EventDB: an event-based indexer and caching system for BESIII experiment

Yaodong Cheng[1] *, *Haibo* Li[1], Qi Xu[1, 2] , *Zhenjing* Cheng[1, 2], *Qiulan* Huang[1]

[1]IHEP computing center, 19B Yuquan Road, Beijing 100049, China
[2]University of Chinese Academy of Sciences, Beijing, China

Abstract. Beijing Spectrometer (BESIII) experiment has produced hundreds of billions of events. The traditional event-wise accessing of BESIII Offline Software System is not effective for the selective accessing with low rate during a physics analysis. In this paper, an event-based data management system (EventDB) is introduced, which can effectively alleviate the problems of low efficiency of data processing and low utilization of resources. Firstly, an indexing system based on NoSQL database is designed. By extracting specified attributes of events, the events interested to the physicists are selected and stored into the database, whilst the real data of event is still stored in ROOT files. For those hot events, the real event data can also be cached into EventDB to improve the access performance. The data analysis workflow of HEP experiments is needed to change if the EventDB system is applied. The analysis program queries the corresponding event index from database, then get event data from database if the event is cached, or get data from ROOT files if it is not cached. Finally, the test on more than one hundred billion physics events shows the query speed was greatly improved over traditional file-based data management systems.

1 Introduction

As the scale of high-energy physics (HEP) experiments continues to expand, more and more data is produced. For example, the BESIII experiment has accumulated about 10PB data. At present, most of high-energy physics experiment data are managed in the granularity of file, and each file contains several events. The event is basic data unit that contains data generated by a particle collision or a fundamental interaction between particles. Generally, it is difficult for one single organization to process all the data of one large high-energy physics experiment, and distributed computing technologies has been widely used in the high-energy physics field. The data are usually distributed to different sites based on file or data set. The file-based method greatly simplifies the complexity of data management and has been a traditional solution for a long time.

Meanwhile, file-based data management are facing a lot of challenges with the rapid growth of experiment data and the emergence of new technologies. For example, full data scanning in a physics analysis needs to read all of the related files, which causes possible I/O

* Corresponding author: chyd@ihep.ac.cn

bottleneck. The cache rate hit is also low if data is cached based on file. Too much unnecessary data is transferred if a physics analysis is executed in remote sites. Actually, physicists are only interested in just a few of events for a data analysis. For example, only 1/1000, sometimes even more less events are useful for a BESIII physics analysis [1]. If there is one kind of data management system which can globally select events interested to user according to some criteria, it will be beneficial to save I/O resources, improve CPU utilization and reduce data analysis time.

However, it is not a trivial matter to improve the efficiency of existing file-based systems because of some challenges, for example:

1) It is difficult to select events efficiently in file-based storage system. A data analysis has to scan all the related files in order to get a few of interested events. Thus, a lot of I/O operations are wasted.

2) If one site does not have sufficient storage space and enough network bandwidth, it is difficult to run data analysis tasks which need a large of amount of input data. In this case, it required that only a subset of data are transferred on demand.

3) The existing data analysis workflow is extremely complicated, and it is difficult to keep them unchanged to adopt a completely new data management system. In the field of high-energy physics, most of experiments develop their own application software systems. The new-developed data management system should not have a great impact on these applications.

To effectively solve the above problems, we designed and implemented the EventDB system, an event indexer and caching system for the BESIII experiment. This paper is structured as follows. Section 2 summarizes the related works. Section 3 introduces the design and implementation of the EventDB system including the architecture, metadata management, cache service and data transfer. In Section 4 the test results using BESIII data are explained. Finally Section 5 discusses the conclusions and future work.

2 Related works

Recently some systems are developed to adopt index technology in order to select events efficiently. In these systems the event index and real event data are stored separately. The event index are kept in special files or database while the real event data are still stored in ROOT files. ROOT is a de facto framework for data storage and processing in high-energy physics. For example, Liu Beijiang from IHEP (the Institute of High Energy Physics, Chinese Academy of Sciences) extracted a series of attributes (called TAGs) from events and stored them in separate ROOT files. Event TAGs are event-level metadata which support efficient identification and selection of events of interest to a given analysis. The method was used in the BESIII experiment [1]. Bloomfield et.al. from the University of Melbourne stored the location information of the specific screening conditions in ROOT files and applied it to the Belle-II experiment in Japan [2]. In this way, the user directly eliminates the pre-screening process when doing the analysis. However, it is difficult to support global sharing and query in these file-based index model.

With the increase of the number of events, many researchers have proposed to use NoSQL databases [3] to store the index information in recent years. The ATLAS Collaboration adopted HBase to build event index database EventIndex [4]. Additionally, the ATLAS collaboration used Oracle relational database to build EventIndex Oracle-based system (EIO) [5] for providing event-wise services. The ATLAS EventIndex uses Hadoop technologies for efficiently storing all indexed data, and a subset is copied into Oracle for accessing. Each of the both systems has advantages and disadvantages depending on the use case. For example, the main use cases of EIO are “Event Lookup”, “Event Overlap” and so on.

Sun Gongxing and others from IHEP proposed to store event data in HBase to speed up the data analysis process for the BESIII experiment [6]. It will change the BESIII data storage and analysis model greatly. For example, all the BESIII event data must be stored into HBase instead of distributed file system, and a set of JNI-based interfaces has to be developed to support data access across C++ and JAVA languages. As a result, it is difficult to support future large-scale data storage and analysis. The EventDB system described in this paper focuses on the event index and pre-selection, and does not change the BESIII storage and analysis model.

3 The design and implementation

3.1 Design conceptions

The EventDB system aims at allowing fast and efficient selection of events of interest to an analysis. Trillions of event indexes are built to point to those events in millions of files. The EventDB is implemented based on a NoSQL database, and follows some design conceptions:

1) Event-level metadata are extracted and kept in NoSQL database while the real event data in ROOT files. The original workflow of an analysis is unchanged after the interested events are selected from the EventDB system.

2) Providing faster event selection and better data access performance through the EventDB data management system.

3) Easy to be compatible with existing physics analysis applications. The existing physics analysis applications for the BESIII experiment like BES Offline Software System (BOSS) [7] or SNiPER [8] should use the system to access local and remote data without much modification.

3.2 Architecture

The architecture of EventDB is depicted in Figure 1. There are three main components including EventIndexer, EventAccess and EventExtractor. EventIndexer is an event index database. EventAccess includes two subcomponents: the Event-oriented Data Transfer System and the Event-oriented Caching System. The EventExtractor is an event TAG extractor, which scans all of event files and extracts specified event attributes into NoSQL databases.

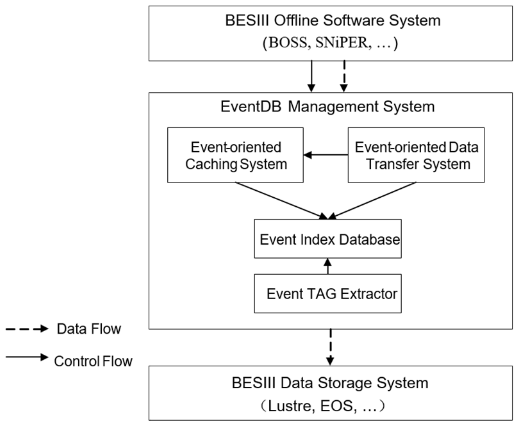
BESIII Offline Software System
(BOSS, SNiPER, …)
EventDB Management System
Event-oriented Caching System
Event-oriented Data Transfer System
Event Index Database
Event TAG Extractor
Data Flow
Control Flow
BESIII Data Storage System
(Lustre, EOS, …)

Fig. 1. The architecture of EventDB composed of EventIndexer, EventAccess and EventExtractor

3.2 Implementation

3.2.1 Event-level metadata system

Event-level metadata system is intended to query and select events of interest to an analysis. It includes two submodules: the EventExtractor and EventIndexer.

The EventExtractor is responsible for scanning the event files and extracting event attributes meaningful for a data analysis. Take an example of the BESIII experiment, 22 attributes including run number, event number, and total track number and so on are extracted. Then the EventExtractor stores these attributes into underlying databases.

The EventIndexer adopts an HBase [9] cluster to build a large-scale event index. HBase uses a lexicographically ordered index structure to build a primary key and cache it in memory, which provides good query efficiency. Inverted index technology is introduced here, and event attribute name and value are designed as Rowkey so that a binary search can be performed on the primary key. The events that satisfies the same query condition are merged into one record of HBase and returned as a query result. The event attribute index in HBase is constructed as shown below.

Table 1. Event Index examples in HBase.

Rowkey	EventIndex
-8026#Neutral#003	pips1.dst-0
-8026#Neutral#004	pips1.dst-0, pips1.dst-5, pips2.dst-8
-8026#Neutral#005	pips1.dst-0, pips1.dst-5, pips2.dst-9
-8026#Neutral#006	pips1.dst-0, pips1.dst-5, pips2.dst-10,…
-8026#Neutral#007	pips1.dst-0, pips1.dst-5
-8026#Neutral#008	pips1.dst-0, pips1.dst-5, pips2.dst-12

Figure 2 shows the storage capacity occupied by the EventIndexer. For comparison there were nearly 100 million BESIII events with an original data size of 2.2TB in the testing system, while the event index contains less than 10 million records with the size of 55GB.

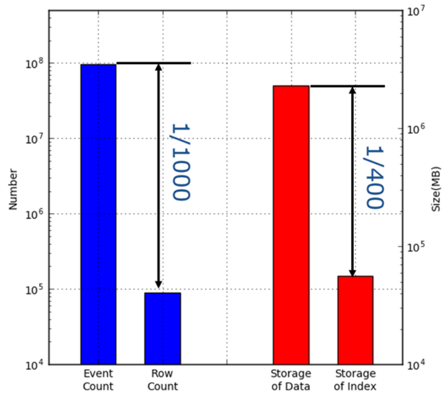

Fig. 2. The comparison of storage capacity occupied by the EventIndexer and real data.

The workflow for creating indexes is shown in Figure 3. The EventDB mainly stores the basic event attributes, such as RunID, EventID, FileID, VersionID, BeamEnergy, Ntracks and Nshowers, which are chosen by physicists. The EventDB builds a multidimensional index on these attributes to improve query performance. The FileDB is a file catalogue which allows users to access their files with a logical filename without knowing their physical location, like the LFC (LCG File Catalogue) [10] which is used in WLCG (Worldwide LHC Computing Grid). The information in the database is generated by the EventExtractor which scans the event files managed by the distributed file system.

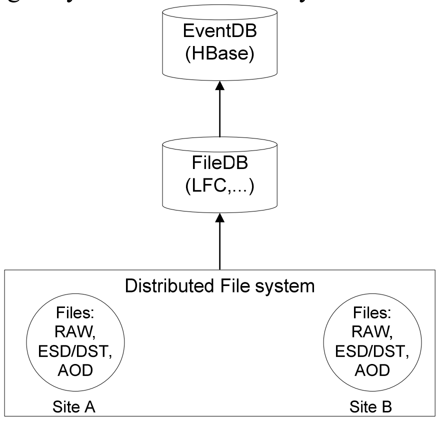

Fig. 3. The workflow of creation of indexes.

3.2.2 Event cache service

The event cache service is one of subcomponents of the EventAccess, which is responsible for caching hot event data on SSD or memory. The data is cached in the granularity of the event instead of the whole file. When an event is accessed by an application, the service will serialize its data into an entry value and store it in a key-value cache system. Then if the event is accessed again, the service will get the entry value from the cache, de-serialize it into ROOT object, and return it into the application directly. It is required to assemble the event cache module into BESIII offline software system such as BOSS or SNiPER.

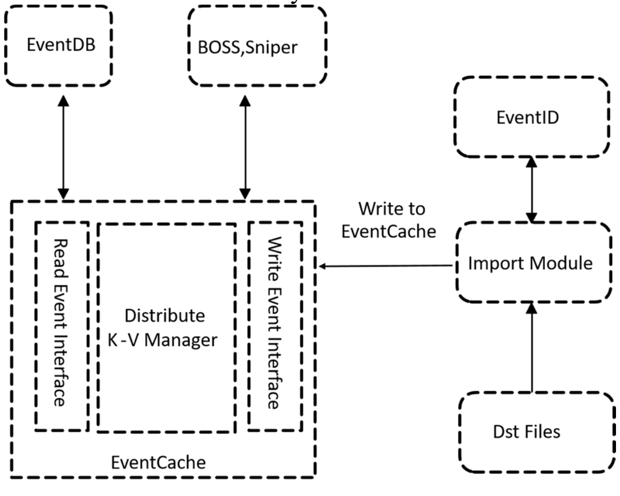

Fig. 4. The Architecture of the event cache service.

3.2.3 Event-oriented data transfer service

It is not necessary to transfer the whole input file if an analysis task is running remotely in another site because the EventDB system provides event-level query, selection and caching functionalities. So the event-oriented data transfer service, called the LEAF system [11] was implemented. It is also one of subcompacts of the EventAccess employing the XRootD framework [12] to implement data transfer server, data transfer client and even use XrootdFS (a Posix Filesytem for XrootD) to mount remote data as a local file system.

The amount of data is greatly reduced because the LEAF system only transfer the events interest to an analysis task. The LEAF system consists of three parts: the data transfer server, the data transfer client with local cache, and xrootd plugin. User does not need to know where the event data is located. The architecture of the LEAF system is shown in figure 5.

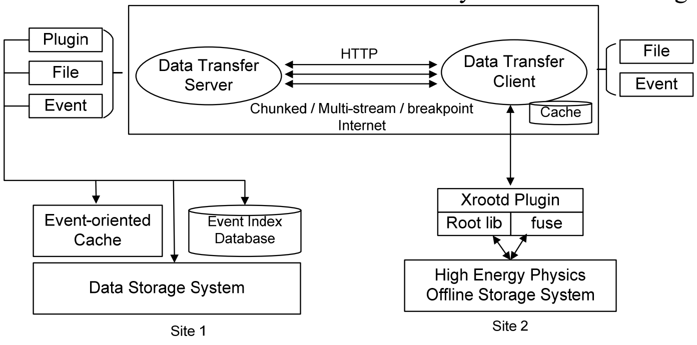

Fig. 5. The architecture of event-oriented data transfer service.

3.2.4 Use cases

The EventDB system is designed to process HEP data in the granularity of event. The workflow of event-based data processing is a little different from that of file-based data processing, which is depicted in figure 6.

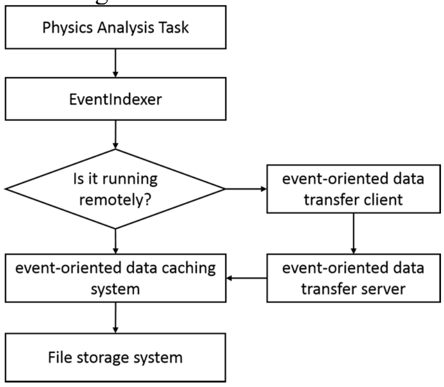
Physics Analysis Task
EventIndexer
Is it running remotely?
event-oriented data transfer client
event-oriented data caching system
event-oriented data transfer server
File storage system

Fig. 6. The workflow of event-based data processing.

1) An analysis task firstly specifies some criteria to get event list via the EventIndexer interface usually generating a JSON index file, for example:
EvtQuery -v 700 -r -8107 -q "range(NTracks,4,5)" -f 700.json
2) Check if the task is running remotely.
3) If the task is running at a remote site, it will get the real event data via the event-oriented data transfer service with the JSON file generated by step 1. The data transfer service calls the event-oriented caching service.
4) If the task is running at local site, it calls event-oriented caching service directly.

5) If the event is not in the cache, the event-oriented caching system will read the event data from underlying file storage system, such as Lustre.

4 Test bed and results

We have setup a test bed composed of two sites including Beijing and Chengdu to evaluate the performance of the EventDB system. The distance between the two sites is about 2000 KM. The network latency is about 35 ms, and the bandwidth is 1 Gpbs. In the test, about one hundred billion events were imported and indexed in the EventDB, and the analysis task for evaluation selected 11237 events from 99130 events in files. As shown in the table below, we compared the performance of the EventDB system in four cases, depending on site location, whether to use the EventIndexer, the event cache or not.

Table 1. Test cases.

Case	Site	Event Indexer	Event Cache
1:Original analysis (file-based)	local	No	No
2:Original analysis + EventIndexer	local	Yes	No
3:Original analysis+Event Cache+ EventIndexer	local	Yes	Yes
4: Original analysis + Remote + Event Cache + EventIndexer	remote	Yes	Yes

The test results showed that the event retrieval time was 9.3 minutes with event indexer (case 2), which saves about 30% of the time compared with the original file-based analysis (case 1). If the event cache was enabled (case 3), the retrieval time was reduced to 3 seconds, and the speed was increased by nearly 78% compared with case 1. If the analysis task was running at a remote site with event indexer and event cache enabled (case 4), the retrieval time was 11.68 minutes, which was a little better than the original file-based analysis (case 1).

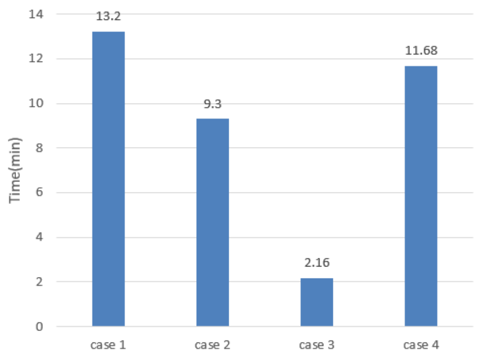

Fig. 6. The performance evaluation of EventDB.

5 Summary

Future challenges for high-energy physics in the domain of data management are not simply an increase of data volume. The significantly increased computational requirements will also place new requirements on data access. In particular, the increasing availability of very high-speed networks may reduce the need for CPU and data co-location. Such networks may allow for more extensive use of data access over the wide-area network (WAN). The use of event-based data streaming along with more traditional dataset-based or file-based data access, will be particularly important for optimizing the utilization of opportunistic computing cycles on HPC facilities, commercial cloud resources, and campus clusters. The EventDB is a system supported and developed by 'Big Scientific Data Management Systems' project in China. It has been applied and evaluated in the BESIII and HXMT (Hard X-ray Modulation Telescope) experiments. The test on more than ten billion physics events showed that the query speed was greatly improved than traditional file-base data management systems. Currently, we have setup a Hadoop cluster with 10 servers, and about 110 billion BESIII events and 400 billion HXMT events were indexed in the EventDB system to support a few of use cases such as BESIII event lookup, HXMT event display and so on.

Acknowledgements

This work was supported by the National key Research Program of China “Scientific Big Data Management System”（No.2016YFB1000604）and National Natural Science Foundation of China (No. 11675201 and 11575223).

References

1. Liu B J, *High performance computing activities in hadron spectroscopy at BESIII*, **J.phys.Conf.ser***,* 523(1), 012008 (2014). DOI: 10.1088/1742-6596/523/1/012008.
2. Martin S, et.al., *Index files for Belle II-very small skim containers.* **J.phys.Conf.ser**, 898(9), 092031 (2017). DOI : 10.1088/1742-6596/898/9/092031.
3. Han J, et.al. , *Survey on NoSQL database*, **6th International Conference on Pervasive Computing and Applications** (2011). DOI: 10.1109/ICPCA.2011.6106531.
4. Sánchez J, et.al. *Distributed Data Collection for the ATLAS EventIndex*, ***J.phys.Conf.ser***, 664(4), 042046 (2015). DOI : 10.1088/1742-6596/664/4/042046.
5. Gallas E J, et.al, *An Oracle-based event index for ATLAS*, **J.phys.Conf.ser,** 898(4), 042033 (2017). DOI : 10.1088/1742-6596/898/4/042033.
6. Lei X F, et.al., *BESIII Physics Data Storing and Processing on HBase and MapReduce*, **J.phys.Conf.ser,** 664(7), 072032(2015). DOI : 10.1088/1742-6596/664/7/072032.
7. Li W D, et.al., Chapter 2 *The BES-III Detector and Offline Software*, **International Journal of Modern Physics A**, 24, (2009). DOI : 10.1142/S0217751X09046424.
8. Xia X, et.al. *Application of SNiPER framework to BESIII physics analysis*, **Chinese Physics C,** 41(5), 056202 (2017). DOI : 10.1088/1674-1137/41/5/056202.
9. Vora M N, *Hadoop-HBase for Large-Scale Data*, **IEEE International Conference on Computer Science and Network Technology**, pp. 601-605, 2011. DOI : 10.1109/ICCSNT.2011.6182030.
10. Munro C, Koblitz B, Santos N, et.al. *Performance comparison of the LCG2 and gLite file catalogues,* **IEEE Nuclear Science Symposium Conference Record**, vol. 2, pp. 847-851, 2005. DOI : 10.1109/NSSMIC.2005.1596388.
11. Cheng Y D, et.al., *LEAF: A data cache and access system across remote sites*, **J.phys.Conf.ser,** 1085(3), 032008, 2018. DOI : 10.1088/1742-6596/1085/3/032008.
12. Dorigo A, et.al., *XROOTD - A highly scalable architecture for data access*, **WSEAS Transactions on Computers** 4(4):348-353, 2005.