BESIII Physical Analysis on Hadoop Platform

Jing HUO¹, Dongsong ZANG¹², Xiaofeng LEI¹², Qiang Li¹², Gongxing SUN¹
¹Institute of High Energy Physics, Beijing, China
²University of Chinese Academy of Sciences, Beijing, China
E-mail: huojing@ihep.ac.cn

Abstract. In the past 20 years, computing cluster has been widely used for High Energy Physics data processing. The jobs running on the traditional cluster with a Data-to-Computing structure, have to read large volumes of data via the network to the computing nodes for analysis, thereby making the I/O latency become a bottleneck of the whole system. The new distributed computing technology based on the MapReduce programming model has many advantages, such as high concurrency, high scalability and high fault tolerance, and it can benefit us in dealing with Big Data. This paper brings the idea of using MapReduce model to do BESIII physical analysis, and presents a new data analysis system structure based on Hadoop platform, which not only greatly improve the efficiency of data analysis, but also reduces the cost of system building. Moreover, this paper establishes an event pre-selection system based on the event level metadata(TAGs) database to optimize the data analyzing procedure.

1. Introduction

1.1 The current BESIII computing architecture
High Energy Physics experiment is a typical data-intensive application. The BESIII computing system now consist of 3PB+ data, 6500 CPU cores, and it is estimated that there will be more than 10PB data produced in the future 5 years. The current BESIII cluster is a traditional Data-to-Computing structure shown as Figure 1, its data storage is separated from computation. Therefore, huge volume of data will be transferred from the storage system to computing nodes via the network during the data analysis procedure.

Figure 1. The current architecture of BESIII cluster
As the traditional PC Farm system, the BESIII cluster faces three major problems. Firstly, the CPU utilization is low because of the high I/O waiting time, which is caused by huge volume of data transfer. Secondly, the probability of the file system failure increases with the expansion of Lustre’s scale (the number of disks). The statistics shows that IHEP had 300 occurrences of disk errors in the whole year 2012. Finally, the Lustre file system needs high performance network equipments and special storage devices, and it is a quite expensive solution.

1.2 The new technology
In recent years, the Internet companies brought forward some new technologies to store and process Big Data. Google has become leader in Big Data processing. E.g. it has published three famous papers: The Google file system [1], MapReduce: simplified data processing on large clusters [2], and Bigtable: A Distributed Storage System for Structured Data [3]. Hadoop [4] is an open source project based on Google’s papers, which is developed by Yahoo! and Apache. It includes three main parts: a distributed file system named HDFS [5], a distributed programming model and job execution framework named MapReduce, and a distributed database named Hbase [6].

Supported and widely used by many companies, Hadoop has become a de-facto standard for Big Data processing in enterprises. Recently, Hadoop has also captured the attention of more and more scientists in many areas, such as bioinformatics, astronomical image processing. And it is also partly used in HEP: for instance, there are 7 CMS sites in USA using HDFS as the storage element [7], INFN (Italy) uses Hadoop and MapReduce to process HEP data [8], etc.

However, the standard Hadoop platform is not completely suitable for HEP physical analysis. The reason is that Hadoop is designed to process text data, while HEP data is stored as objects with a tree structure in DST file, and the <key, value> pairs based data processing is not suitable for HEP ROOT [9] processing too. These differences determine that we have to redesign the MapReduce procedure for HEP physical analysis, and develop new libraries to support HEP physical analysis I/O.

2. The New Computing-to-Data Structure
Based on the analysis of Hadoop platform and the features of HEP data processing, we design and develop a new data processing platform for HEP, the architecture is shown in Figure 2.

This new architecture can solve the problems we face in the current BESIII computing system. Firstly, HDFS uses the local disks of the computing nodes to store data. Jobs will be scheduled on nodes where the data stays by MapReduce framework. This means most of the jobs can read data from local disks, so it can significantly reduces the pressure on network. Due to that, the CPU utilization can be highly improved. Secondly, HDFS uses data replication to provide fault tolerance and high availability. Every file and its replications are distributed in different nodes, even in different racks. So, as long as the number of failure nodes is less than the number of replications, the system can provide service normally. Finally, Hadoop doesn’t need high performance network equipments and special
storage devices. Assume BESIII computing system comprises 1000 computing nodes and each of them provides 4TB*4(disks) storage capacity, HDFS can provides 16PB storage capacity, and this only costs 1/3 of Lustre.

3. Implementation

3.1 System architecture

The architecture of new BESIII computing system is shown in Figure 3.

![Figure 3. System architecture](image)

The System Service Layer provides file storage, job management, data I/O libraries and database service. The Hadoop MapReduce framework provides resource management and job scheduling. The HEP data I/O libraries and Application framework layer provide libraries and frameworks that the application system layer depends on. For example, the RootRecordReader and RootRecordWrite depend on ROOT framework, and the CLHEP [10] provides the basic function libraries which are frequently used in HEP. The Thrift interface and TAG information management are in charge for the generation of TAG, and help to do the filtration in parallel. In application system layer, we provide two major services: Event Analysis System and Event Pre-Select System. The AFS [11] system provides user authentication, ID mapping, and access control. It also provides user space for users to set different environment variables for their own jobs. In addition, we use Puppet [12] for cluster management, and use Ganglia [13] for monitoring. This architecture also provides CLI and Web interfaces to users.

3.2 Data access

According to the features of HEP data processing, we store each file in one block in HDFS, and develop a data access interface(HEP data I/O libraries) for the C++ framework ROOT [14]. According to that, the execution performance has been improved by reading local data.

As shown in Figure 4, we develop two classes to provide file access and directory operation. THDFSFile inherits from TFile, which is the interface to access DST file in ROOT. THDFSFile provides functions to access and store objects, and it is implemented based on libhdfs, which can make C++ program access HDFS through JNI. THDFSSystem inherits from TSystem, it provides directory operations and access rights management.
3.3 MapReduce procedure for HEP data analysis

We separate the task execution from the MapReduce job scheduling framework, and move them to the C++ side. We develop the corresponding function libs for data accessing and intermediate data processing, which makes the C++ programs run efficiently under the MapReduce framework. We also improve the “reduce” procedure according to the characteristics of HEP data analysis by simplifying the shuffle and sort phases of the intermediate data.

We implement this system referring to the HCE [15] mechanism. Described as Figure 5, in the Map part, C++ process receives the filename from the JVM, calls the ROOT framework and user program to process DST file according to filename, and reports the information of task process and counter. Then the results of Map tasks will be serialized and stored as IFile(＜key-len, key, value-len, value＞) in the local disks. In the Reduce part, java JVM obtains all the outputs from the Map tasks, and stores these files as IFiles in local disk after sorting and combination. Then the C++ program will deserialize the IFiles and transfer them to Reduce program to generate the final results.

4. Events Pre-selection

We also optimize the physics events analysis procedure by establishing an event level metadata (TAGs) store based on Hbase, and an events pre-selection system based on TAGs, which reduces the number of events that need to be further analyzed by the users by 2-3 orders of magnitude.

4.1 Events filtration

The TAG data contains some simple but important attributes of an event, and the pointer to event (event ID) in RAW/DST file. The relationship between TAG, RAW, and DST are shown in Figure 6. The size of the TAG is only 1% of event data, so it wouldn’t increase so much storage space.
The TAG data is stored in Hbase, Figure 7 shows its generation model, and Figure 8 is an example of TAG table. In this model, we use a C++ program based on ROOT as the Mapper to processes the DST file to generate the TAG data, and write it into Hbase.

![Figure 7. The generation model of TAG](image1)

![Figure 8. The TAG table](image2)

The events filtering model based on TAG is shown in Figure 9. The TAG table in HBase is split into multiple Map tasks to do the filtering in parallel. The event IDs, which match the user’s requirement will be selected and written into another table in HBase.

![Figure 9. The events filter model](image3)

### 4.2 HEP physical analysis with events pre-selection

The analysis job will be split into multiple Map tasks according to locations of the DST files and the number of the events after filtering. As described in Figure 10, the Map task reads the complete information from DST file selectively according to the Event ID, and calls the user program to analyze it. The results of the Map tasks will be transmitted to Reduce tasks to be merged into the final results.

![Figure 10. The events analysis job model](image4)

### 5. Evaluation
The system is evaluated by analyzing the real data from BESIII experiments. The testing program is one program of the BESIII offline software called Rhopi events analysis. The testing environment is set up with 8 nodes, every node with 8 cores CPU of 2.4GHz, 24GB memory and 1000M Ethernet card.

Figure 11 shows the test result of CPU usage of the new architecture and the current system. Compared with the current system, the I/O waiting time of Hadoop is decreased from 10% to 2%, and CPU usage is increased from 80% to 90%.

![Figure 11. CPU utilization](image)

Figure 12 shows the test results of execution time. After filtering with TAG, the number of events is reduced to only 5% of the total number; and the execution time is decreased to 17%. We can further improve the execution time by reorganizing DST file according to the selective reading pattern caused by pre-selection. After reducing the size of Basket from 30M to 500K, the execution time is reduced to 2.3% of the original model.

![Figure 12. Test results of execution time](image)

![Figure 13. Test result of parallel efficiency](image)

In order to test the parallel efficiency of Hadoop, we analyzed 39,361,594 events (159GB) under different number of work nodes (each node launched 8 processes). The result is shown in Figure 13, the execution time decreases with the number of nodes added.

6. Conclusion

The MapReduce model is getting more and more popular in both Internet and scientific research fields. But due to the complexity of High Energy Physics data processing, we encounter some difficulties in
the application of MapReduce. Fortunately after studying the features of HEP data processing in detail, we give a feasible solution to these problems in this paper. And the evaluation shows that the filtering can improve the efficiency of analysis, and the procedure of analysis can be done highly in parallel.

Future work involves improving the format of events storage, random write in HDFS, and the execution of reconstruction and simulation jobs on Hadoop.

7. Acknowledgment
This work was supported by the National Natural Science Foundation of China (NSFC) under Contracts Nos. 11375223, and 61161140454.

References
[1] Ghemawat S, Gobioff H, Leung S. The Google file system [C]//Proc. of the nineteenth ACM symposium on Operating systems principles. New York, NY, USA: ACM Press, 2003.
[2] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]//Proc. of the 6th conference on Symposium on Operating Systems Design & Implementation. San Francisco, CA, USA: ACM Press, 2004.
[3] Chang F, Dean J, Ghemawat S, et al. Bigtable: A Distributed Storage System for Structured Data[J]. ACM Transactions on Computer Systems (TOCS), Volume 26 Issue 2, June 2008
[4] Apache Hadoop. [EB/OL]. (2005). http://hadoop.apache.org.
[5] Apache Hadoop, HDFS Architecture Guide.[EB/OL].(2012). http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html
[6] Apache Hbase.[EB/OL].(2012). http://hbase.apache.org/
[7] Bradley D, Dasu S, Maier W, et al. A highly distributed, petascale migration from dCache to HDFS[C]//Proc. of HEPIX Fall 2011. Vancouver, BC, CA: 2011
[8] Riahi H, Donvito G, Fanò L. Using Hadoop File System and MapReduce in a small/medium Grid site[J]. Journal of Physics: Conference Series, Volume 396 Part 4, 2012
[9] The ROOT team. ROOT [EB/OL]. (2010). http://root.cern.ch.
[10] CLHEP - A Class Library for High Energy Physics.[EB/OL].(2011). http://proj-clhep.web.cern.ch/proj-clhep/
[11] OpenAFS.[EB/OL].(2012). http://www.openafs.org/
[12] Puppet Labs. What is puppet?[EB/OL]. (2013-02-15). https://puppetlabs.com/puppet/what-is-puppet/
[13] Ganglia Monitoring System. [EB/OL]. (2013-02-15). http://ganglia.info/
[14] Antcheva I, et al. ROOT - A C++ framework for petabyte data storage, statistical analysis and visualization[J]. Computer Physics Communications 180, 2499-2512, 2009
[15] Apache Hadoop, Hadoop C++ Extention [EB/OL]. (2013-02-15). https://issues.apache.org/jira/browse/MAPREDUCE-1270