. We benefit from the flexibility of DIRAC.

The development of BESDIRAC transfer system is still ongoing. The kernel will be improved. The accounting will give more information. In the future, the integration of dataset management in DFC will be considered too.

## References

[1] Ablikim M *et al.* 2010 *Nucl. Inst. and Meth. in Phys. Res.* A **614** 345-99
[2] Ablikim M *et al.* 2013 Measurement of the integrated luminosities of the data taken by BESIII at $\sqrt{s} = 3.650$ and 3.773 GeV *Preprint* arXiv:1307.2022v1 [hep-ex]
[3] Tsaregorodtsev A *et al.* 2010 *J. Phys.: Conf. Ser.* **219** 062029
[4] DIRAC Project `http://github.com/DIRACGrid/DIRAC`
[5] Nicholson C *et al.* 2012 *J. Phys.: Conf. Ser.* **396** 032078
[6] Casajus A and Gracian R 2010 *J. Phys.: Conf. Ser.* **219** 042033
[7] BESDIRAC Project `http://github.com/mirguest/BESDIRAC`

[8] Casajus A *et al.* 2011 *J. Phys.: Conf. Ser.* **331** 072059
[9] Casajus A and Sapunov M 2010 *J. Phys.: Conf. Ser.* **219** 082004