Cross-domain Data Access System for Distributed Sites in HEP

Qi XU (IHEP-CC, Chinese Academy of Sciences)
Outlook

- Data Storage in HEP
- Motivation
- System Architecture
- Performance Test
- Conclusion
Data Storage in HEP

BESIII Experiment Data  $> 100$TB/year

Daya Bay Experiment Data  $> 400$TB in Total

Yangbajing Experiment Data  $> 200$TB/year

LHAASO Experiment Data  $> 2$PB/year
Data Storage in HEP

- Distributed Sites in HEP.
- Large-scale Data Sharing.
- Batch Mode with High Latency Response.
- Computing and Scheduling Based on Files.
Motivation

- Current data sharing mode is hard for unified management.
- A huge consumption of resources: network, storage and CPU.
- Hard to seek and read events based on files, scheduling is not flexible.
- Long time to get response in batch mode, inefficiently.
Motivation

Test 1:
Analysis Based on File System Log
File Size is 478.75MB, Read 22MB, Read Ratio 4.6%
Motivation

Test 2:
Throughput Test of Traditional Distributed File System with Different Latency in WAN

The change of I/O performance with different network delay

Throughput (MB/s) vs Network Delay (ms)
System Architecture

Streaming Transmission & Cache Service in Cross-domain Data Access

Diagram showing the system architecture with components labeled as follows:
- **Client** (C1, C2, C3)
- **Cache**
- **Storage Pool**
- **Metadata**
- **Remote Site**
- **WAN**
- **LAN**

Files labeled as:
- file A
- file B
- ...
- file X
System Architecture

System is Consist of Three Loose Coupling Unix Services

- **CacheD**: Consists of three parts: MetaD, DataD and Daemon Process.

- **TransferD**: XrootdProtocol

- **File Plugin**: Similar to POSIX File System API
System Architecture

The structure of the cross-domain data access system
System Architecture

**MetaD:** Storage and Management for metadata

![Diagram of system architecture]

**Goal**
System Architecture

Fileid Parent_fileid: Uniform File View

<table>
<thead>
<tr>
<th>rowid</th>
<th>fileid</th>
<th>parent_fileid</th>
<th>name</th>
<th>...</th>
<th>mtime</th>
<th>bitmap</th>
</tr>
</thead>
</table>

```
/fileview

local
  daocheng
dayawan

user
  bes
  project
  lhaaso
dbrn

xuq
wangs
offline
data
archive

work
...```
System Architecture

**Bitmap:** On-demand Data Access

```
rowid  fileid  parent_fileid  name  ...  mtime  bitmap
```

```
0  1  0  0  1  1  0  1
```

Data Block Cached
Return to Clients

Not Cached

Get Data by TransferD

Data Block Cached
Return to Clients
System Architecture

Daemon Process: Concurrent Message Process

Diagram:
- Requests
  - Counter
  - Thread Pool
    - Task1
    - Task2
    - Task3
    - ... (continues)
  - Queue
    - TASK1
    - TASK2
    - Task3
    - ...

Do Jobs
System Architecture

DataD: Storage for Cache Files

Original File (Remote Site)
Absolute Path
/dev/storage/xuq/rawdata.01

/dev/disk03/03

Cache Server Parameters
Date Time Clock MAC
uuid_generate
17c055b8-5ffc-433f-9981-4c8598e938f0

/dev/disk03/17c055b8-5ffc-433f-9981-4c8598e938f0

Cache File (Cache Server)
System Architecture

TransferD: Xrootd Protocol

Optimization

- Multiple streams are supported on a single socket.
- Clients can be redirected to another server at any time.
- Clients may be asked to delay server contact.
- Clients may piggy-back read-ahead lists with any read request.
- Servers may ask clients to perform certain actions at any time.
## System Architecture

**File Plugin**: Similar POSIX File System API

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int cdas_open()</td>
<td>Open file (OW/OR)</td>
</tr>
<tr>
<td>int cdas_close()</td>
<td>Close file (Auto)</td>
</tr>
<tr>
<td>int cdas_getattr()</td>
<td>Get metadata from remote site</td>
</tr>
<tr>
<td>int cdas_read()</td>
<td>Read file (Transfer data block, if not cached)</td>
</tr>
<tr>
<td>int cdas_access()</td>
<td>Whether file is accessible</td>
</tr>
<tr>
<td>int cdas_opendir()</td>
<td>Open directory, get DIR_ID</td>
</tr>
<tr>
<td>int cdas_readdir()</td>
<td>Read directory (Metadata of files in it)</td>
</tr>
<tr>
<td>int cdas_rfsync()</td>
<td>Sync files to remote site</td>
</tr>
<tr>
<td>int cdas_refresh()</td>
<td>Update cache files</td>
</tr>
<tr>
<td>int cdas_unlink()</td>
<td>Delete cache files</td>
</tr>
</tbody>
</table>
### Performance Test

**I/O Performance of Lustre EOS and the System are Tested in Bandwidth 1000Mbps**

Read performance (Throughput) of different file system (MB/s)

<table>
<thead>
<tr>
<th>Network Delay (ms)</th>
<th>Lustre</th>
<th>EOS</th>
<th>CDAS (file not cached)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>79.85</td>
<td>30.12</td>
<td>110.10</td>
</tr>
<tr>
<td>1</td>
<td>45.93</td>
<td>7.26</td>
<td>110.25</td>
</tr>
<tr>
<td>10</td>
<td>13.87</td>
<td>1.03</td>
<td>103.13</td>
</tr>
<tr>
<td>100</td>
<td>1.49</td>
<td>0.12</td>
<td>24.46</td>
</tr>
</tbody>
</table>
Performance Test

I/O Performance with Network Delay Jitter is tested in Bandwidth 1000Mbps

Change of I/O Performance in CDAS with Different Network Delay

The change of I/O performance with different network delay
Performance Test

Test of Command Line Interface

- **ilist**: show metadata of files in the remote site.
- **idelete**: delete metadata or data blocks of files in cache.
- **iget**: get data blocks of files from the remote site.
- **iput**: put new files to the remote site.
Performance Test

```
[root@bigdata07 client]# /ilist -l /xrootd
-rw------- 1 root root 104857600 Nov 22 11:21 100ce_ul
-rw------- 1 root root 104857600 Nov 22 11:34 100cs_ul
-rw------- 1 root root 104857600 Jun 07 20:50 100m
-rw------- 1 root root 104857600 Nov 22 20:48 100mb_ul
-rw------- 1 root root 104857600 Nov 18 17:17 100mm
-rw------- 1 root root 104857600 Sep 23 21:28 100upup_ul
-rw------- 1 root root 104857600 Sep 23 21:43 100upupup_ul
-rw------- 1 root root 104857600 Nov 22 13:06 101_ul
-rw------- 1 root root 0 Nov 22 16:54 110_ul
-r------- 1 root root 5 Sep 24 17:20 1726
-rw------- 1 root root 104857600 Sep 24 17:54 1up_ul
-r------- 1 root root 2147483648 Nov 18 15:41 2000m
-r------- 1 root root 209715200 Nov 18 15:43 200m
-rw------- 1 root root 0 Nov 22 20:49 200mb_ul
-rw------- 1 root root 0 Nov 22 13:21 202 Ul
-r------ 1 root root 28 Sep 24 17:27 ceshi
-r------ 1 root root 104857600 Sep 14 13:55 ceshi100
drwx------ 1 root root 64 Nov 06 15:49 dir
-r------- 1 root root 27 Sep 23 21:23 upup_ul

[root@bigdata07 client]# /iget -fvm /xrootd/100m /tmp
DEBUG: Cache file: /cdfs_data/0/429546d7-a073-4a9d-a3fa-3b03db0f401b filesize 104857600
10485760 bytes 67.61 MB/sec avg 67.61 MB/sec instDEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
DEBUG: read 10485760 bytes of size 10485760
104857600 bytes transferred in 0.97 seconds (103.45 MB/s)
```
Conclusion

- Streaming Transmission and Cache Service are adopted to the system.
- The system is consist of TransferD, CacheD and File Plugin.
- Cross-domain data access is localized and transparent for users in the system.
- Data access is on demon and effective in the system.
- Excellent I/O performance of the system with high network latency in WAN.
- Stable I/O performance of the system with Network Delay Jitter.
- Make the most of resources in distributed sites.
- The system is suitable for cross-domain data access between distributed sites.
Thank you for your attention