LEAF – a data cache and access system across remote sites

Yaodong Cheng, CC-IHEP, CAS

chyd@ihep.ac.cn

ACAT2017, 21-25 August 2017，University of Washington, Seattle

Contents

- Background and Motivations
- Architecture and implementations
- Evaluation results

Background

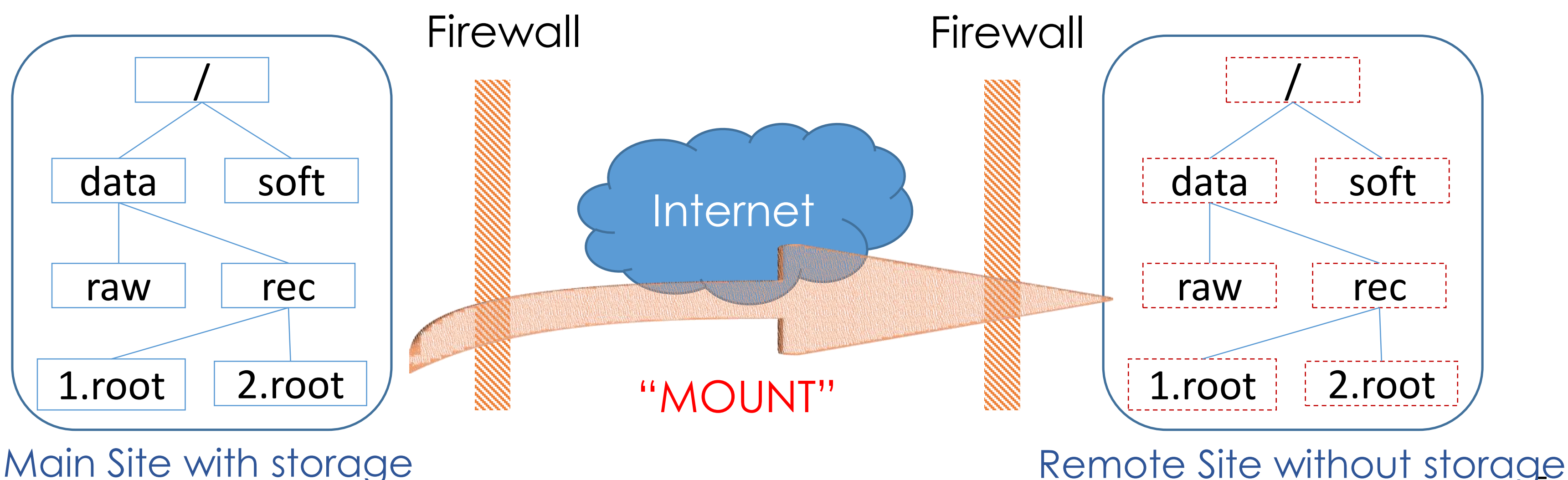
- Distributed computing system is widely used in high energy physics
- WLCG: World Lhc Computing Grid
  - 167 sites from 42 counties
  - >600K CPU cores, 300PB disk, 300PB tape
  - Each site has dedicated people taking care of the system
- BESGrid
  - Grid for BESIII and other experiments in China
  - Use Dirac as middleware
  - 18 sites from 7 countries
  - Most of sites don’t have dedicated administrator
- Public Clouds are gradually popular in HEP
  - Amazon AWS, Microsoft Azure, Alibaba Aliyun, …

Existing problems

- 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

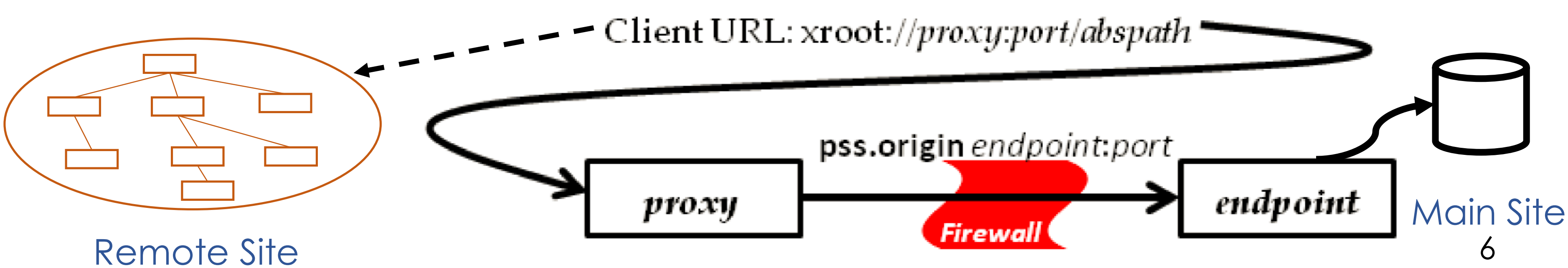
Potential solutions

- Present the same file system view at local and remote sites, then application can run everywhere
- Open/read/write data directly on the remote sites
  - Sufficient network bandwidth
- Mount file system directly in remote site
  - Eg. Lustre , EOS , …
- Xrootd proxy



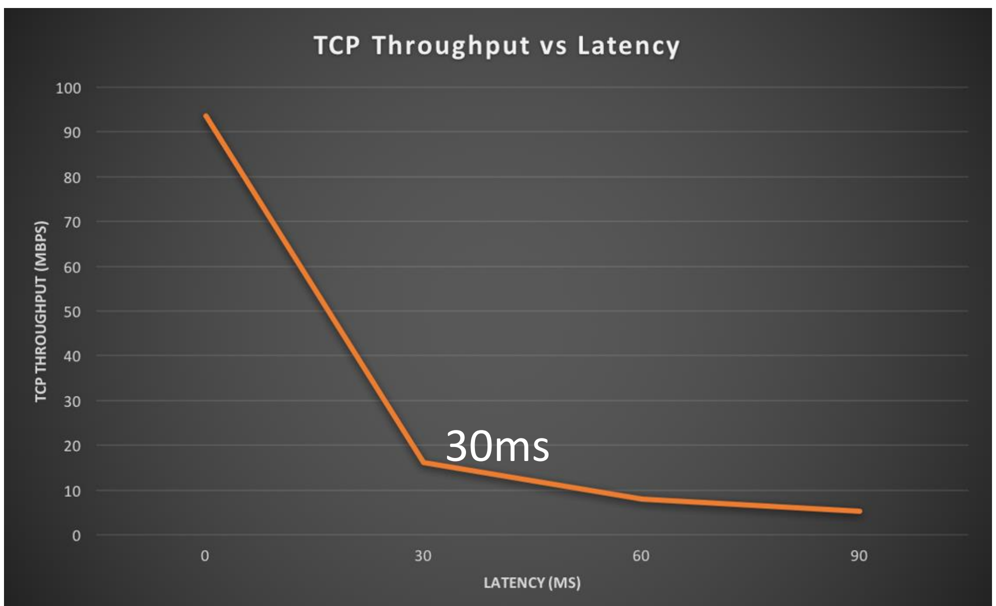
Xrootd proxy

- Standard Proxy Service
  - It is primarily intended to be used for LAN access to bridge firewalls for remote clients
- Disk Caching Proxy
  - Intended to improve WAN access as it is able to cache files or parts of files on disk at a remote location.
  - Caching files or file segments may minimize WAN traffic
- XrootdFS
  - Provides a file system view of an xrootd cluster using FUSE



WAN performance

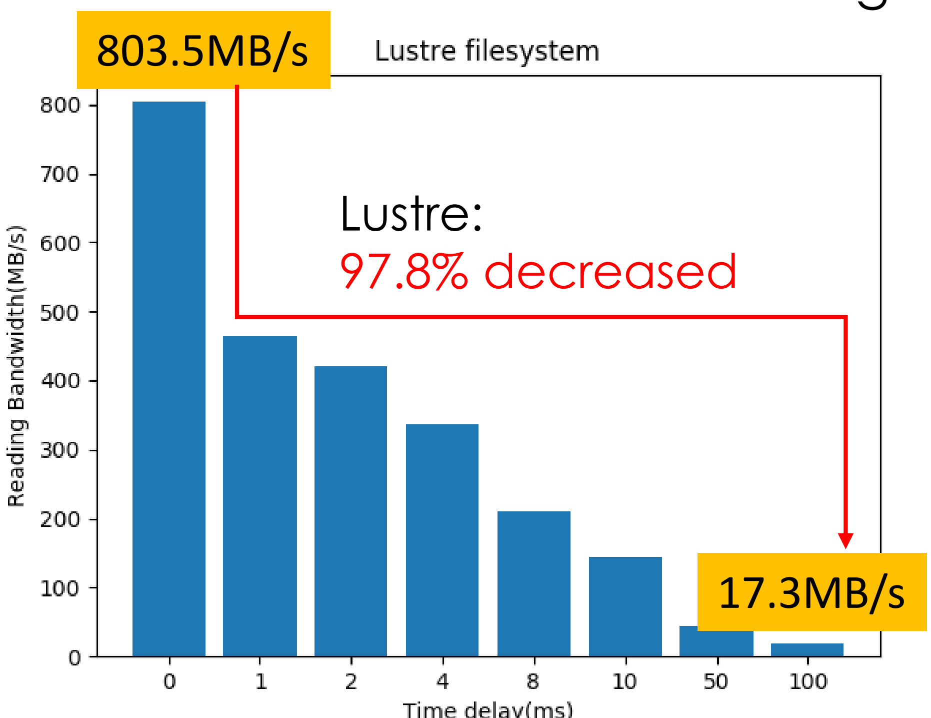
- WAN (Wide Area Network) has three major performance indictors：**Latency、Throughput、Packet loss**
- Throughput is not only decided by bandwidth
- TCP Throughput is directly impacted by latency
  - decreased by 94% in case of 100ms
- TCP Throughput is impacted by packet loss, eg with 2% loss
  - decreased by between 84% and 96%

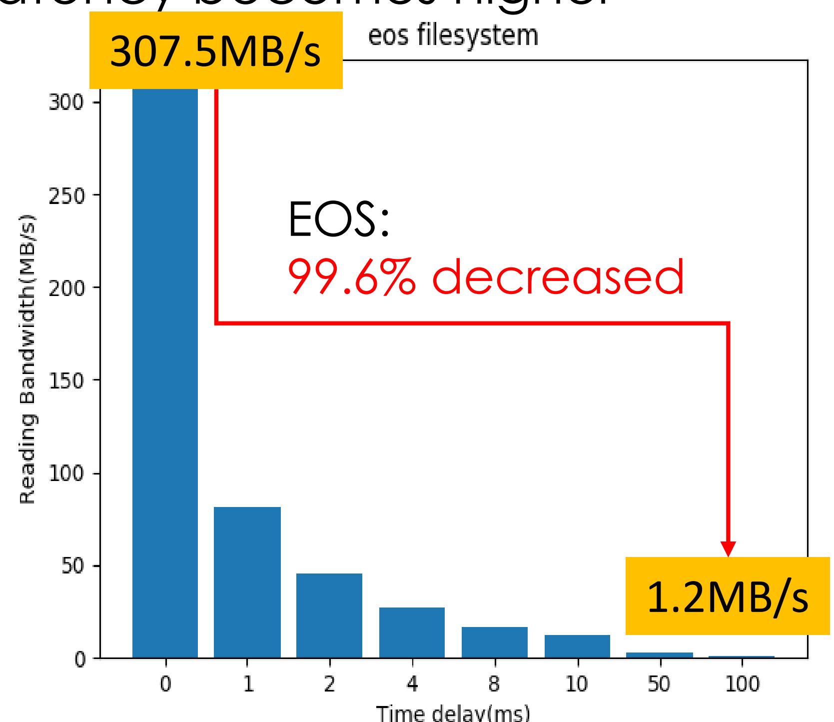


Round trip latency	TCP Throughput with no packet loss	TCP Throughput with 2% packet loss
0 ms	93.5 Mbps	3.72 Mbps
30 ms	16.2 Mbps	1.63 Mbps
60 ms	8.7 Mbps	1.33 Mbps
90 ms	5.32 Mbps	0.85 Mbps

FS Performance Vs Latency

- High latency is inevitable over WAN
  - IHEP <-> CERN: ~350ms ; IHEP<->CCIN2P3: ~270ms
  - IHEP <-> JINR: ~400ms ; Beijing<->Chengdu: ~35ms
- use ‘tc’ to simulate network latency
- Performance is decreasing as latency becomes higher





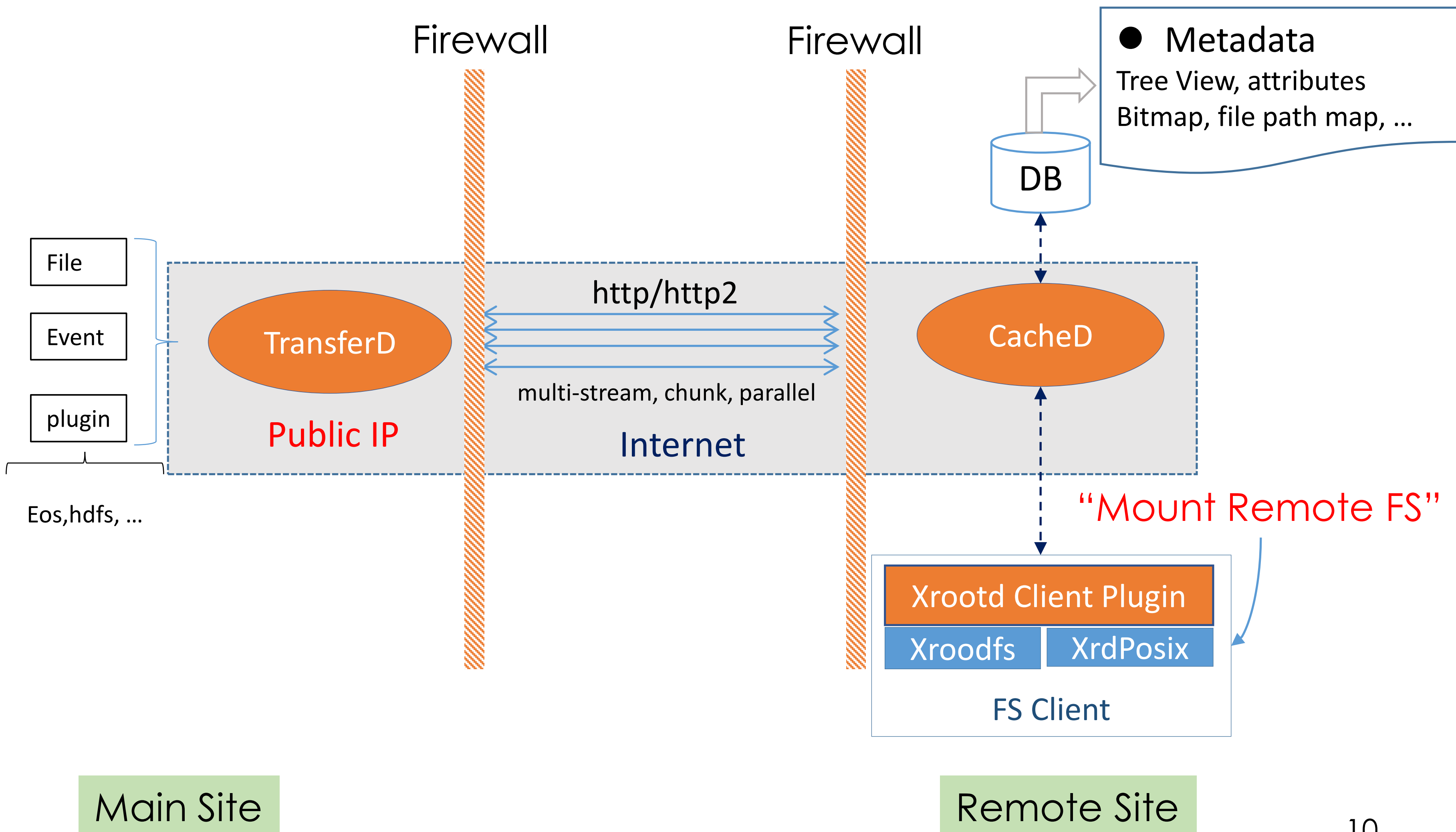
Our vision

- 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

Solution

- 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

Architecture



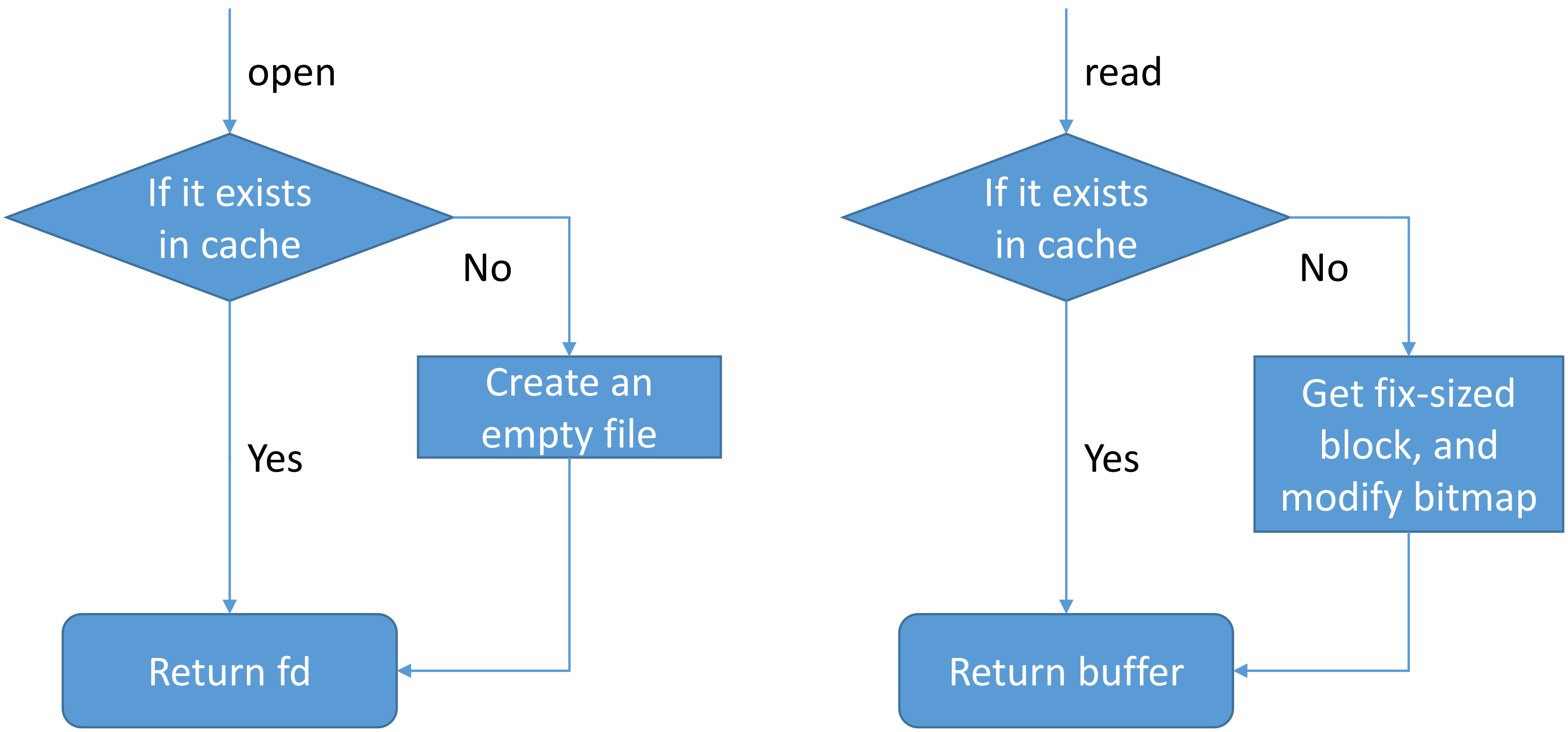
File Transfer Service

- 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

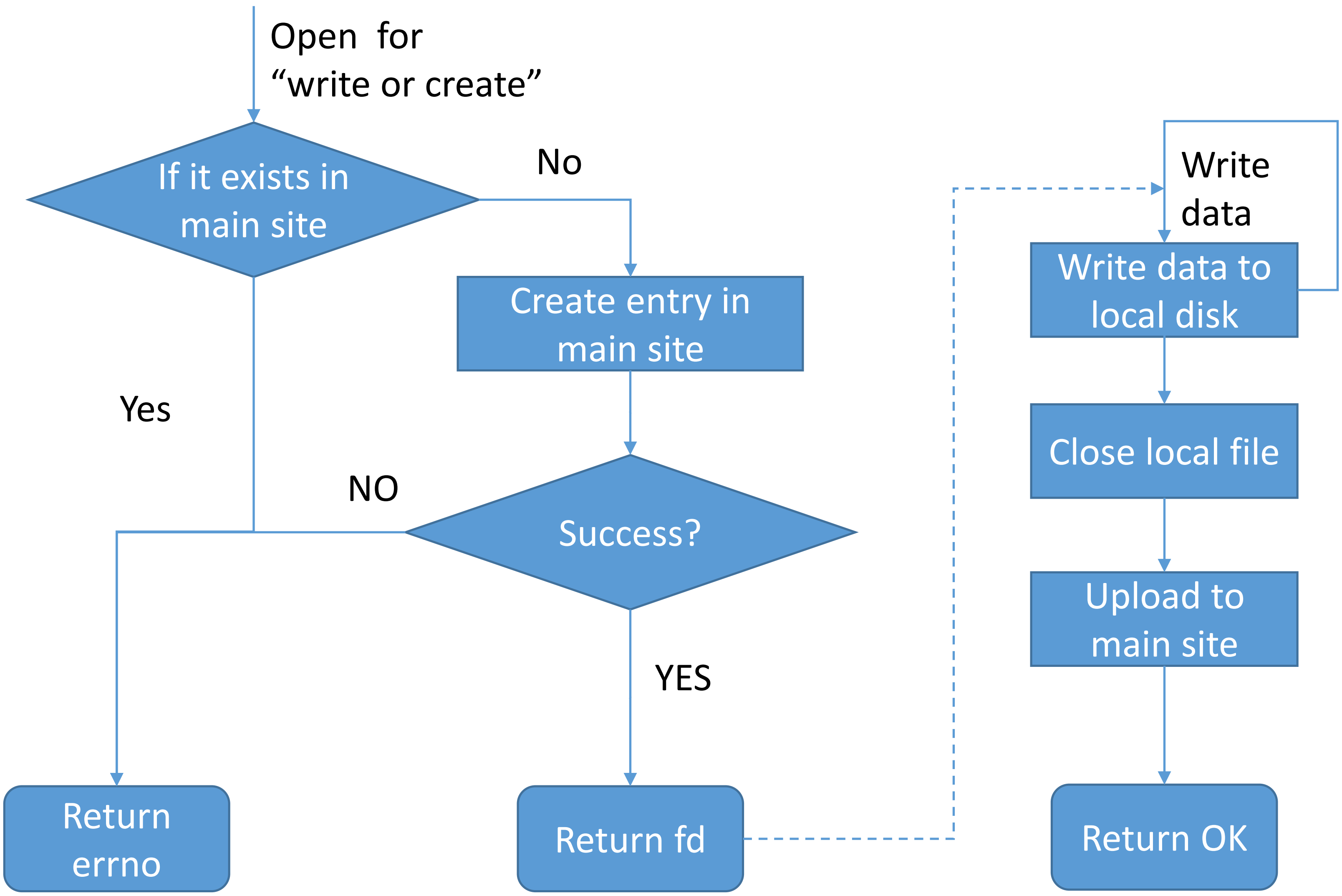
- Three components
  - CacheD: daemon running at remote site
  - DB: store file metadata and bitmap
  - Client tool and library: called by xrootd client plugin
- CacheD will get all entries periodically from main site once the “exported” file system is defined
- DB supports Mysql and Ramcloud currently
- CacheD creates a empty file on local disk once it receives ‘open’ request from client
- CacheD gets fixed-size block (1MB) from offset specified by ‘read’ operation
- CacheD puts the whole file in local disk, then upload it to the main site later in case of ‘write’ operation

Open/read workflow



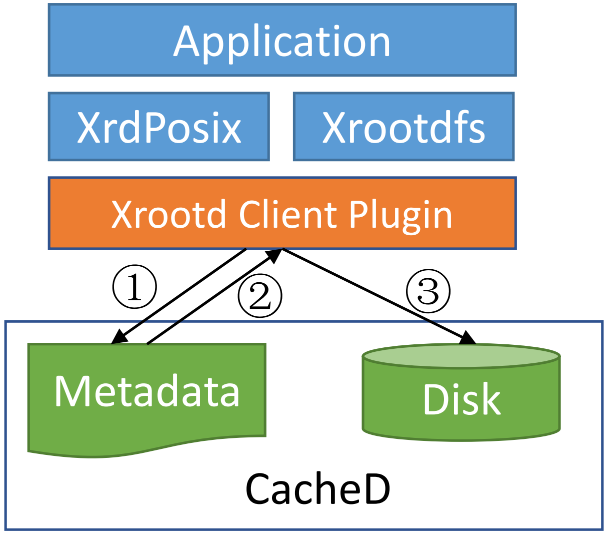
Write workflow

Not allow to modify existing file in remote site



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 call 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

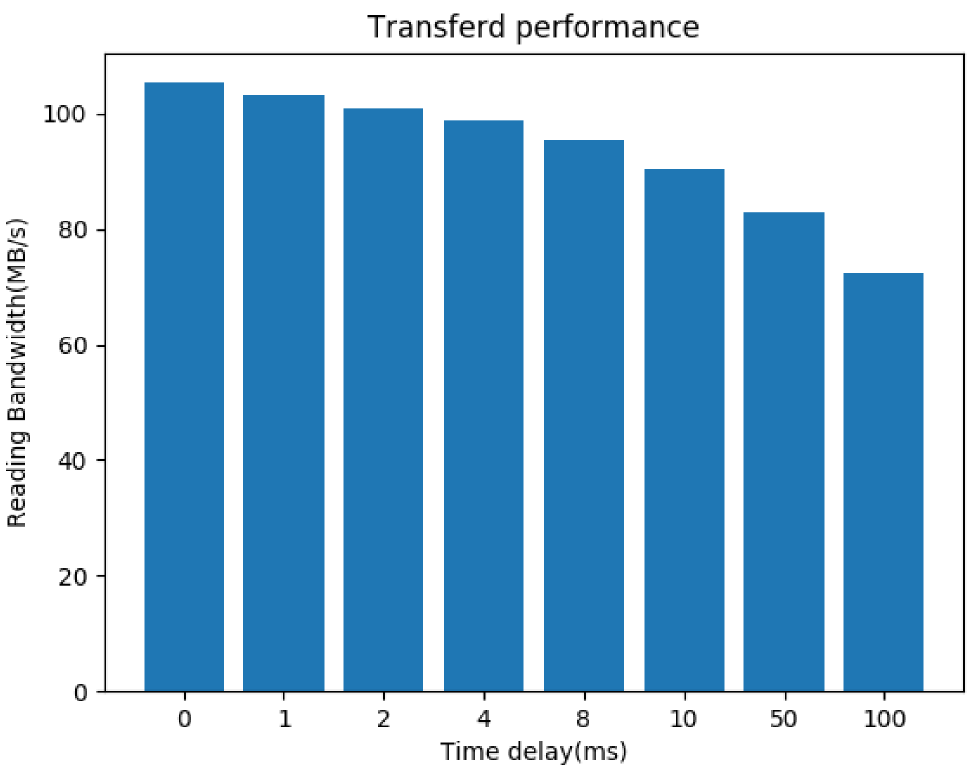
    url = root://cached.domain:1094

    lib = /usr/lib/libXrdLeafClient.so

    enable = true

Performance evaluation

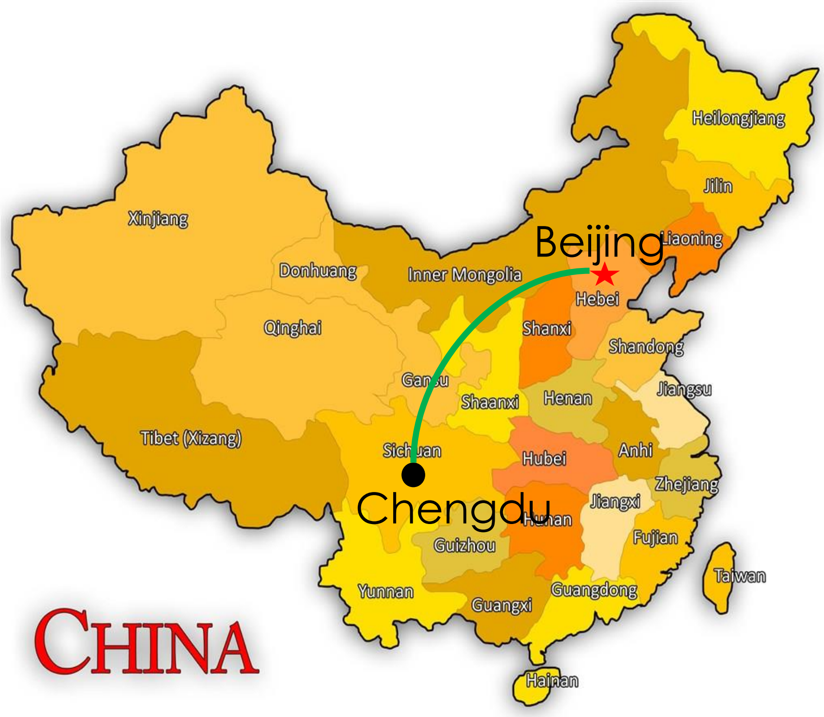
- bandwidth: 1000Mbps
- Latency: 1~100ms using tc simulation
- Transfer parameters: long-lived, 1M block, 10 streams
- Results: decreased by 31% (105MB→72.5MB), better than EOS/Lustre

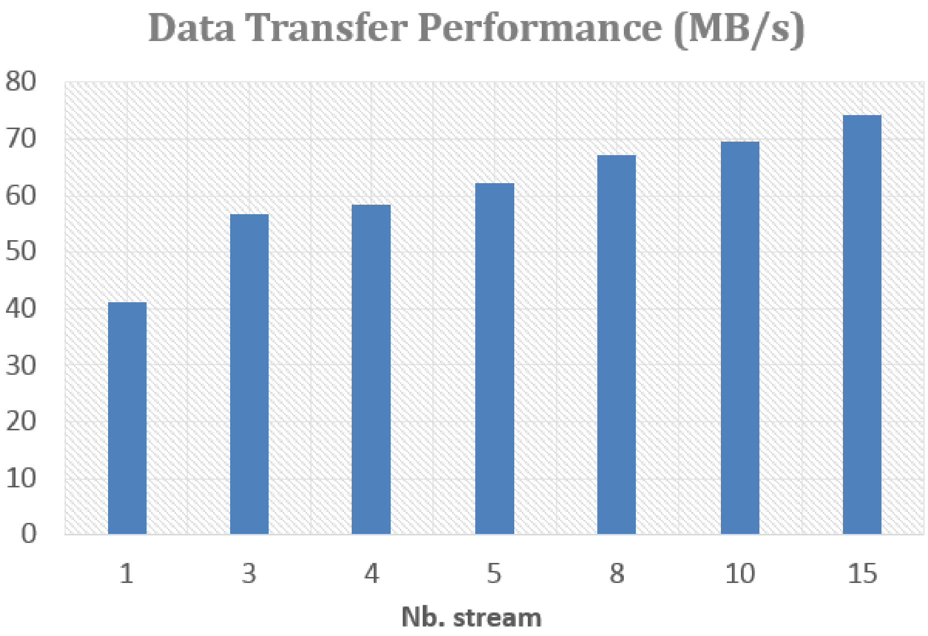


Round trip latency	Transfer performance (MB/sec)
0 ms	105.3
10 ms	90.7
50 ms	82.8
100 ms	72.5

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 BES analysis
  - refer: http://iopscience.iop.org/1742-6596/523/1/012008

    ACAT2013: High performance computing activities in hadron spectroscopy at BESIII

Event transfer workflow

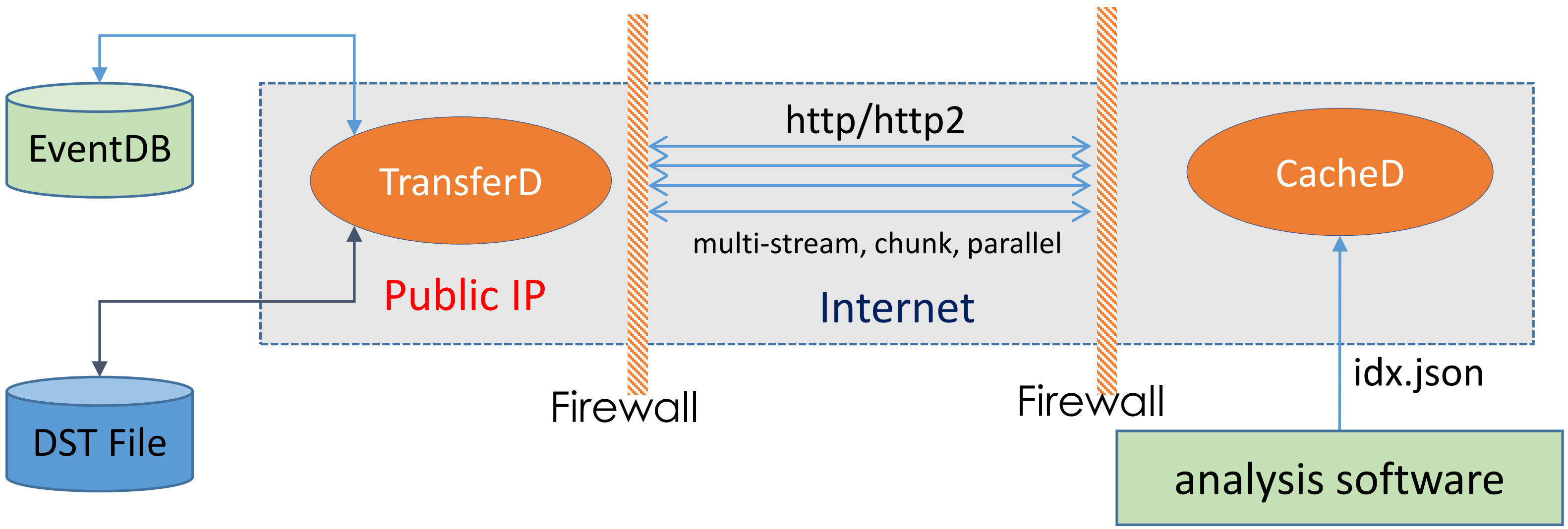
1) Analysis software tells CacheD which events will be used in an analysis, usually giving a json index file

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



Conclusion

- Distributed computing such as grid and cloud is widely used in high energy physics field
- Access data directly from remote site on demand will bring more flexibility
- LEAF is an extension of main storage system, which aims at improving data access performance over WAN
- Implemented as a xrootd plugin supporting most of HEP applications transparently
- Adding new functions, eg HTTP2 support, event-level transfer, etc

Thank you!

Any Questions?