Integrating LEAF to data management workflow in LHAASO

Haibo Li, Yaodong Cheng, Qi Xu, Qiulan Huang
On behalf of IHEP Computing Center, CAS
September, 2018

Outline

- LHAASO experiment introduction
- LEAF architecture and implementation
- Evaluation results
- Summary

The LHAASO project

- The Large High Altitude Air Shower Observatory (LHAASO) project is a new generation allsky instrument to perform a combined study of cosmic rays and gamma-rays in the wide energy range 1011--1017 eV.
- The experiment located in Daocheng, Sichuan province (at the altitude of 4410 m)

Offline data processing platform

- After the experimental data is acquired by DAQ, it enters the offline computing platform.
- Provide support services for data storage, transmission, sharing, analysis and processing.

Small on-site data center at Haizi Mountain observatory (~ 4500m)

~2000CPU cores and 300TB disk storage for calibration and rapid reconstruction

~300Mbps

Operation center at Daocheng city

~1Gbps

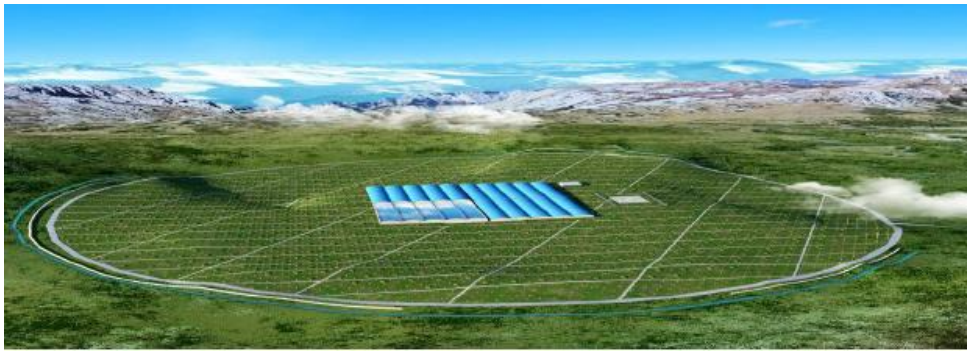
Distributed Computing sites (grid/cloud)

Large Offline data center at CC-IHEP

~4000 CPU cores and 4PB disk storage, 20PB tape storage for simulation, reconstruction, analysis, data storage and archive

LHAASO Computing requirements

- ~6 Petabytes of data annually generated by the LHAASO detectors
  - 6 PB of raw data, and >200TB of reconstruction data
  - Totally >60PB for ten years
- >2 Petabytes of data generated by MC simulation
- To build one distributed computing system containing about 6000 CPU cores to process the data
  - ~ 4500 CPU cores for reconstruction, analysis, …
  - ~ 1500 cores for production



The challenge of distributed computing

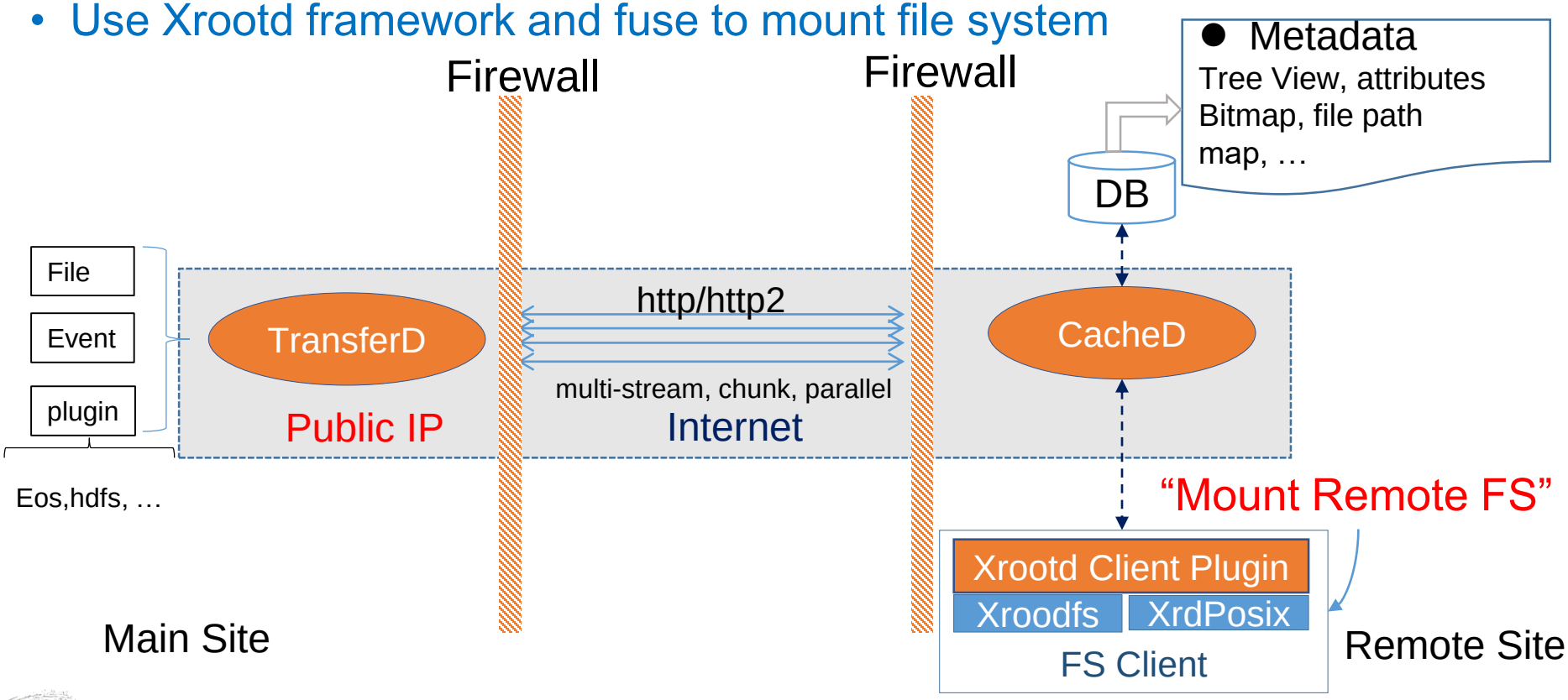
- Computing job is usually scheduled to the site where the input data was pre-staged in using file transfer system
- Low CPU efficiency
  - If one site doesn't have enough storage space, the CPU couldn't be fully used.
- Not flexible
  - Site manager decides which data will be transferred
- Difficult to work in dynamic cloud environment
  - VMs can be created in public cloud on demand, but analysis job can't run without input data
- Too much data is transferred
  - The whole file is transferred to remote site, but user's job only is usually interested in a few of events in the file

What's LEAF?

- LEAF is a data cache and access system across remote sites
  - Same file system view at local and the remote sites
  - Good access speed over WAN
  - Client requests are served as soon as one small fraction of file is available before one whole file is fully downloaded
  - Portable, compatible and scalable
  - Secure and reliable

Architecture

- Full Metadata synchronization from main site periodically
- Data transfer technologies: multi-stream, chunk, non-block, etc
- Use HTTP protocol to go through firewall
- Use Xrootd framework and fuse to mount file system



File Transfer Service

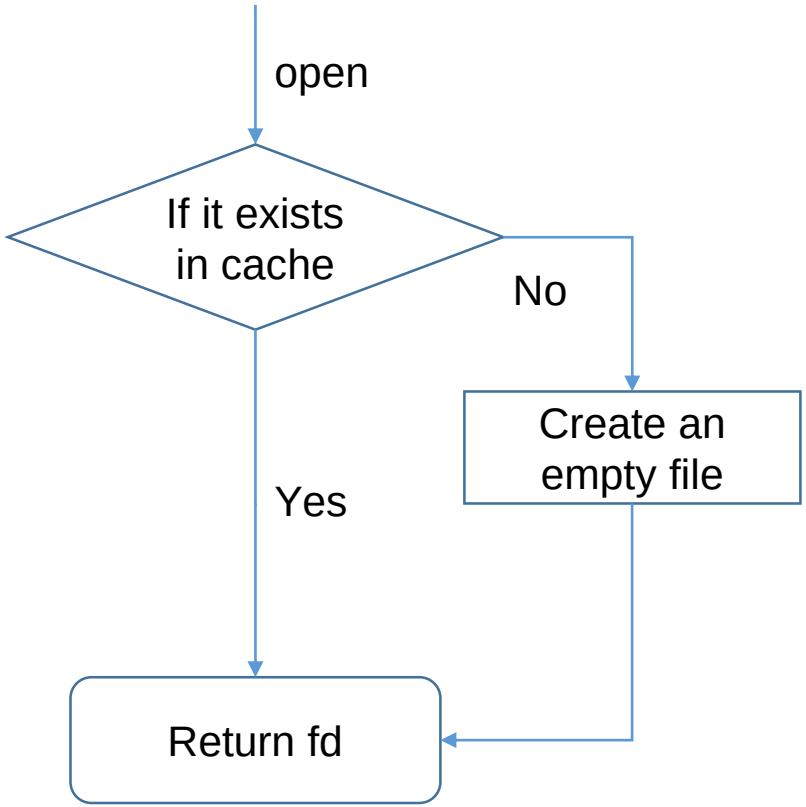
- Two components
  - TransferD: daemon running at Main site
  - Client library: deployed at remote site, called by CacheD
- Based on Tornado web framework
  - a Python web framework and asynchronous networking library
  - support non-blocking network I/O, suitable for long polling, WebSockets, long-lived connection
- If file transfer service receives a request, it will download or upload data using multi-streams in parallel
- Client routines have theses parameters: file path, file operation (stat, getdir, read, write, …), mode, offset, …
- Easy to go through firewall using HTTP protocol
  - Usually client doesn't have public IP behind the firewall

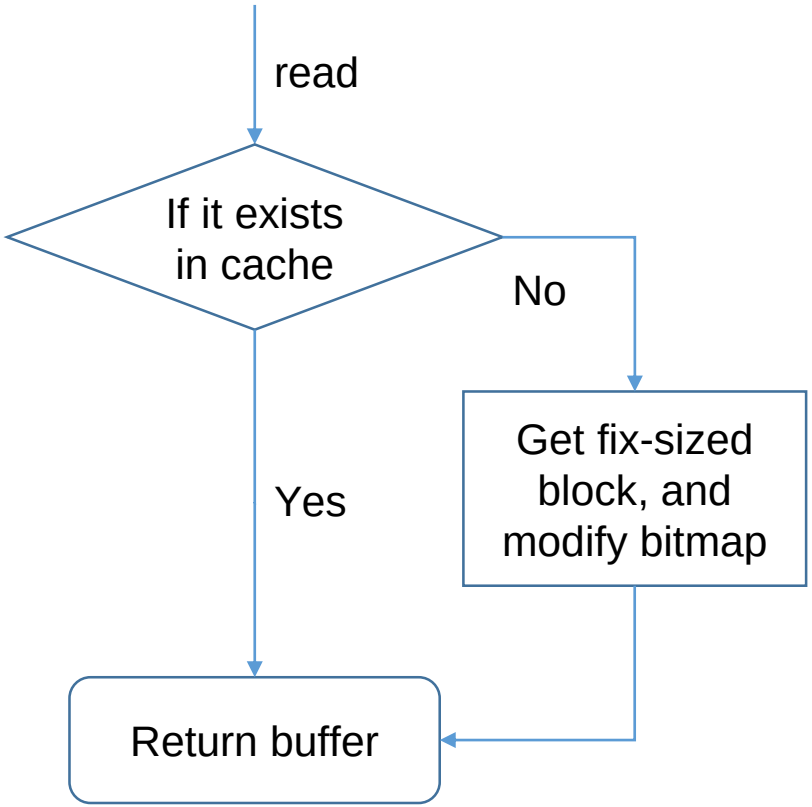
Disk Cache Service

- Three components
  - CacheD: daemon running at remote site
  - DB: store file metadata and bitmap
  - Client tool and library: called by xrootd client plugin
- CacheD will get all entries periodically from main site once the "exported" file system is defined
- DB supports Mysql and Ramcloud currently

Open/read workflow

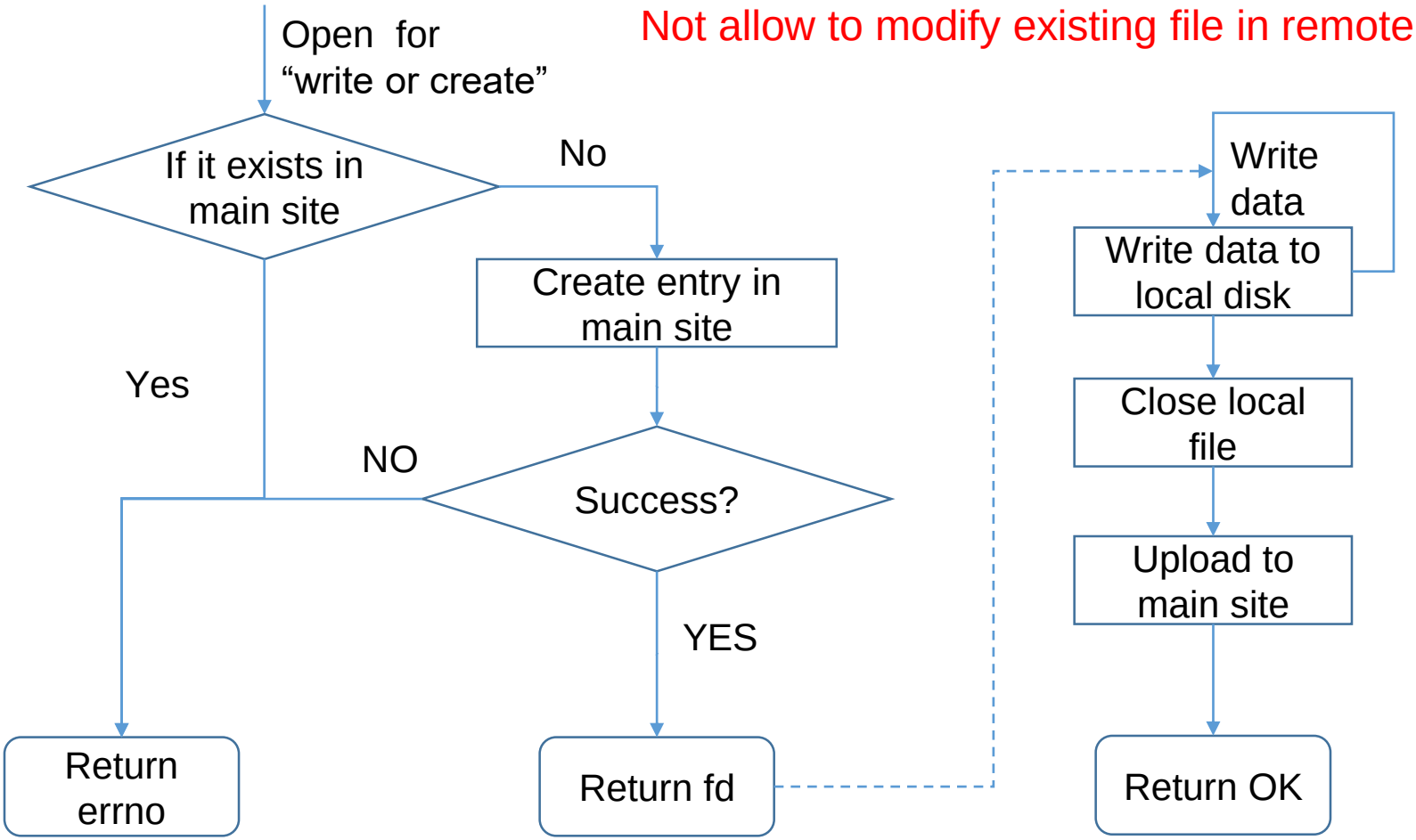
- CacheD creates an empty file on local disk once it receives ‘open’ request from client
- CacheD gets fixed-size block (1MB) from offset specified by ‘read’ operation





Write workflow

- CacheD puts the whole file in local disk, then upload it to the main site later in case of ‘write’ operation



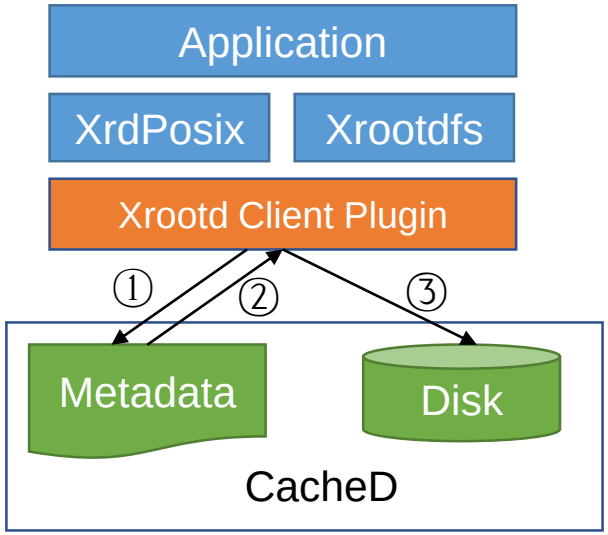
Xrootd client plugin

- Application access data using xrdposix API or xrootdfs
- Implement a xrootd client plugin
  - 1) check if the block is in cache. If not, it calls cached to get the block from main site
  - 2) return physical path of the file
  - 3) get real data from disk using xrootd
- Xrootd client plugin manager
  - /etc/xrootd/client.plugins.d
  - Manage a map between URLs and plug-in factories

    url = root://cached.domain:1094

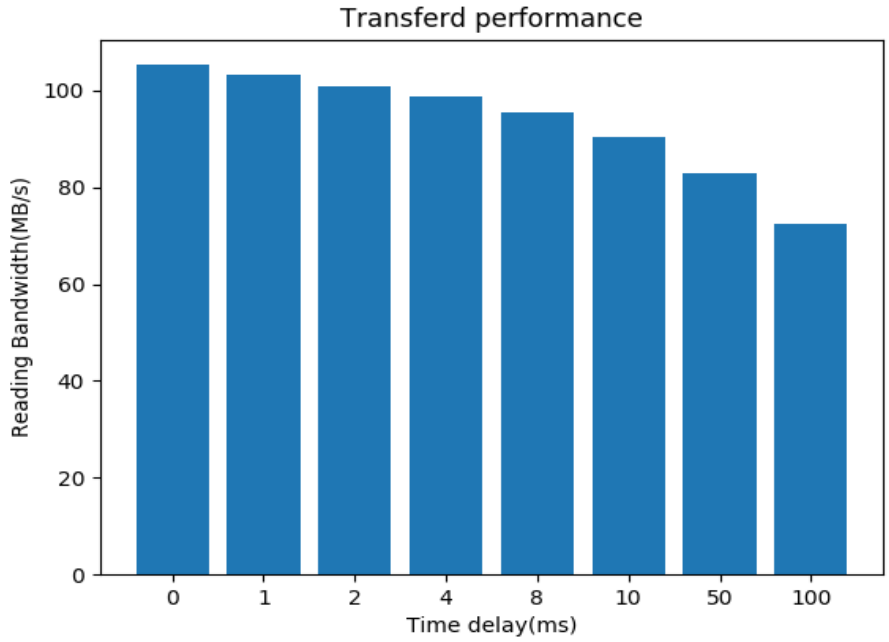
    lib = /usr/lib/libXrdLeafClient.so

    enable = true



Performance evaluation

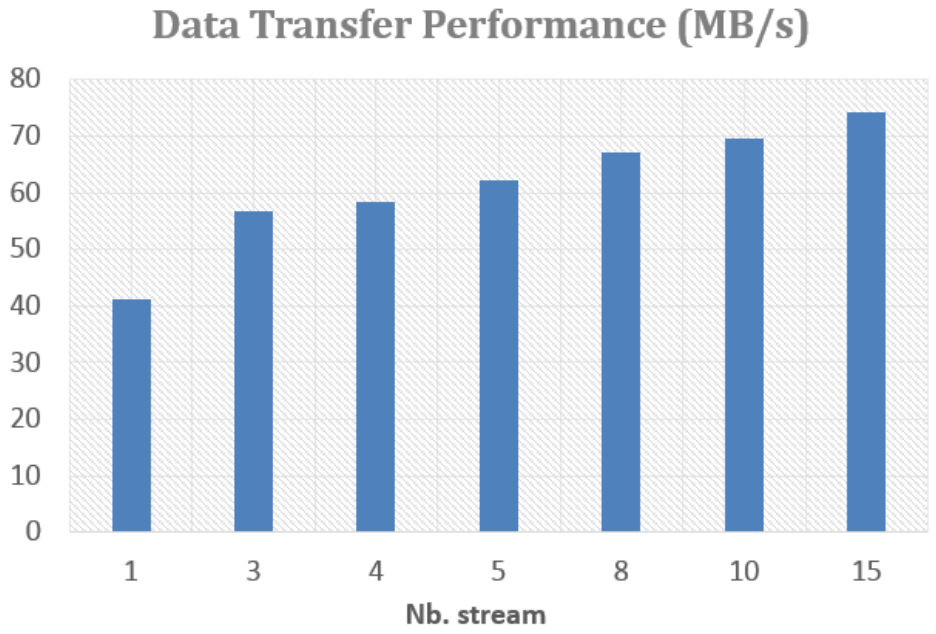
- bandwidth: 1Gbps
- Latency: 1~100ms using tc simulation
- Transfer parameters: long-lived, 1M block, 10 streams
- Results: decreased by 31% (105MB->72.5MB), better than EOS/Lustre



Round trip latency	Transfer performance (MB/sec)
0 ms	105.3
10 ms	90.7
50 ms	82.8
100 ms	72.5

Testbed

- Two sites：IHEP (Beijing) <-> CLAS (Chengdu)
- Distance: ~2000KM，Latency: ~35ms
- Bandwidth: ~1Gbps，Iperf: ~80MB/s
- Performance is getting better with the increasing of stream number



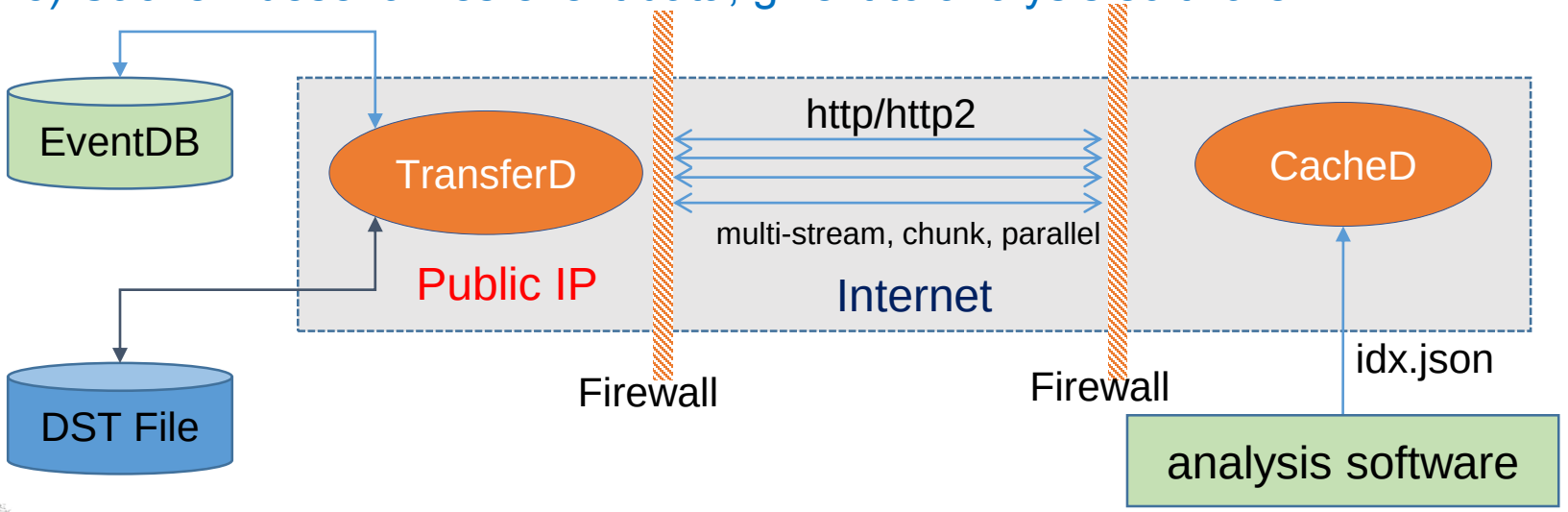
Ongoing Work

- Event-level data transfer
- Depends on another work: EventDB
  - Event-level metadata system intended to discover and select events of interest to an analysis
  - Store event TAGs and its location in files
  - Export index file after selection
- Data transfer service get events in parallel after it receives the request of event index file
  - Index includes file name and event offset
- Only transfer events of interest to reduce the traffic greatly
  - For example: 0.1% events of interests in job analysis
  - Refer: http://iopscience.iop.org/1742-6596/523/1/012008

ACAT2013: High performance computing activities in hadron spectroscopy at BESIII

Event transfer workflow

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



Summary

- LHAASO distributed computing has a need for remote data transmission
- LEAF provides a data cache and access solution for accessing data directly from remote site
- Implemented as a xrootd plugin supporting most of HEP applications transparently
- Adding new functions, eg HTTP2 support, event-level transfer, etc

Thanks for your attentions!