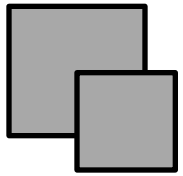


BESIII Physical Analysis on Hadoop Platform

CHEP2013

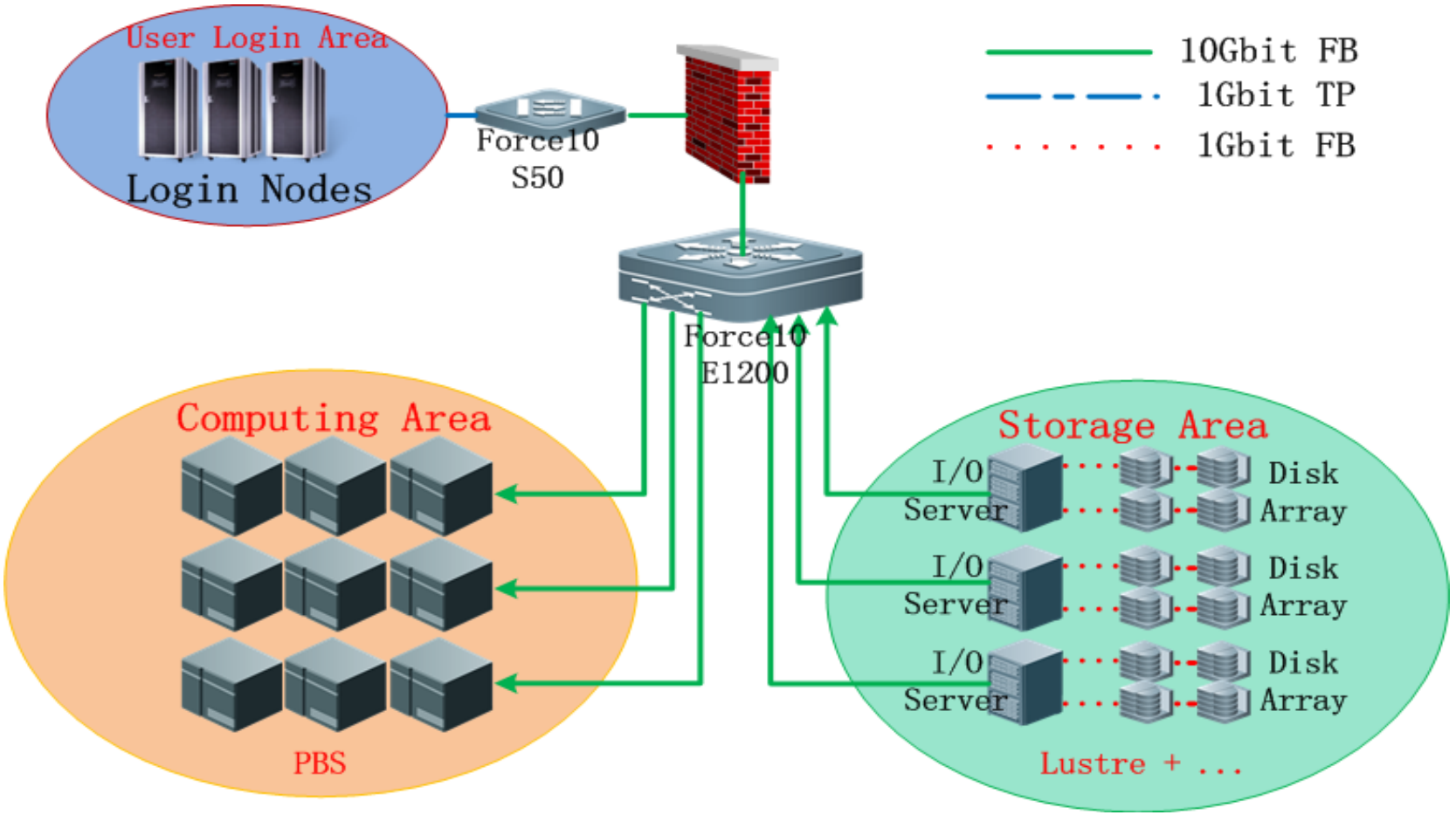
October 14-18, 2013 in Amsterdam,NL

Sun Gongxing/IHEP-CC

Outline

- The current data processing system
- Problems & Challenges
- The new data processing system on Hadoop with MapReduce framework
- Pre-selection of event & I/O optimization
- Summary
- Future work

The Architecture of BESIII Data Processing System



>5000CPU Cores

~3PB

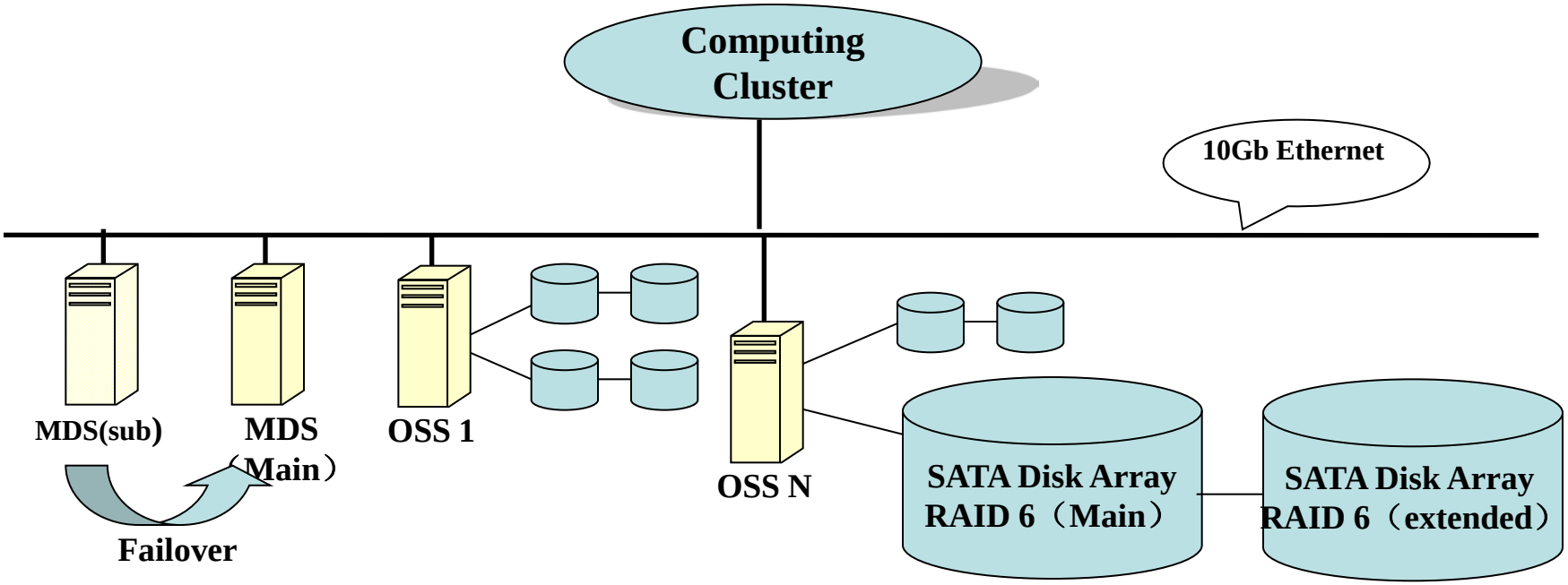
The Current BESIII Storage System

Version:***1.8.6***

I/O servers: ***42***

Storage capacity: ***3PB+***

quite expensive solution with hard maintenance!



Problems & Challenges

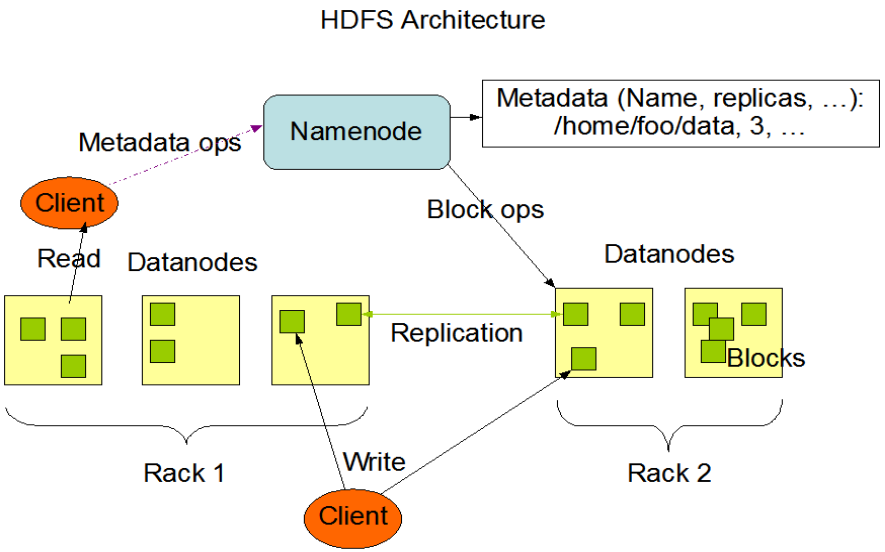
- More data taking in the future
  - 10PB of data will be produced in 5 years
- Network I/O becomes the bottleneck for data-intensive jobs
- More money should be invested to buy better network equipments and storage devices
- New techniques should be explored

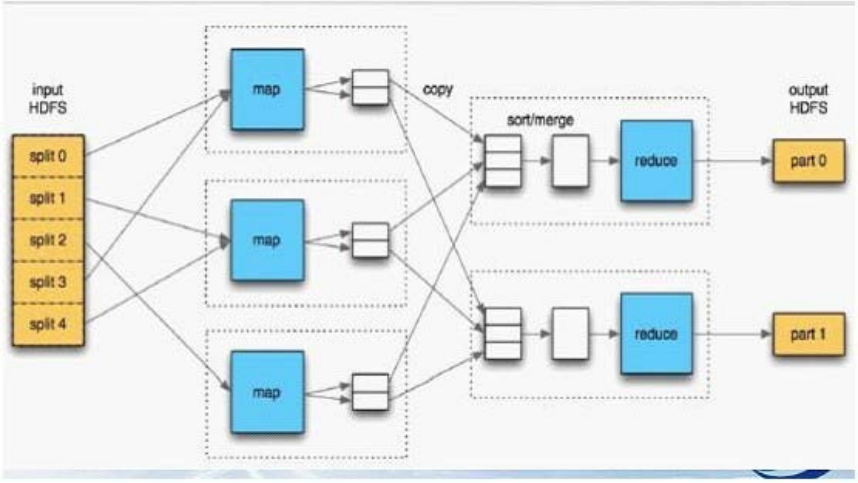
Developments in Computer Technology

- PC becomes more powerful
  - —16 cores CPU
  - —4TB disk * 4
  - —100+ GB memory
- Key-Value data structure
  - —High concurrency
  - —High scalability
  - —High speed

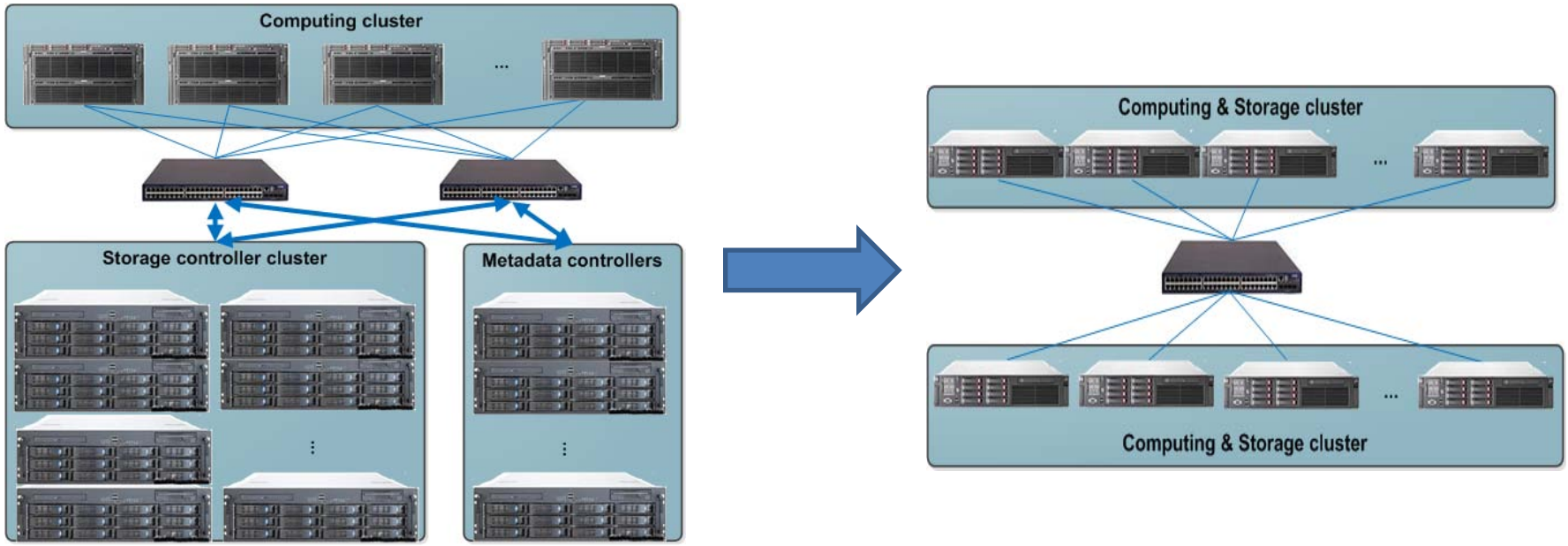
Hadoop & MapReduce & Hbase

- Hadoop
  - An open source framework based on Google's paper: Google File System(GFS)、MapReduce and BigTable
  - Include HDFS & MapReduce & Hbase
  - Widely used in the production environment
- MapReduce
  - A distributed programing model by Google
  - A job execution framework
- Hbase
  - A distributed database just like BigTable



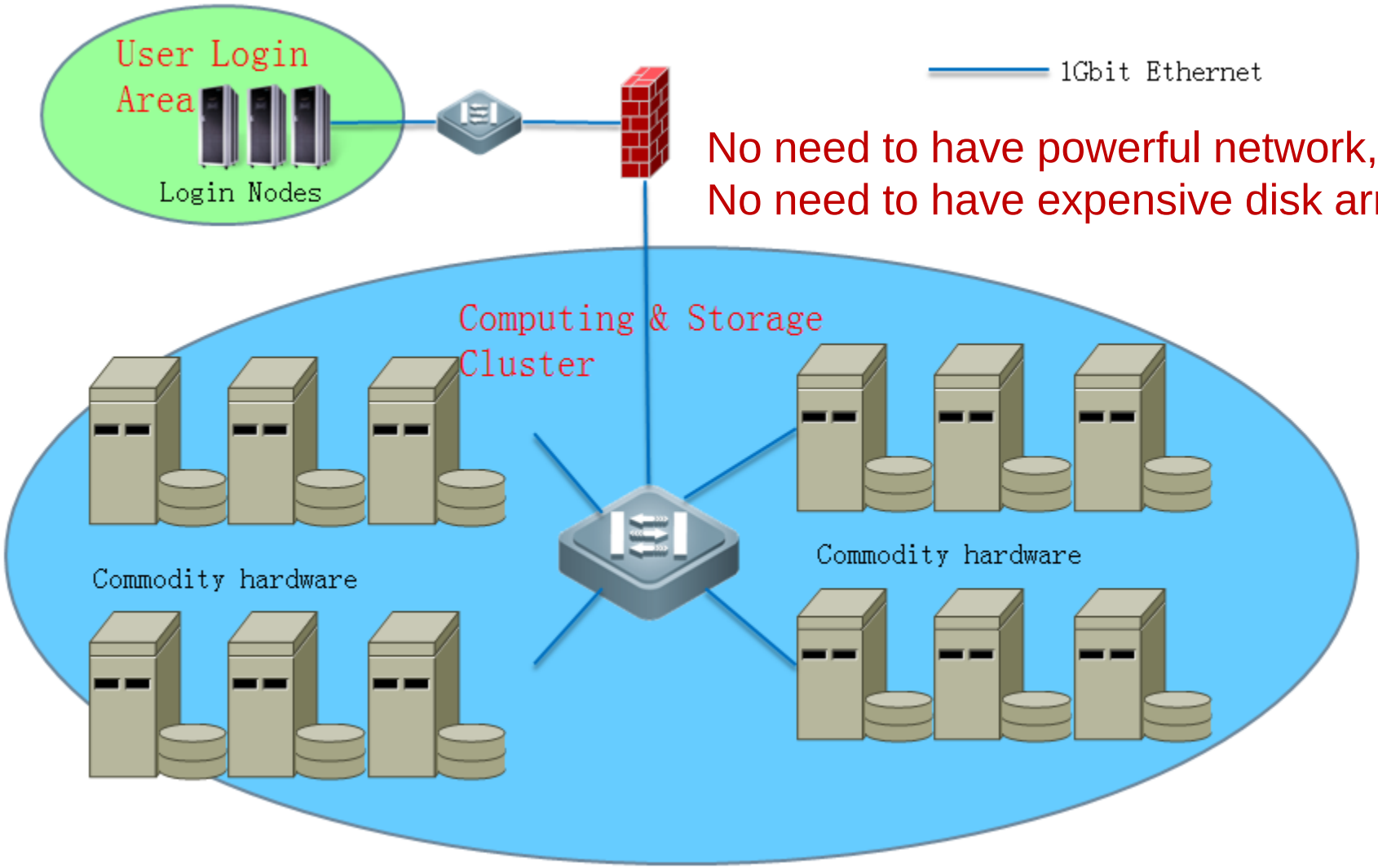


Change in Computing Model



Moving data to computation Moving computation to data

The New Architecture



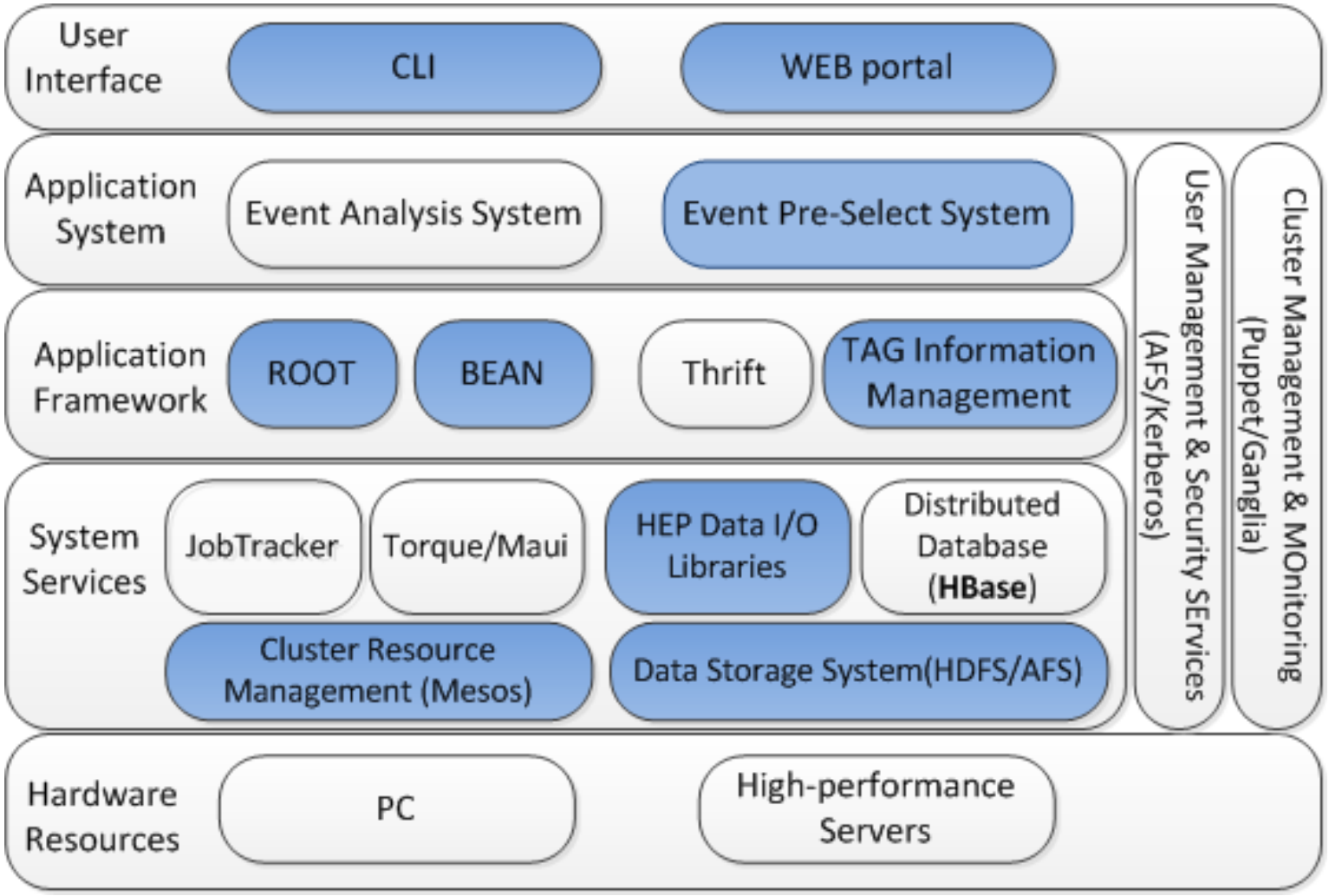
The Benefits of The New Technology

- Much cheaper
  - Use local disks of computing nodes for storage
  - Nearly no network I/O, no need for powerful network
- High cpu utilization with low I/O waiting
  - Schedule jobs to nodes where the data is
- High availability & fault-tolerance
  - Use software to deal with hardware failures
- Very easy to scale out & up

Work to Be Done for HEP Map-reduce Job

- HEP data analysis I/O
  - Access files in HDFS using C++ codes
  - Random write in HDFS
- HEP data MapReduce procedure
  - Use C++ codes instead of java codes in MapReduce
  - Simple reduce(merge) procedure for HEP
- Libraries and UI to make user programming easy

BESIII Computing Framework Based on Hadoop

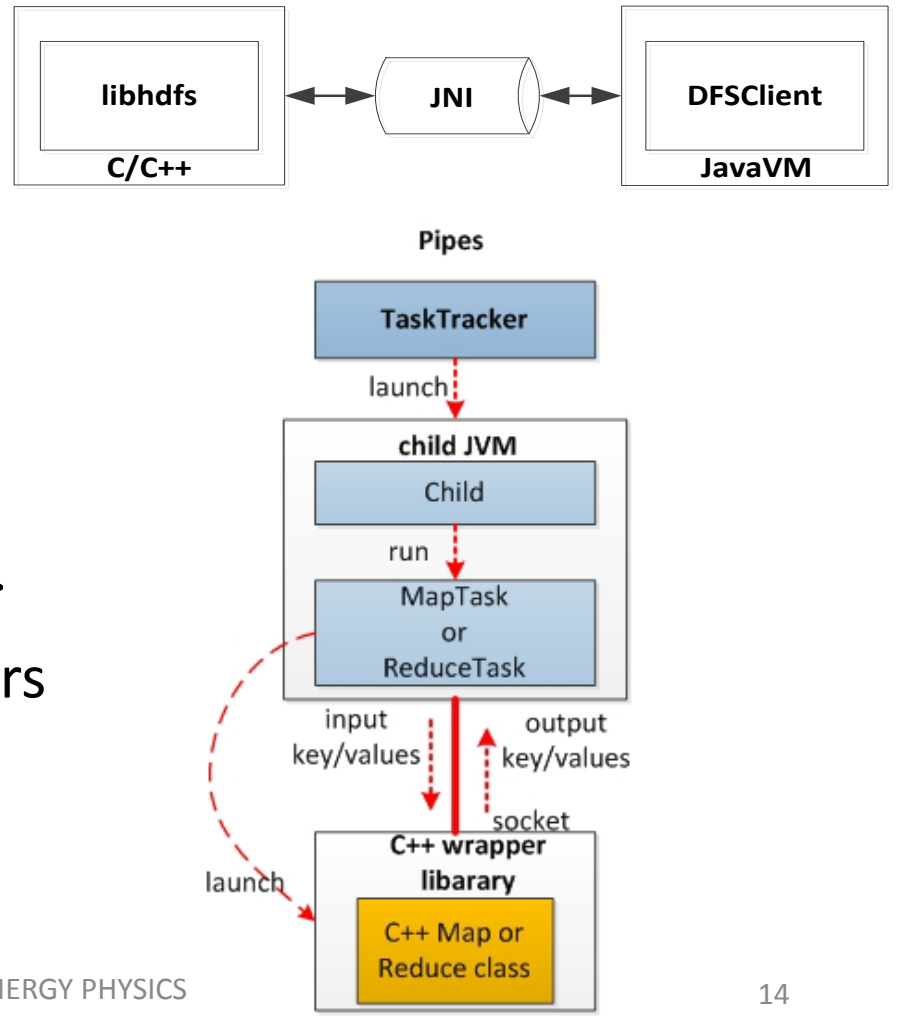


New Libraries and Classes to Support HEP Applications

- I/O libraries
  - THDFSFile
  - THDFSSystem
- Job execution libraries
  - MapRunner
  - ReduceRunner
  - RootFileInputFormate
  - RootFileRecordReader
  - OutputCollector
  - RootSerializer
  - RootDeserializer
  - ReduceInputReader
  - RootFileWriter
  - RootFileReader

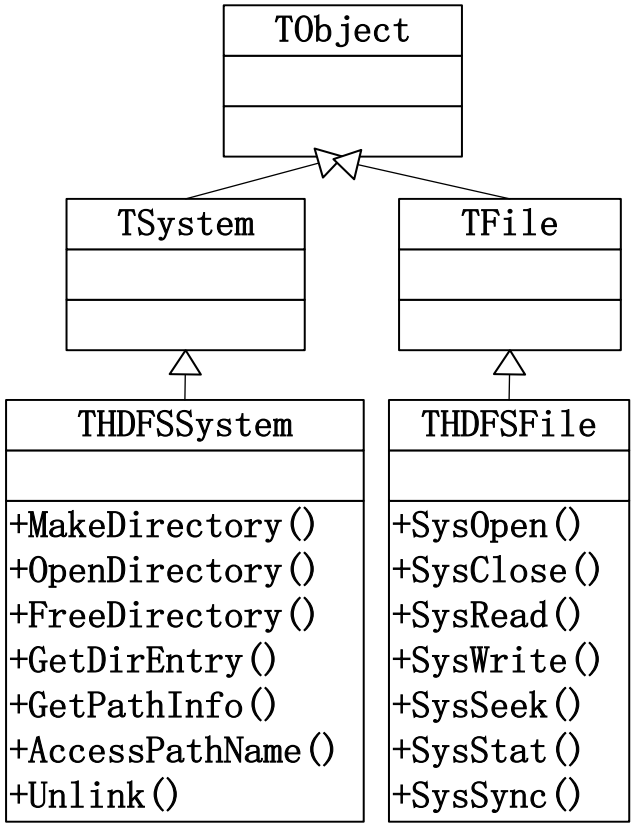
Original Way to Use C++ in Hadoop

- Data Access:
  - C/C++ library libhdfs Use JNI to launch a java JVM, and launch the DFSClient in JVM
- C++ program execution
  - Hadoop Pipes
  - Data input/output by <key, value>
  - C++ code process <key, value> pairs
  - Not suitable for HEP



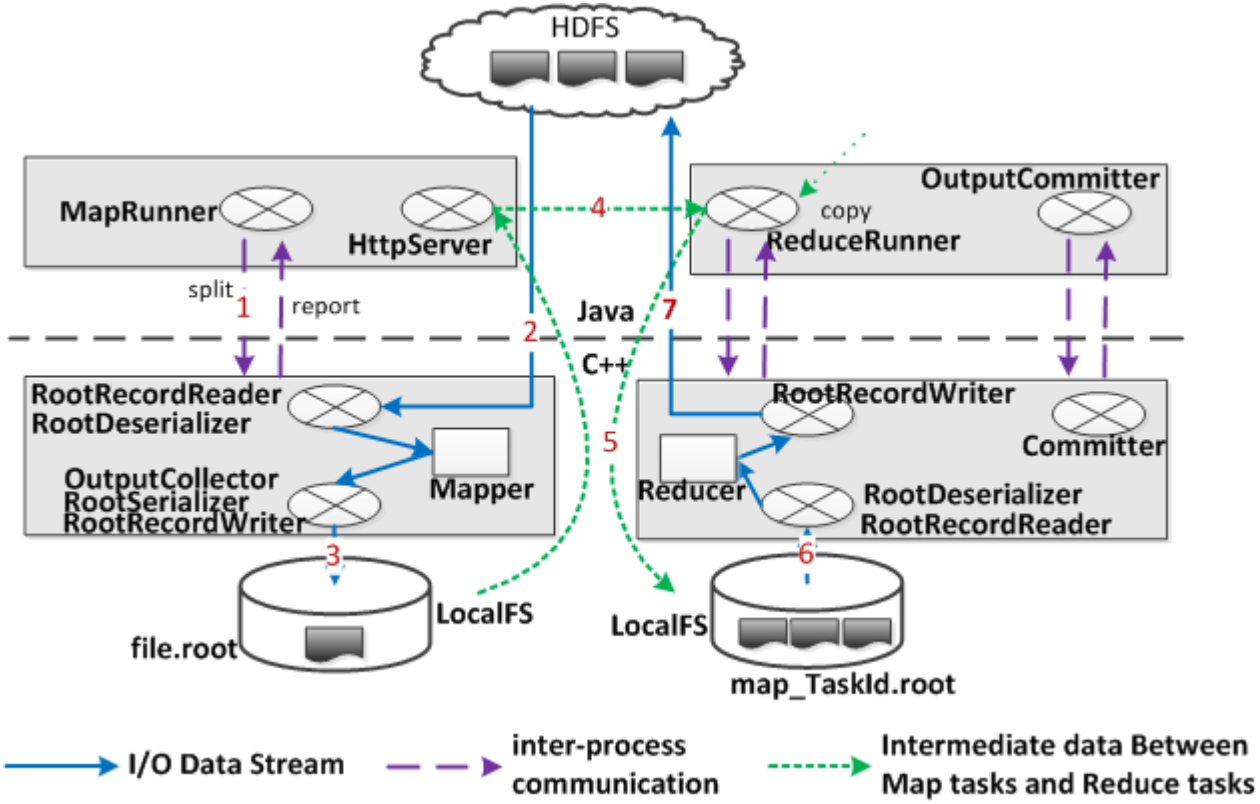
Optimization of Data Access in HDFS

- New developed ROOT library
  Based on libhdfs:
  - Class THDFSFile inherits from class TFile to make ROOT read files in HDFS through JNI
  - THDFSSystem inherits from TSystem, providing directory operations and ringhts management



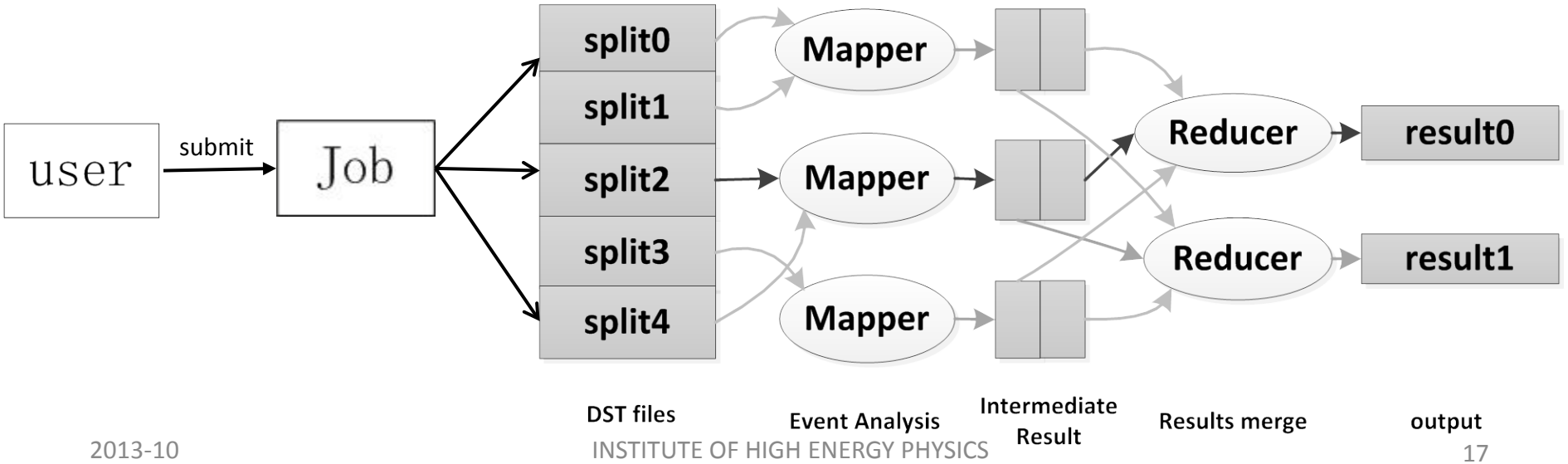
HEP MapReduce Procedure

- HEP MapReduce (different from Internet applications):
  - Java side is in charge of job splitting and scheduling
  - C++ side is in charge of I/O and computation



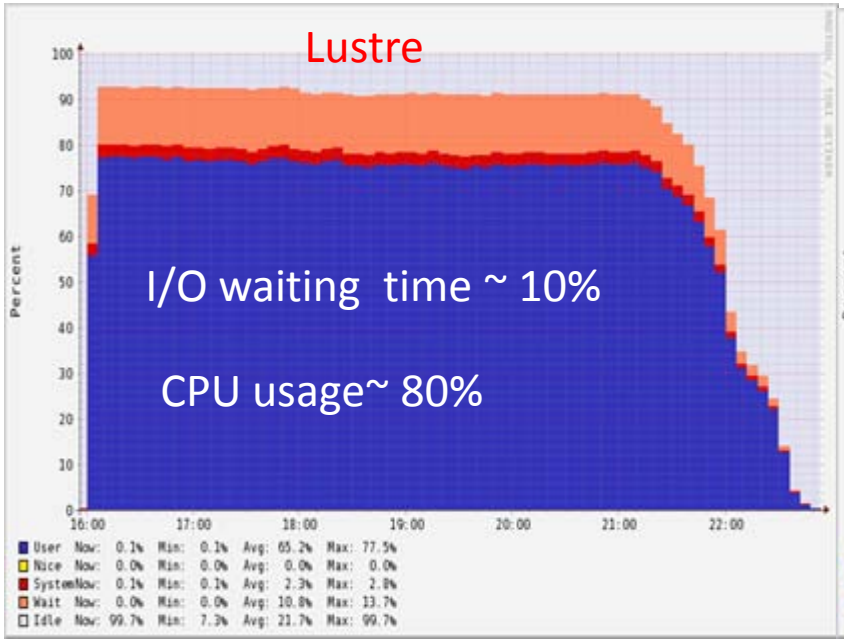
Job Submission and Execution Procedure

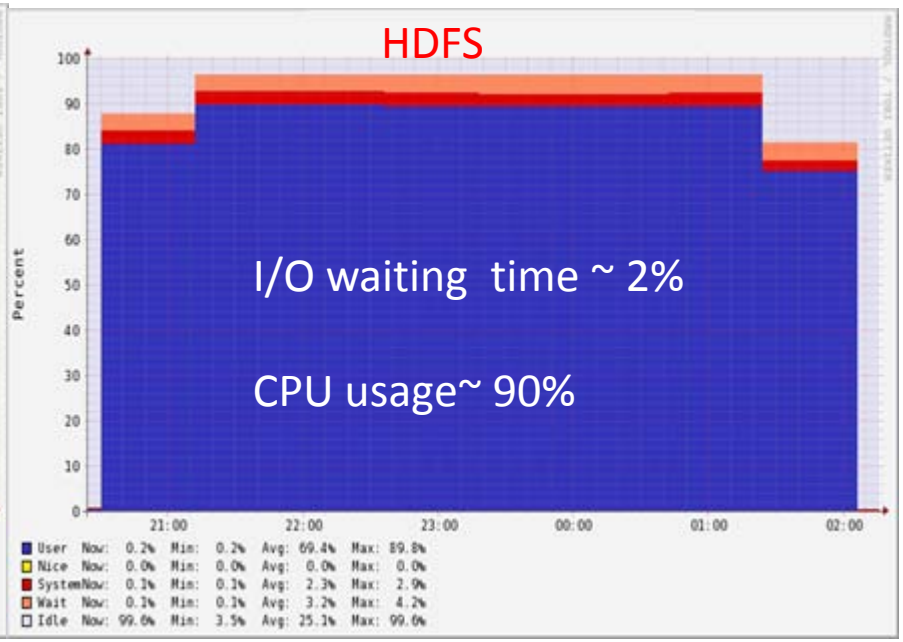
- Submit one job to process a large number of files(i.e 1000)
  - http://twiki.ihep.ac.cn/twiki/bin/view/Hadoop/HowToSubmitAJob
- The job is automatically split into multiple tasks, each task deal with one or several files
- Jobs are scheduled according to files' locations
- Intermediate results are collected and merged into final file(s)



Test Results

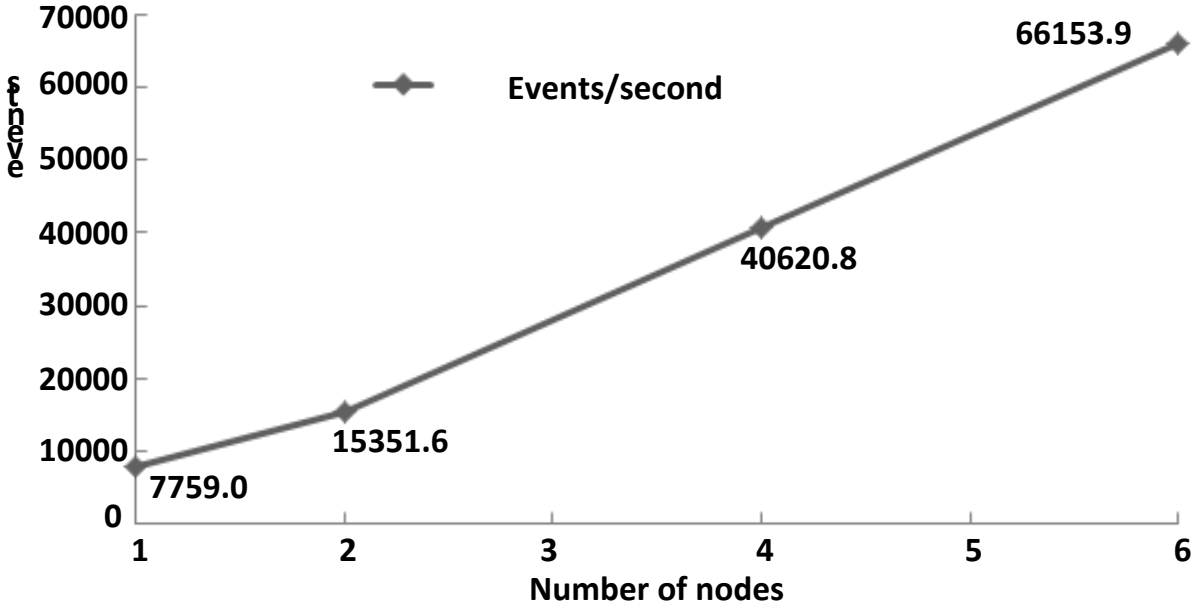
- HDFS & Lustre performance on same cluster
- 7 nodes, 8 cores cpu & 24GB memory
- Jobs read data from LustreFS with enough bandwidth
- Lustre's waiting time is 5 times of Hadoop's





Scalability Test of The Cluster

- Jpsi events analysis with Rhopi analysis codes
- The performance scales linearly with number of work nodes
- The cluster work properly when add or remove nodes: Good scalabliity & redundancy

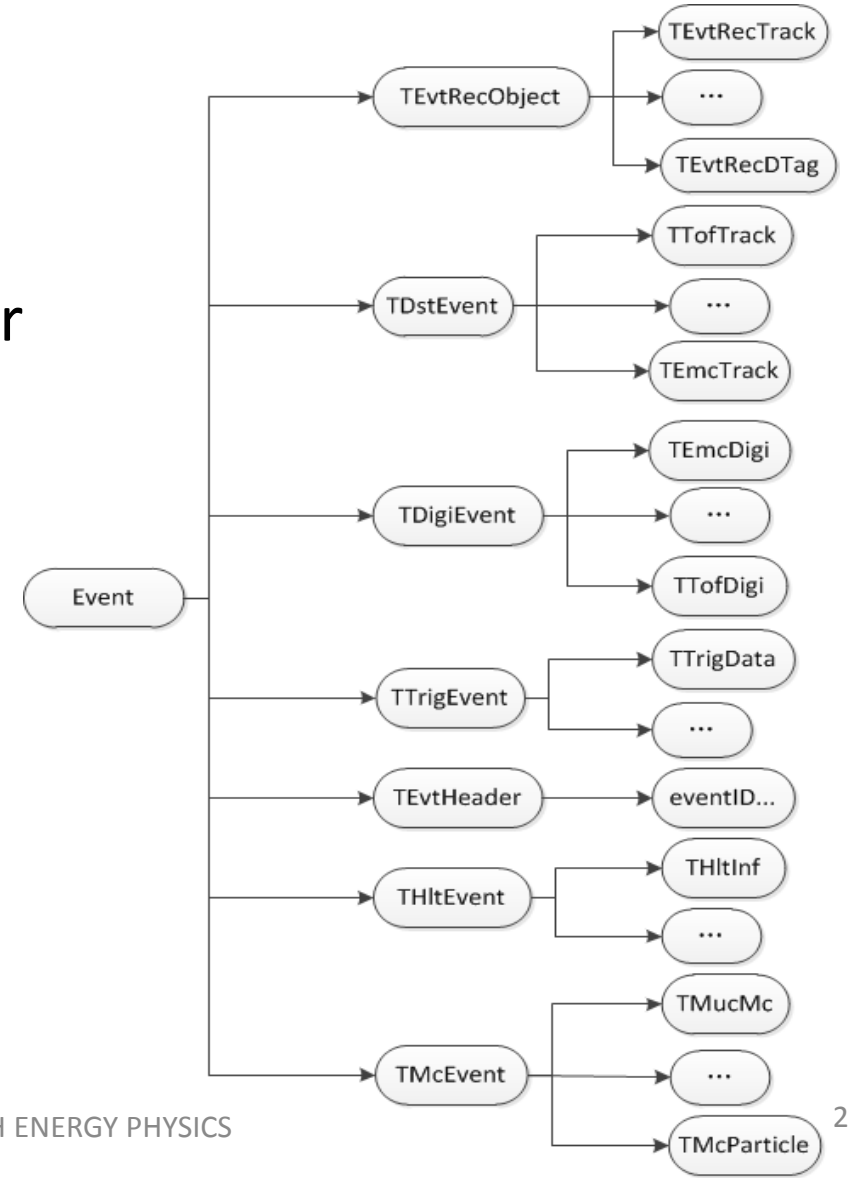


Events Re-clustering & Pre-selection

- Events re-clustering in DST files
- Events pre-selection with TAG

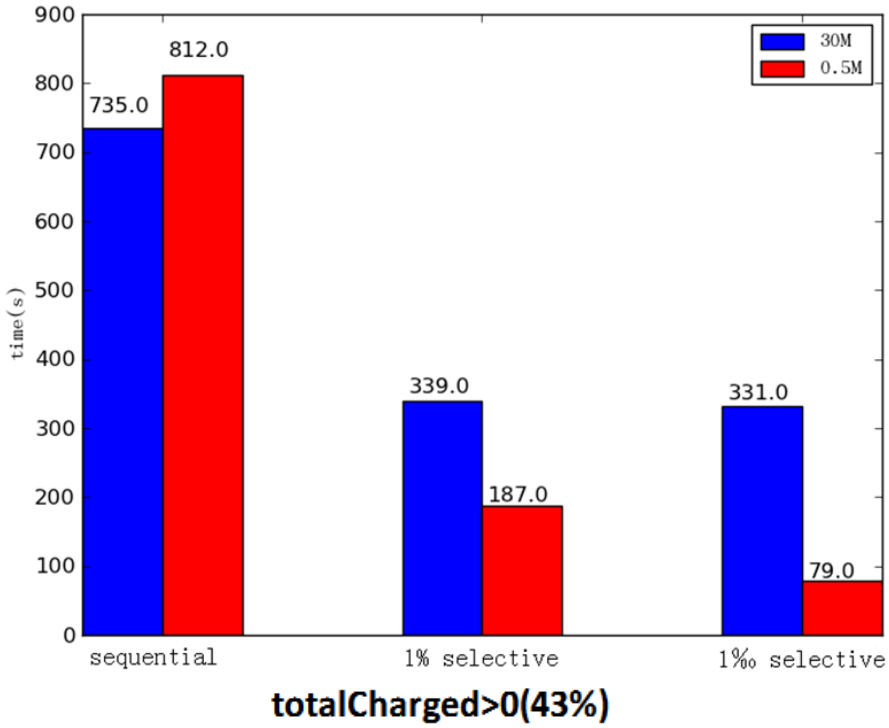
Original I/O & Data Structure

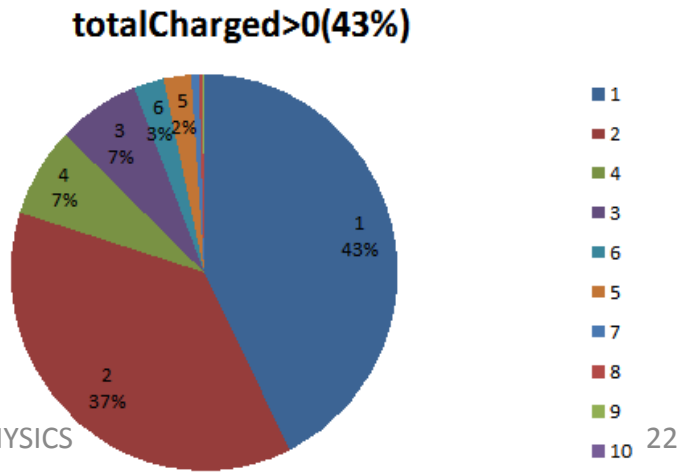
- Seven Branches:
  - TEvtRecObject TDstEvent
    TDigiEvent TEvtHeader
    THltEvent TMcEvent
    TTrigEvent
- Flush Size:
  - 30M
- Event Storage Order:
  - Random



I/O Optimization in DST Files

- Change event flush size from 30M to 500K
- Recluster events by total charged tracks



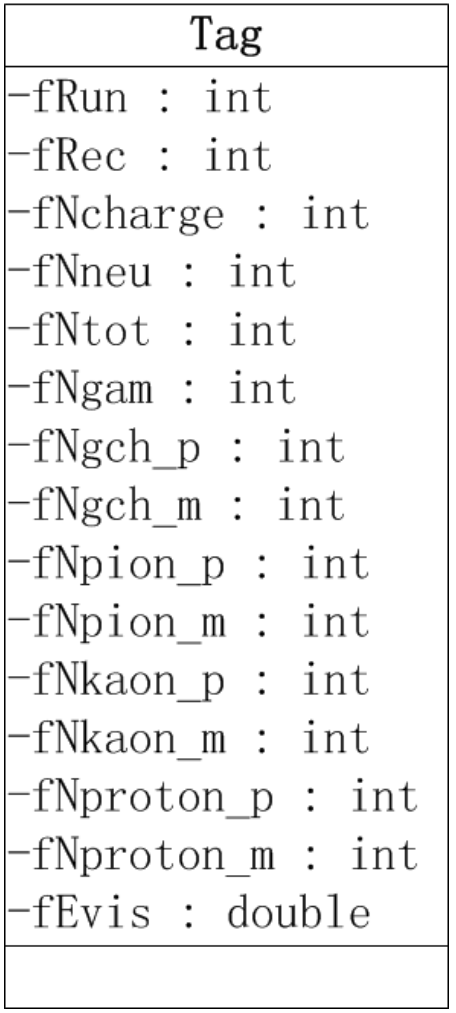


Events Pre-selection for Physics Analysis

- For physics analysis, only a few events will be selected
- Now BOSS reads all the events data from DST files, and data I/O takes more than 50% of the time of the whole job excution
- So we need a more efficient way of events selection
- We build TAG Hbase system to provide preliminary analysis with the MapReduce.

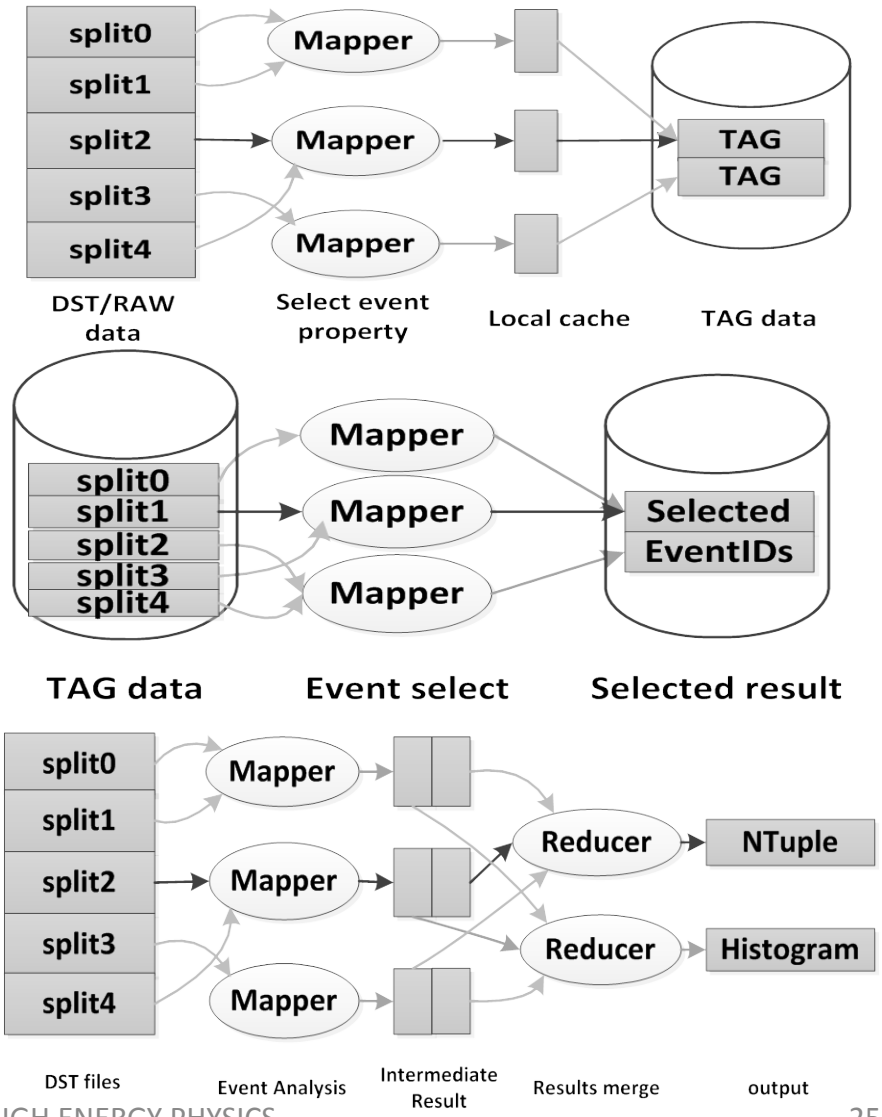
Tag

- Store some basic information of an event
  - fRun: run number
  - fRec: event number
  - fNcharge: the total number of charged tracks
  - fNneu: the total number of neutral tracks
  - fNtot: the total number of tracks
  - fNgam: the number of good photons
  - fNgch_p/m: the number of good charged tracks +/-
  - fNpion_p/m: the number of good pions +/-
  - fNkaon_p/m: the number of good kaons +/-
  - fNproton_p/m: the number of good protons +/-
  - fEvis: definition of the range of the visable energy



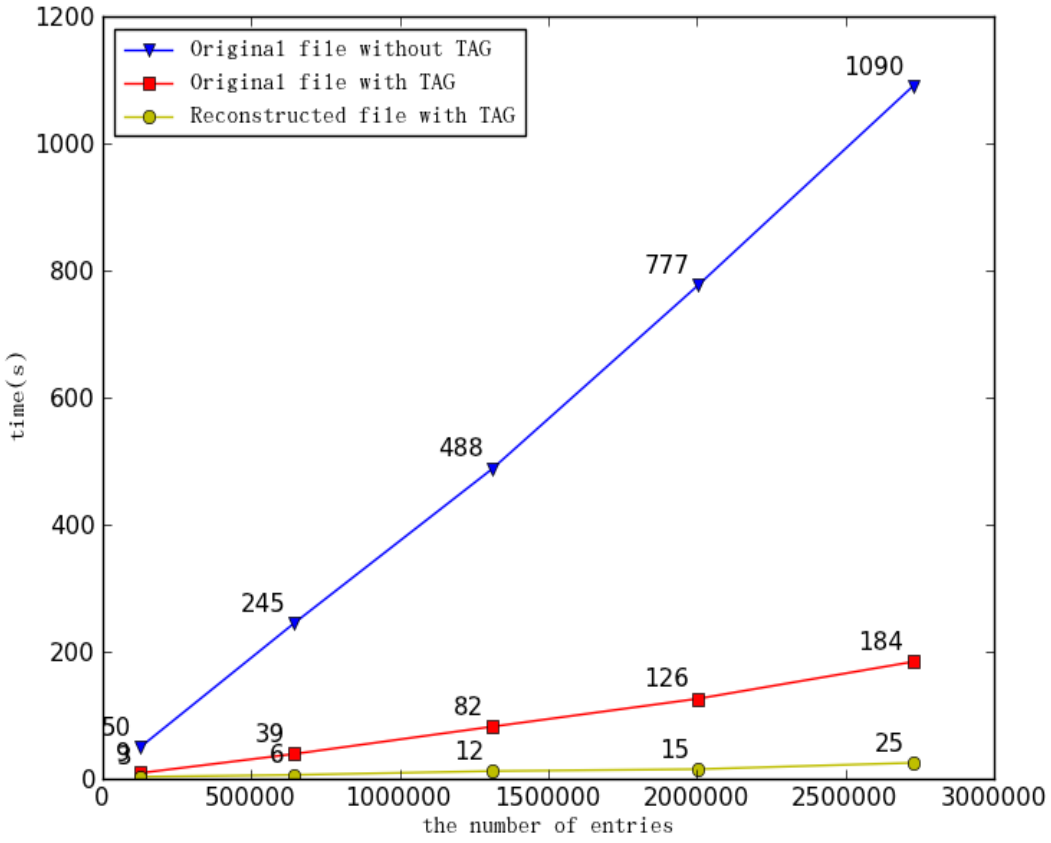
Events Pre-selection Model Based on TAG

- TAG information is generated and stored in Hbase
- TAG table is split and processed in parallel
- The ROOT file is read selectively, and the intermediate output is merged into final results



Performance Improvement of Events Pre-selection and Events Re-cluster

- 2,727,074 events Rhopi analyze
- Pre-selection reduce the job time to 1/5
- Pre-selection and file restructure reduce the job time to 1/40



Summary

- The new data processing system
  - —Reduces the cost to construct system due to efficient resource utilization
  - —Significantly improves the job execution time with events pre-selection & file reconstruction

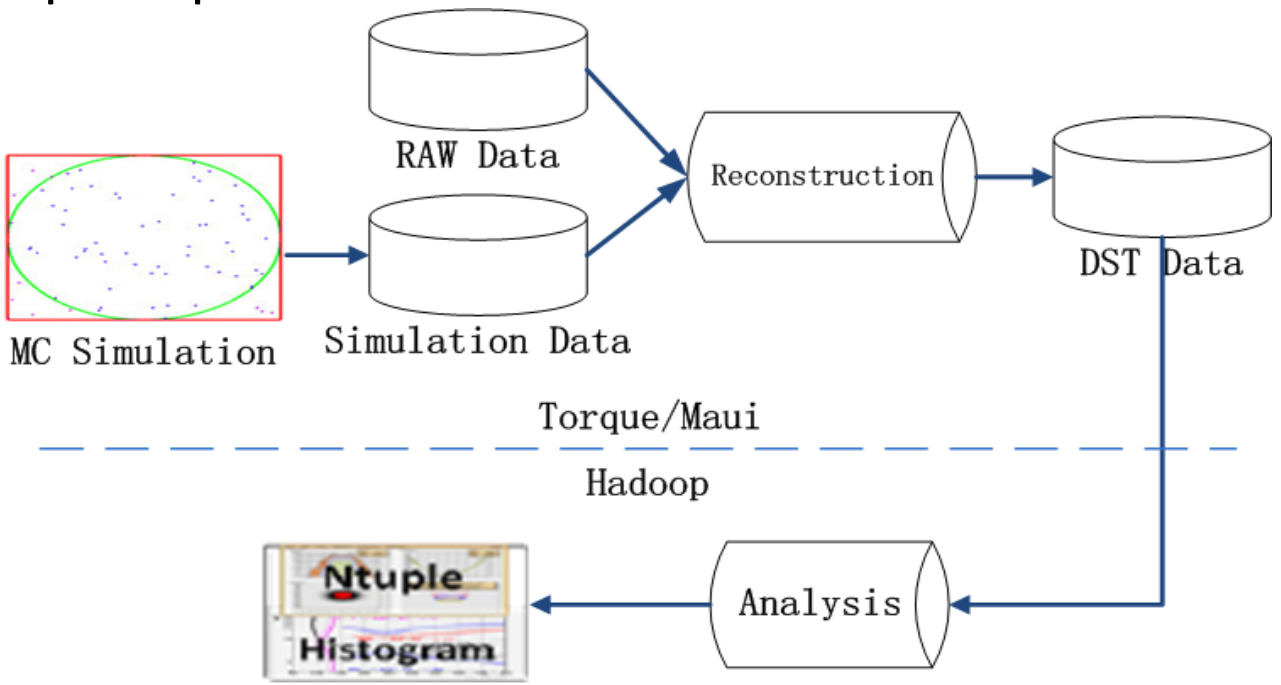
Future work

- Implement random write in HDFS
- Use HDFS random write to improve the MapReduce procedure
- Use Mesos to Integrate Torque and Hadoop on One Cluster

  – Mesos is a cluster manager that provides efficient resource isolation and sharing across distributed applications. It can run Hadoop, MPI, Torque, Spark, and other applications on a dynamically shared pool of nodes

Benefits for Using Mesos

- Dynamic resource sharing among different distributed frameworks
- Improve cluster resource utilization
- Torque/Maui for simulation/reconstruction and Hadoop for analysis
- A complete process for HEP data




THANKS
FOR
LISTENING