STORAGE BACKENDS FOR SCALABLE SYNC&SHARE BASED ON SEAFILE

Krzysztof Wadówka, Paweł Woszuk, Maciej Brzeźniak,

Supercomputing Department
AGENDA

- Challenges
- Seafile and it's I/O requirements
- PCSS & PIONIER-BOX: who we are, aims
- Our tests
- Results & observations
CHALLENGES IN DATA MGMT

• DATA STORAGE:
  • growing **volume**: PetaBytes
  • pressure for **performance**: GB/s, IOPS
  • user expectation for **ease of use**

• POWER users:
  • AMU, Faculty of Physics: **550GB @BOX**
    • syncing simulation files, backups,
    • sharing: publications, documents
  • PR Dept, PSNC: **400GB @BOX**
    • sharing photos, movies,
    • serving images to Wordpress-based portal

Source: IDC
WHAT IS SEAFILE?

- **Specialised solution** designed for sync & share

  - **reliable** - data model, synchronisation algorithm

  - **effective** - low-level implementation (C)

- **Backends:**
  - Filesystem, **NFS**, etc.
  - S3, Swift / **Ceph**
FOCUS ON PURPOSE, NOT (TOO) MULTI-FUNCTION

Source: http://www.fastcarinvasion.com/must-see-moment-tractor-crosses-way-racing-car/
SEAFILE SYNC MECHANISM: SNAPSHOT-BASED (NOT PER-FILE VERSIONING)
SEAFILE SYNC MECHANISM:
ONLY DELTAS INCLUDED IN COMMITS,
CONTENT DEFINED CHUNKING ALGORITHM USED FOR DEDUP
### SEAFILE VS OTHERS

#### LARGE FILES*) PERFORMANCE TEST (2016)

<table>
<thead>
<tr>
<th>SPEED</th>
<th>Seafie [GB/s]</th>
<th>theOther [GB/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5x1 GB file upload</td>
<td>0.17</td>
<td>0.11</td>
</tr>
<tr>
<td>5x1 GB file download</td>
<td>0.29</td>
<td>0.71</td>
</tr>
</tbody>
</table>

**LARGE FILES *)**
- 5 GB file
# SEAFILE VS OTHERS

## SMALL FILES*) PERFORMANCE TEST (2016)

<table>
<thead>
<tr>
<th>SPEED</th>
<th>Seafile [files-dirs/s]</th>
<th>theOther [files-dirs/s]</th>
<th>difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client 1: upload</td>
<td>627</td>
<td>27</td>
<td>23x</td>
</tr>
<tr>
<td>Client 2: download:</td>
<td>940</td>
<td>43</td>
<td>22x</td>
</tr>
</tbody>
</table>

**SMALL FILES *)**
- Linux kernel source v. 4.5.3
  - 706 MB of data
  - 52 881 files
  - 3 544 directories

**SEAFILE 5. COMMUNITY, SINGLE 2-CPU SERVER, 120-DISK FC ARRAY, EXT4**
I/O REQUIREMENTS

• Well-optimized synchronization engine & architecture:
  • no large overheads on CPU
  • no high DBMS load:
    • minimum data in the DB (only shares, etc.),
    • metadata in storage backend

• Has potential to stress to I/O system:
  • storing blocks of data and meta-data
  • a nice killer application for the back-end
OUR AIMS

• Provide a **country-wide sync&share service:**
  
  • large user base
  
  • millions of files

• **Exploit the potential** of Seafile

• **Reuse/consolidate storage systems** on the back-end incl.:
  
  • traditional solutions (FC-based disk arrays, cluster filesystems)
  
  • new technologies (object storage etc.)
- POLISH NREN & SERVICES PROVIDER

**PIONIER NETWORK**
- 8000 kms of own fibers
- 3500+ public institutions
- Links to Geant, AMS-X, CERN

**Archival Storage Services:**
- 14+PB of space, 10 DCs
- 300+ client institutions
- Based on „National Data Storage“ software developed in-house

**Cloud computing services:**
- Several 1000s of servers in 21 DCs
- 1000s of users
IN DATA MGMT E-INFRASTRUCTURES

• EUDAT: Collaborative pan-European Data Infrastructure
  • PSNC delivers resources and services: B2SAFE, B2SHARE
  • R&D on object storage federations, HTTP-based federations

• INDIGO-DataCloud: European PaaS-based cloud for e-Science:
  • Participating in work related to extending CDMI protocol/standard with QoS-related mechanisms
  • Providing interfaces to object stores
EAGLE' HPC SYSTEM

- **1.4 PFlops** cluster
- 80th @ TOP500 on Nov 2015
- **33k cores / E5-2697v3**
- **301 TB RAM**
- Infiniband FDR
- **Scratch** on 6PB, 120GB/s **Lustre**
- **Homes** on 5PB, 20GB/s **GPFS**
QUESTION:

Having paid IBM already for GPFS why not use them for sync & share?

Better use Ceph as everybody does ;)

Seafile server

GPFS

NSD client

$$$$

Seafile server

NFS client

Seafile server

libRADOS client

Seafile server

NFS server

RADOS

Seafile server

NFS server

Ceph

GPFS

GPFS

$
<table>
<thead>
<tr>
<th>Seafile:</th>
<th>GPFS back-end:</th>
<th>Ceph back-end:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 1x load-balancer</td>
<td>• 2 GPFS servers</td>
<td>• 8 Ceph servers</td>
</tr>
<tr>
<td>• 2 Seafile servers</td>
<td>• 2 CPUs, 128GB RAM</td>
<td>• 2 CPUs, 128GB RAM</td>
</tr>
<tr>
<td>• Maria DB Galera</td>
<td>• 2x 10GbE, 2x 10GbFC</td>
<td>• 2x10GbE</td>
</tr>
<tr>
<td>• MemcacheD</td>
<td>• GPFS v4.2.2</td>
<td>• Ceph Jewel</td>
</tr>
<tr>
<td>• Storage back-ends:</td>
<td>• Disk array:</td>
<td>• Disks:</td>
</tr>
<tr>
<td>• Ceph</td>
<td>• Huawei OS 5500</td>
<td>• 2x replication</td>
</tr>
<tr>
<td>• GPFS</td>
<td>• 120 HDDs, RAID6</td>
<td>• 12xHDD/server = 96</td>
</tr>
</tbody>
</table>

**Ceph back-end:**

- 8 Ceph servers
  - 2 CPUs, 128GB RAM
  - 2x10GbE
  - Ceph Jewel
  - Disks:
    - 2x replication
    - 12xHDD/server = 96
  - Interface
    - 2x 10GbE + 8x 10GbE
    - LibRADOS
LOAD-BALANCED SETUP

Architecture:

• Load-balancer
• Seafile servers
• Storage back-ends:
  • Memcached
  • MySQL/Maria DB

Architecture scales horizontally

• Seafile application servers work independently
• They share minimum information through memcached
TEST RESULTS
GPFS is faster 1.5-3 times than Ceph for large files.
RESULTS [MB/S]
LARGE FILES TEST (4,4GB FILES)

Seafile tests

Back-end reference tests
RESULTS [FILES/S]
SMALL FILES TEST (45K X 100KB FILES)
RESULTS [FILES/S]
SMALL FILES TEST (45K X 100KB FILES)

Caching effect?

54 GBs in total...
RESULTS [FILES/S]
SMALL FILES TEST (45K X 100KB FILES)

Seafile tests
Back-end reference tests
## SEAFILE VS OTHERS
### SMALL FILES PERFORMANCE TEST (TIME)

<table>
<thead>
<tr>
<th></th>
<th>2016 test</th>
<th>2017, clustered Seafile 100kB files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>single Seafile server, very small files - Linux kernel source</td>
<td></td>
</tr>
<tr>
<td><strong>SPEED</strong></td>
<td><strong>Seafile [files-dirs/s]</strong></td>
<td><strong>theOther [files-dirs/s]</strong></td>
</tr>
<tr>
<td>Client 1: upload</td>
<td>627</td>
<td>27</td>
</tr>
<tr>
<td>Client 2: download</td>
<td>940</td>
<td>43</td>
</tr>
</tbody>
</table>
CONCLUSIONS - SEAFILE

- does not block I/Os to the data and meta-data back-end

- Community version performs 20x-30x faster than the others

- Professional Edition in the clustered setup:
  - has potential to scale to *thousands* IOPS
  - *can use power of object storage: Ceph, Swift, S3…*
    - current implementation: e.g. for Ceph 3k out of 9k IOPS
    - future implementations may even improve the efficiency
CONCLUSIONS: BACK-ENDS FOR SYNC&SHARE

• shall support high level of IO parallelism:
  • object storage has advantage while multithreading:
    • Ceph: despite higher per-thread latency
      better aggregated IOPS - parallelism mitigated latency!
  • GPFS is fast for single thread workload:
    • but keeping the filesystem consistent under high load of many parallel threads is costly
• both can be re-used for sync&share and other applications:
  • Ceph: e.g. objects/S3 + RBD
  • GPFS - homes, Seafile back-end (and even Swift API)
• we will most probably go on with Ceph
Thank you!

THANK YOU ;}

EOF ;)