EventDB: an event indexer and caching system for BESIII experiment

Yaodong CHENG

On behalf of Scientific data management Project

CC-IHEP, CAS

chyd@ihep.ac.cn

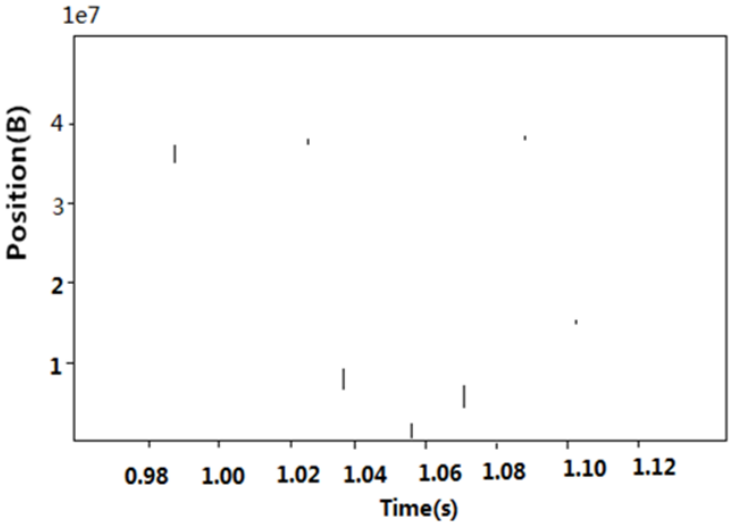
2018-7-10

Contents

- Motivation
- System architecture
- Design and implementation
- Performance Test and results
- Conclusion

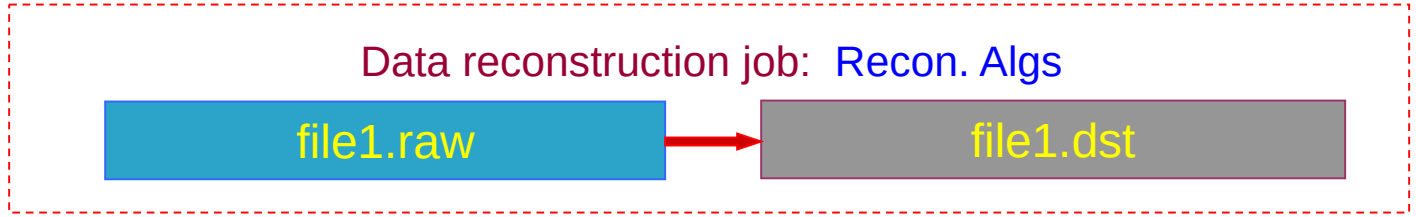
Motivation

- We have found from FS log that stride-read is the main IO pattern during physics analysis
- Several problems with the traditional file-level data management
  - Read too much more data than needed in case of full file scanning
  - File-based cache efficiency is low
  - File-based transmission overhead is high
- Goal: allow **fast** and **efficient selection** of events of interest, based on various criteria, and provide references that point to those events in millions of files

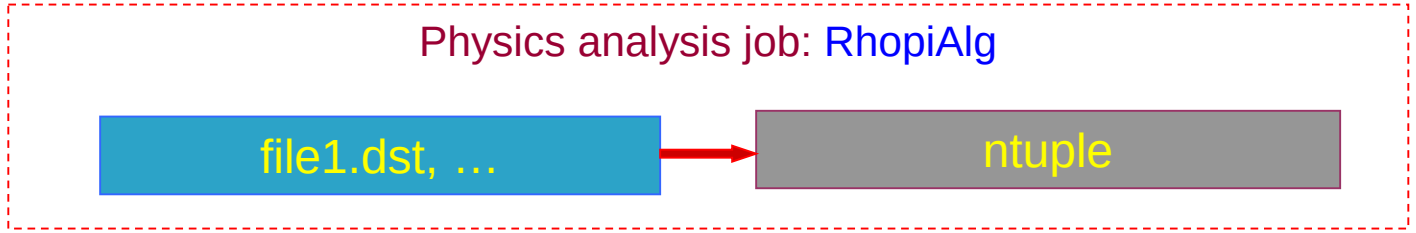


One example of stride-read

Related work: BESIII experiment

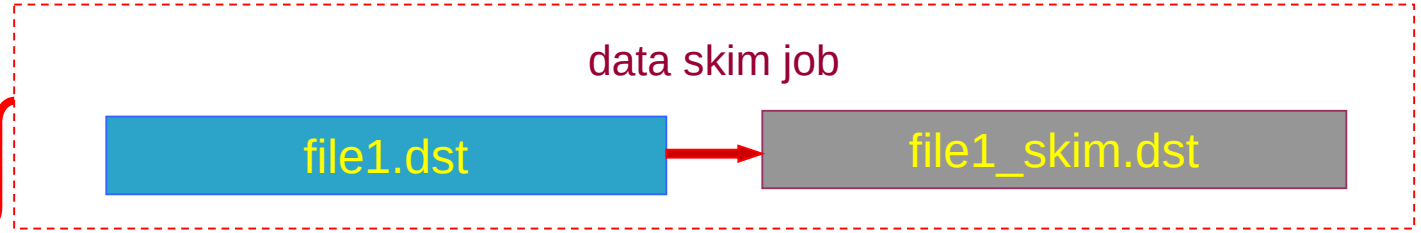


Current data processing (1)

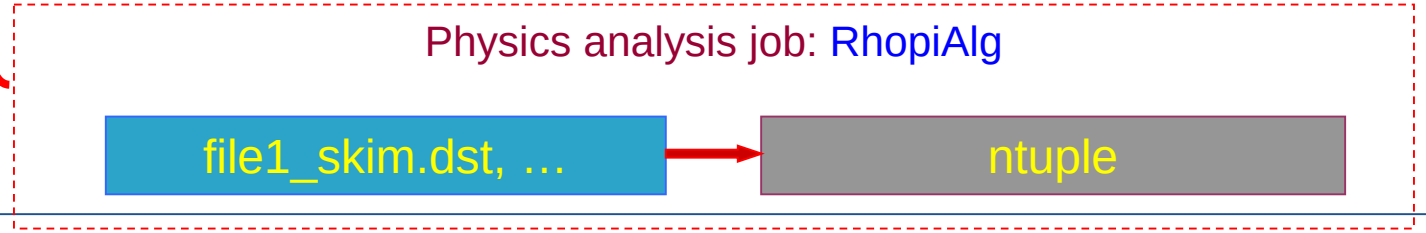


Analyzing all of reconstruction files

Or (2)



Analyzing skimmed files



Faster, but occupies more storage space

Related work: BESIII experiment

Tag based analysis

Reconstruction File

event 0	MdcTrackCol	TofTrackCol	……
event 1	……	……	……
event 2	……	……	……
……	……	……	……
event N	……	……	……

TagWriterAlg

Production job

Tag File

event 0	entry	runId	eventId	Total_charged
event 1	…	…	…	…
event 2	…	…	…	…
……	…	…	…	…
event N	…	…	…	…

Size of TAG file is ~1/700 of DST file

Input for analysis

1) Users use TAG file as job input and set tag selection criteria (“B>=2 && C=1……”)

2) Read each entry in TAG file to choose events satisfied the criteria and read them from DST (reconstruction) files

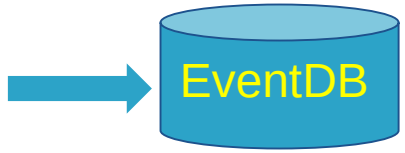
This work was done by Dr. Ziyan Deng etc

Related work: BESIII experiment

- User interface
  - ✓ One TAG file for each reconst. file, can be produced during or after reconstruction
  - ✓ Physics groups or individual users can define the tag files
  - ✓ Specify only tag files in analysis job
- Performance
  - ✓ Size of TAG file is ~1/700 of DST file, depending on what saved in TAG
  - ✓ CPU time of tag based analysis is similar with directly analyzing skimmed DST files, much less than analyzing full DST files.

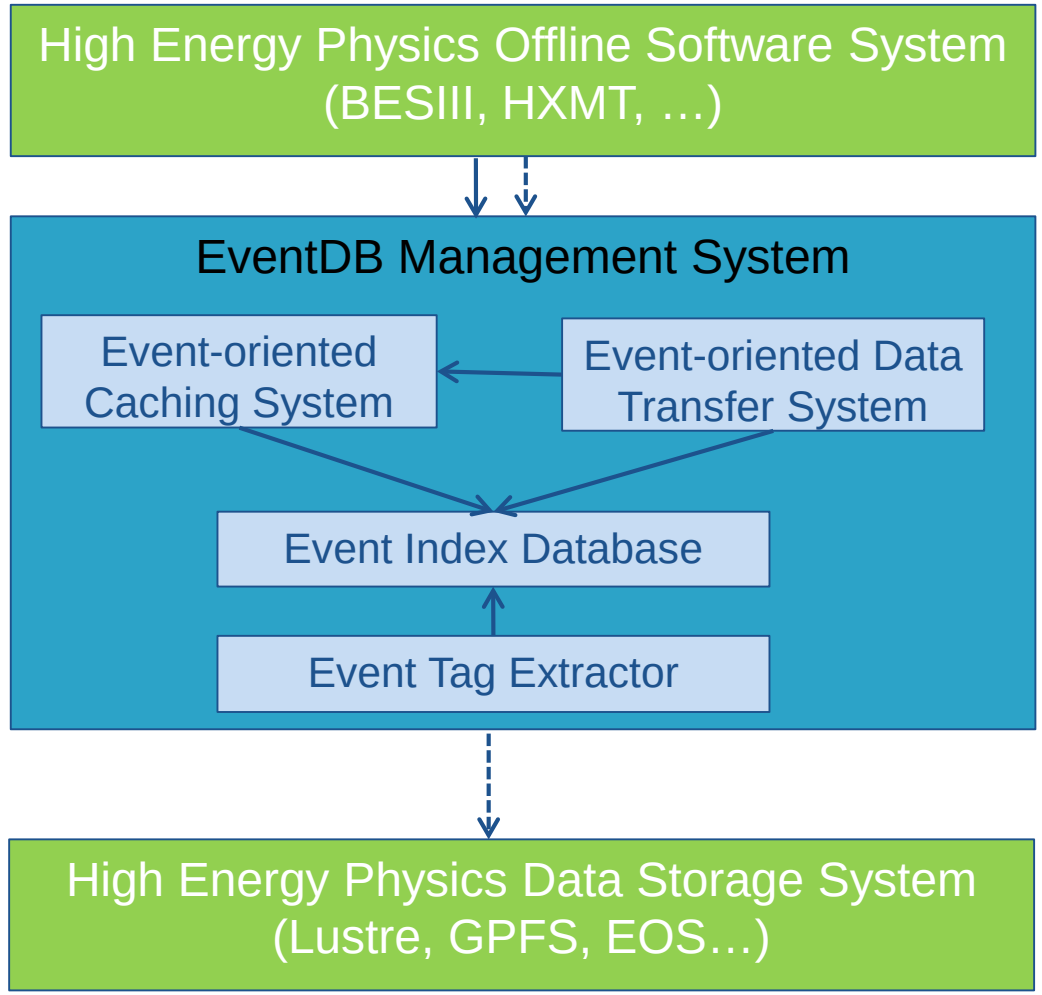
✓ Reduced the disk consumption by skimmed data

✘ Have to scan all of TAG files for specified selection criteria



System architecture

- EventIndexer
  - Event Index Database
- EventAccess
  - Event-oriented Data Transfer System
  - Event-oriented Caching System
- EventExtractor
  - Event Tag Extractor
  - Scan all of DST files and extract event attributes into DB



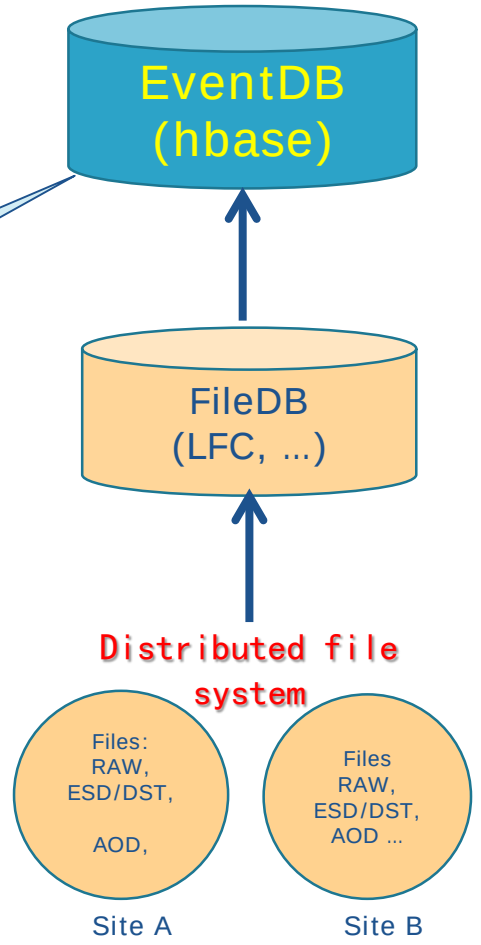
--> Data Flow

⟶ Control Flow

EventIndexer

- Event-level metadata system intended to discover and select events of interest to an analysis
- Store event TAGs and its location in files (filename+EventOffset)
- Export index file after selection

1. The basic attribute of the event: RunID, FileID, EventID, VersionID (TAG)
2. The tag is defined by different physics analysis group



EventIndexer Data Storing

How to store tag in eventIndexer

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
-8026#Neutral#009	pips1. dst-0, pips1. dst-5, pips2. dst-13, …

Part of the clustered data

```
-8107#NShowes#80000020 column=data:FileIndex, timestamp=1498631253359, value=['run8107_fil
                       e1.root-625', 'run8107_file1.root-14465', 'run8107_file1.root-20657
                       ', 'run8107_file3.root-55790', 'run8107_file3.root-60458', 'run8107
                       _file2.root-85331']
-8107#NShowes#80000020 column=data:count, timestamp=1498631253359, value=6
-8107#NShowes#80000021 column=data:FileIndex, timestamp=1498631253364, value=['run8107_fil
                       e1.root-10640', 'run8107_file3.root-52085', 'run8107_file3.root-522
                       75', 'run8107_file3.root-53102', 'run8107_file2.root-85528', 'run81
                       07_file2.root-87922', 'run8107_file2.root-108986', 'run8107_file2.r
                       oot-112296']
-8107#NShowes#80000021 column=data:count, timestamp=1498631253364, value=8
```

Part of the data in hbase

NoSQL database : HBase

Rowkey: runNo#PropertyName#Value

Value: Filename-Eventoffset,……

Attributes extracted for BESIII

- BeamEnergy, beam energy: double
- Ntracks, total number of tracks in the event: int
- Nshowers, total number of showers in the event: int
- BeamVx, x-position of the beam spot: double
- BeamVy, y-position of the beam spot: double
- BeamVz, z-position of the beam spot: double
- number of gamma: int
- number of K+: int
- number of K-: int
- number of Ks: int
- number of pi+: int
- number of pi-: int
- number of pi0: int
- number of Lambda: int
- number of ALambda: int
- number of e+: int
- number of e-: int
- number of mu+: int
- number of mu-: int
- number of p+: int
- number of p-: int
- number of eta: int

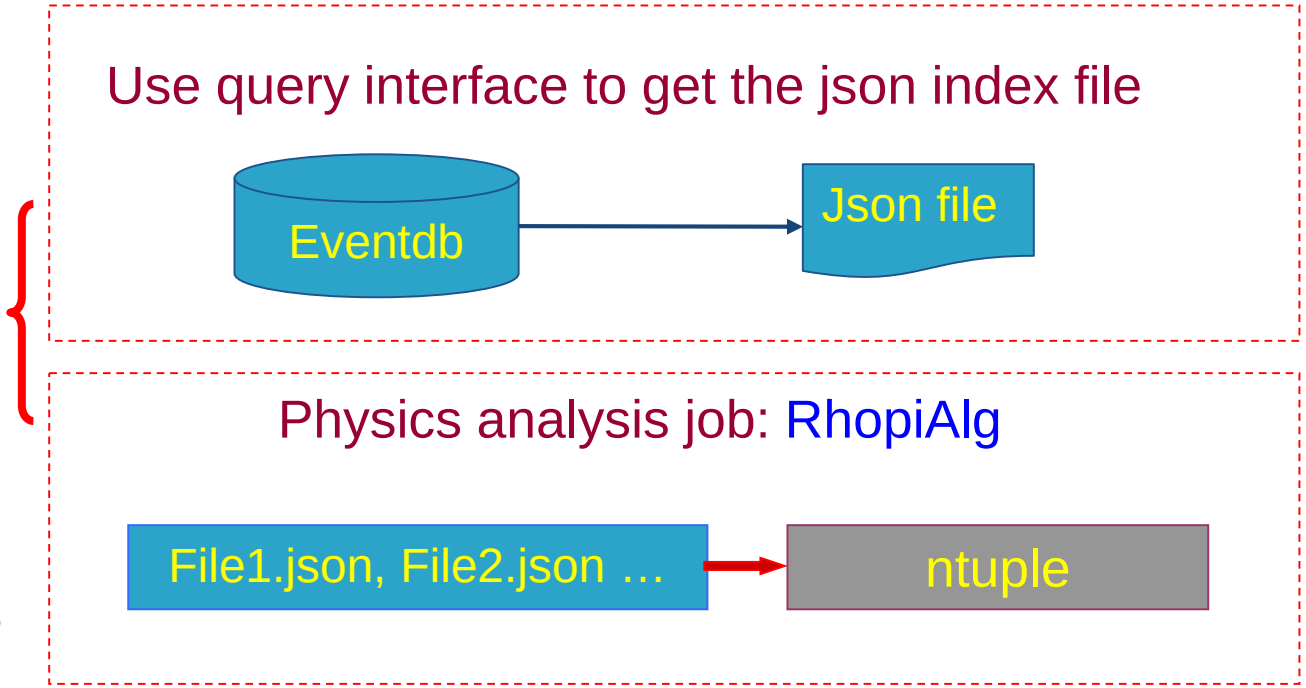
EventDB-based analysis

EventDB based analysis:

1) User get json index file through EventDB by specifying some criteria

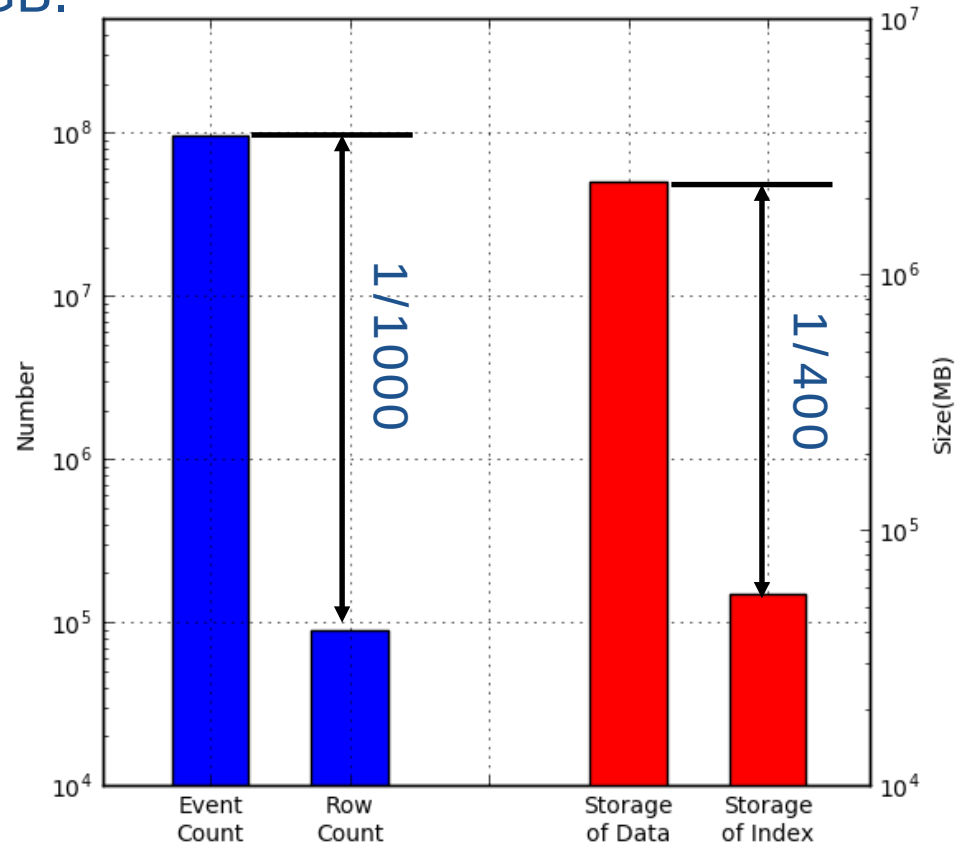
2) Users use json file as job input and set tag selection criteria ("B>=2 && C=1.....")

3) Read each entry in json file to choose events satisfied the criteria and read them from DST (reconstruction) files



Storage Of the Index System

There were nearly 100 million BESIII events with an original data size of 2.2TB in testing system. After compression, the index contains less than 10 million lines of records, the size is 55GB.

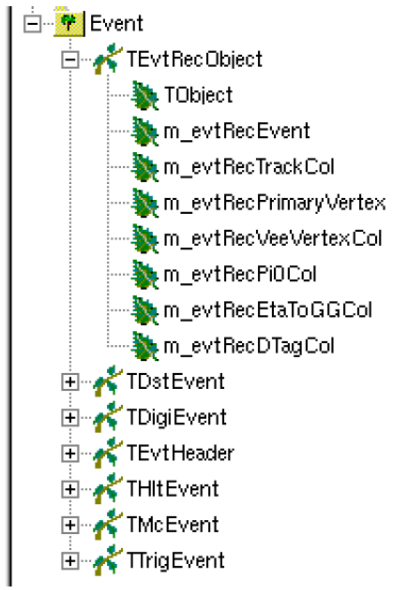


Index number and space

Search time

Event Cache

- Record the file access patterns during the physics experiment analysis to analyze the hot event data
- Keep the high access frequency events into HBase located in the memory or SSD, to improve event accessing performance



DB: Hbase
Rowkey: FileName+EntryID
Value: String of Serialized TTree of one event

Rowkey	columns	
Fname-EntryID	TBossFullEvent	Length

Event Cache Interfaces

The Interface :

HTree(const string& table, const string &family); //HTree data structure imitates TTree interface providing data access services, parameters are hbase table name and column cluster name.

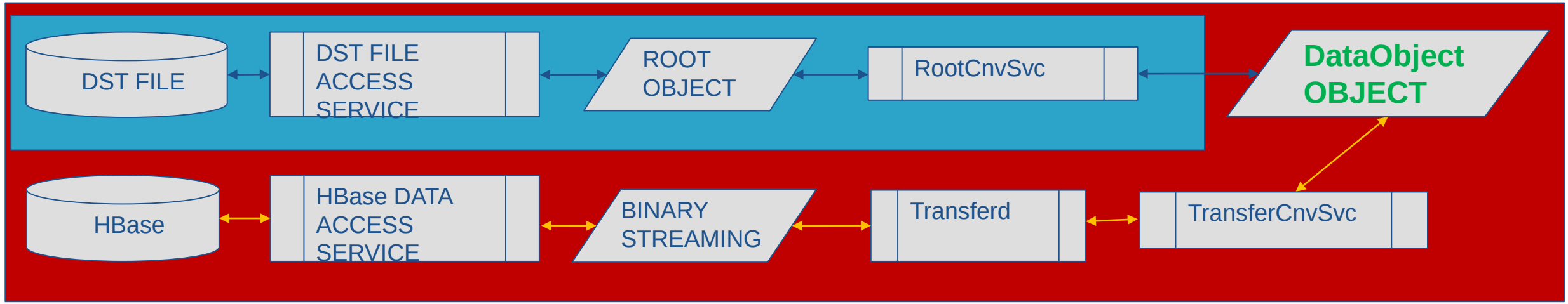
init(const char *server, const char *port); //Initinalize thrift connection service, parameters are the ip and port of thrift service.

tree->setBranchAddress("TEvtRecObject", evtTecObj);

getEntry(const string&rowKey);

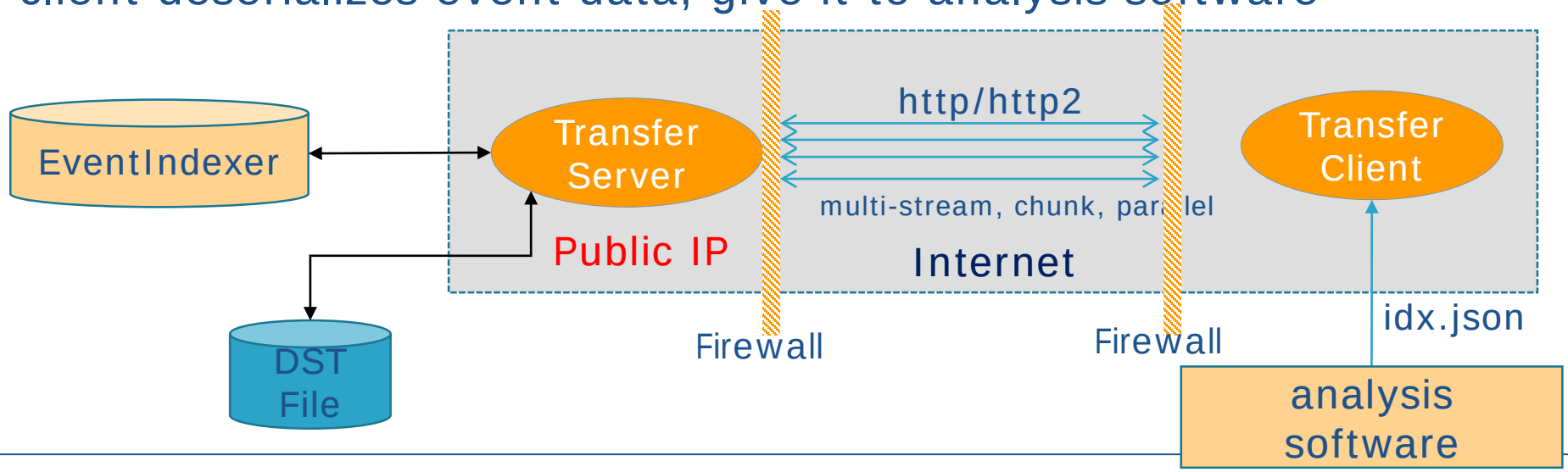
void allEventcopy(char *filename,char* startID,char* num); //import the events into hbase.

Data conversion:



Event Transfer Workflow

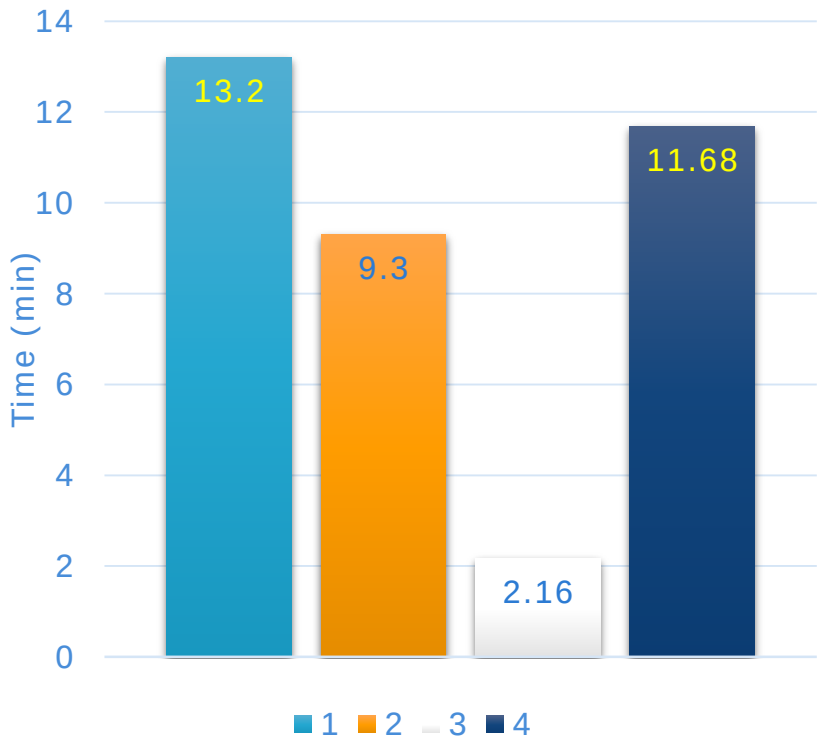
- 1) Analysis software running on remote site tells which events will be used in an analysis, usually giving a json index file
- 2) Transfer server parses the index file and then process it in parallel
- 3) Transfer server firstly get event location (file and offset) from EventIndexer, then retrieve event data from DST file using ROOT framework
- 4) Transfer server serializes event data and transfer it to transfer client
- 5) Transfer client deserializes event data, give it to analysis software



Data analysis performance testing

- Select 11237 from 99130 events
- Network bandwidth: 1000Mbps

Case	site	EventIndexer	Event Cache
1: Original analysis	local	No	No
2: Original analysis + EventDB	local	Yes	No
3: Original analysis + EventCache + EventDB	local	Yes	Yes
4: Original analysis + Remote + EventDB	remote	Yes	Yes



Conclusion

- Event-oriented data management will solve some problems of the file-based data management system
- Use NoSQL database to store the index and data of the event could be more efficiency and easy accessing
- Event-level transfer only transmits data interested, and we will add some new functions, eg HTTP2 support, etc
- Under development and try to use it in production
- Thanks for the related work and help provided by Beijiang Liu, Ziyan Deng, and members from scientific management project

Thank you!