AliEn development and new catalogue backend

ALICE T1/T2 Workshop - Strasbourg

Miguel Martinez Pedreira
Part I: AliEn development status
Global view

- Databases
  - Catalogue + TaskQueue + Admin + Transfers + IS
    - MySQL: 1 master + 1 or 2 slaves
    - LDAP: 1 master + 2 slaves (push for sync, dns alias)

- 7 central services machines
  - Service instances spread among them

- 12 API servers: 10 for job + 2 for users

- ~80 sites – ~55 storages

- AliEn version: v2-19-395
  - Code updates
  - Packages: httpd, openssl, CAs, JRE, ApMon, cleanup unused
  - Fixes for libreadline issues on new releases
Query caching

- Query analysis – MySQL log
  - select ... from HOSTS where hostIndex=\
  - select ... from GROUPS where where Username=? And PG=\
  - select ... from SITEPROXY where site=\
  - select ... from INDEXTABLE ... (table selection)
  - O(4000) Hz queries, mostly query cached though
  - Global cache + process cache
Query caching

- Query analysis – MySQL log
Query caching
Zombies + cleanup (resubmit)

- Conflicting executions found on some jobs
  - Job goes to ZOMBIE -> resubmits -> starts to run again -> old execution comes back after N hours and changes status

- Pilots modifying job statuses with proxy instead of token!
  - Fixed -> jobid-token sent for change status and traces/procinfo

- SAVING taking specially long lately due to SE being full
  - Adjusted periods to make transition to ZOMBIE
    - 10min for ASSIGNED, 2h for SAVING, 1h for other active status
    - Sending heartbeat (no trace) between file uploads
  - Resubmit
    - extra check to cleanup output or booked lfn's from previous execution
    - killMessage

- ZOMBIEEs can come back and keep running if no resubmission or expiration happened
Optimizers

- OCDB
  - added limits for tarball size/number of files
  - 1 publication per 10 minutes
  - No publication errors/sync lost for many months
  - Stratum propagation delay alerts (also for alice.cern.ch repo) (Dario)

- Splitting
  - ERROR_SPLT without error message: fixed problem retrieving packages info
  - Faster quota calculation
  - More clear error messages, missing whereis trace

- Moving to JAliEn optimizers
  - GuidTable
    - Controls G/G_PFN table sizes and creates new ones
  - IndexTable
    - Controls LFN tables sizes and reports
    - Not doing directory creation automatically

AliEn development - Miguel Martinez Pedreira
CVMFS revision matching

- ERROR_V jobs with “no root/alirroth found” issues or missing library while running

- JobManager adds minimal CVMFS revision needed to the requirements of the jobs, based on packages requested by the jobs -> broker table entry + revision table

- JobAgents check for CVMFS revision and I/O errors
  - attr -qg revision  /cvmfs/alice.cern.ch
  - attr -qg nioerr  /cvmfs/alice.cern.ch
  - Doesn’t work on NFS mounted setups for now
  - Revision sent as parameter to match jobs
  - Reports to ML in case of errors: revision, errors and cvmfs commands dump
  - To add Squid tests

AliEn development - Miguel Martinez Pedreira
<table>
<thead>
<tr>
<th>Site</th>
<th>Node</th>
<th>Revision</th>
<th>Error count</th>
<th>Error message</th>
<th>Timestamp</th>
</tr>
</thead>
</table>
CVMFS revision matching

Mon Apr 24 16:01:58 CEST 2017
wn-1-5-14-a.recas.ba.infn.it
SERVICE STATUS: 1130 I/O errors detected;
offline (http://cvmfs-stratum-one.cern.ch/cvmfs/alice.cern.ch via http://90.147.169.204:3128);
offline (http://cvmfs-stratum-one.cern.ch/cvmfs/alice.cern.ch via http://pccms26.ba.infn.it:3128);
offline (http://cvmfs-stratum-one.cern.ch/cvmfs/alice.cern.ch via http://cofin2003.ba.infn.it:3128);
offline (http://cernvmfs.gridpp.rl.ac.uk/cvmfs/alice.cern.ch via http://90.147.169.204:3128);
offline (http://cernvmfs.gridpp.rl.ac.uk/cvmfs/alice.cern.ch via http://pccms26.ba.infn.it:3128);
offline (http://cernvmfs.gridpp.rl.ac.uk/cvmfs/alice.cern.ch via http://cofin2003.ba.infn.it:3128);
offline (http://cvmfs.racf.bnl.gov/cvmfs/alice.cern.ch via http://90.147.169.204:3128);
offline (http://cvmfs.fnal.gov/cvmfs/alice.cern.ch via http://90.147.169.204:3128);
offline (http://cvmfs.fnal.gov/cvmfs/alice.cern.ch via http://pccms26.ba.infn.it:3128);
repository revision 2189
Mon Apr 24 16:03:52 CEST 2017
More fixes/updates

- **JobAgent**
  - Catalogue connection: reuse and/or retry
    - avoid ERROR_SV
  - Fix procInfo information (i.e. process longer than 24h or forks)
  - procInfo headers in the trace
  - JOBLIMIT

- **Util**: process cache tunable time to live

- **top -node <hostname>**

- Fix `get` for zip members on SEs with non standard savedir

- File Quota optimized (queries)
More fixes/updates

- Batch interfaces (Pavlo)
  - ARC: system command timeouts (information retrieval), commands and files used cleanups
  - HTCONDOR, OCCI: new interfaces

- CE config now also allows to block users to run

- moveDirectory: -d option, skip delete

- xrdstat output dump + PFN url when failed

- Parse domains correctly everywhere (‘.’ at the end)
Traceability tests

- Traceability/Isolation WG looking for pilot-pilot and pilot-payload isolation solution
  - Singularity

- Evaluation of security/access models

- Challenge exercise: experiments must find job/s running in a node at a given time-range and give information about payload/files used or other information
  - Created a internal procedure for such cases
    - Easy to follow and documented
    - 1- DB checks
    - 2- Trace processing
    - [3- Find submission host for user jobs (api logs)]
    - ~2h for several hours time-range

AliEn development - Miguel Martinez Pedreira
Part I – Questions?

- **To-Do**
  - As usual, plenty of things to add and improve
    - Hard file quotas
    - Run CPU benchmarks in JobAgent
    - Optimize number of baskets when splitting
    - JDLs masterjob-subjob
    - Query optimization / cache
    - …
  
- **JAliEn prioritized**

- **Jobs record**
  - 102K in T1T2 Bergen
  - 136K in T1T2 Strasbourg 😊
Part II: new catalogue backend
Current implementation

- MySQL-based AliEn File Catalogue
  - 3B logical entries
- One (powerful) DB master
  - 1.5TB RAM, 2.4TB on disk size
- DB slaves for hot standby / backups
  - 4h to dump, ~2 days to restore
Catalogue in a nutshell

- **LFN namespace**
  - `/alice/data/2016/LHC16n/000261088/raw/16000261088034.205.root`
  - 1180 tables (max 50M), 3B entries, namespace split into tables
  - Metadata
    - `-rwxr-xr-x alidaq alidaq 264403565 Sep 09 22:10 0f24bce32446ea22840d188e035b11a9`

- **GUID namespace**
  - `76CEBD12-76A0-11E6-9717-0D38A10ABEEF`
  - 173 tables (max 210M), 2.8B entries, split by time intervals (append)
  - Version 1 UUIDs (MAC+timestamp)

- **Physical File Pointers**
  - `root://alice-tape-se.gridka.de:1094//10/33903/76cebd12-76a0-11e6-9717-0d38a10abeef`
  - `root://voalice10.cern.ch//castor/cern.ch/.../16000261088034.205.root`
  - 3.5B entries, 1B physical files, pointers to ZIP members, 70PB over 70 Storage Elements
Catalogue in a nutshell

- **LFN namespace**
  - `/alice/data/2016/LHC16n/000261088/raw/16000261088034.205.root`
  - 1180 tables (max 50M), 3B entries, namespace split into tables
  - Metadata
  - `rwxr-xr-x` alidaq alidaq 264403565 Sep 09 22:10 0f24bce3246ea22840d188e035b11a

- **GUID namespace**
  - `76CEBD12-76A0-11E6-9717-0D38A10ABEEF`
  - 173 tables (max 210M), 2.8B entries, split by time intervals
  - Version 1 UUIDs (MAC+timestamp)

- **Physical File Pointers**
  - `root://alice-tape-se.gridka.de:1094/10/33903/76cebd12-76a0-11e6-9717-0d38a10abeef`
  - `root://voalice10.cern.ch//castor/cern.ch//.../16000261088034.205.root`
  - 3.5B entries, 1B physical files, pointers to ZIP members, 70PB over 70 Storage Elements
DB query rates

- Averages (1y)
  - 9223 Hz Reads
  - 618 Hz Changes
  - 282 Hz Deletes

- 77500 running jobs

- 15:1 select/change ratio

- 10:1 read/write data volume
The AliEn catalogue in time

Number of rows in the AliEn Catalogue namespaces

AliEn Catalogue - Miguel Martinez Pedreira
Future needs

- In Run3 we will have 5x more computing resources (300K CPUs + 5000 GPUs)
- 10x more disk and tape storage => ~10x more files to manage
- The goal is to sustain ~200kHz queries (stable)
  ~1Mhz queries (peaks)
- Numbers are a bit inflated
  - Query cache represents half or more of the select
  - Many of those queries won’t be needed:
    - New backend schemas simplify (next slides)
    - Improvements on the framework
    - Preparing file envelopes for jobs at split
    - More aggressive caching in JAliEn

- Looking for a solution providing:
  - Horizontal scaling
  - No single point of failure
  - High query rate
  - HA
Apache Cassandra

- Provides all the requirements mentioned before:
  - Horizontal scale
  - Add nodes to keep up ops/s
  - HA – No point of failure
  - Performance (see later initial benchmarks)

- Consistency
  - Tunable levels, key factor for us

- We move from N to few tables for the namespace
  - Simplification

- Easy setup

- Mapping certain SQL operations not trivial
  - Groupings, quota calculations, ‘where’ possibilities…
  - NoSQL re-implementation, CQL helps too
### First schema in C*

<table>
<thead>
<tr>
<th>EntryID</th>
<th>JobID</th>
<th>Checksum</th>
<th>Metadata</th>
<th>URLs</th>
<th>(Path)</th>
<th>Child</th>
<th>Ctime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PS: some tables will stay in MySQL

**AliEn Catalogue** - Miguel Martinez Pedreira
Cassandra benchmarks

- Setup a 5-node ring
  - Server power: 16-48 cores, 100-350GB RAM
  - Java 8 Oracle, no swap, nofile/memlock limits (no degraded mode)
  - Mapped namespace into a column family that is able to do `whereis` and `ls`: entry contains lfn+pfns metadata
    - Starting a new round of benchmarking on a implementation that allows `find` as well
  - RF 3, LeveledCompaction + LZ4 compression

- Data dump
  - MySQL to Cassandra -> slow
  - Artificial lfns and dirs -> very quick!

- Execution
  - Java sized thread pool, configure hierarchies, number of LFNS, etc…
Cassandra benchmarks

- Initial benchmarking shows promising results

Time to retrieve logical and physical information of a file

Operations per second based on number of clients
New schema in C*

LFN index

(ParentID)
FileName
ChangeTime

EntryID
Flag

LFN metadata

ParentId
(EntryID)

Owner
Group
Type
Size
Permissions
JobID
Checksum
Metadata<string,string>
PFNs<int,string>
Flag

SE lookups

(SENumber moduloID)
EntryID

Size
Owner

PK
((SENumber, hash(EntryID) % n ), EntryID)
New vs old schema

**Cons**
- Need to loop over the lfn_index (hierarchy) to do file operations
- To avoid contention on the servers and thus latency:
  - Can be cached by client
  - (We hope) can be cache by Cassandra -> RowCache

**Pros**
- Some complex and heavy operations become much easier
  - mv dir: delete old parent and insert new
  - rm dir: mark/delete parent
- Easy to keep a trash bin
Monitoring

- Cassandra internally calculates and exposes an ample set of metrics -> MBean (JMX)

- Naming is misleading and documentation is scarce

- Pluggable to some extended tools

- ML will have a dashboard to have a global view of the cluster and detect problems -> link to C* monitor

- Feeding most important metrics:
  - Read/Write latencies and throughput
  - KeyCache hits/requests
  - Compaction+GC stat
  - Timeouts, Unavailable, Exceptions
  - CF stats
  - Usual machine status: load, cpu/memory/network usage…
Stress test: read

Read performance

Read operations / second

<table>
<thead>
<tr>
<th>Series</th>
<th>Last Value</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. aliendb06f.cern.ch</td>
<td>13569</td>
<td>12047</td>
<td>12778</td>
<td>13701</td>
</tr>
<tr>
<td>2. aliendb06g.cern.ch</td>
<td>12567</td>
<td>11081</td>
<td>11784</td>
<td>12777</td>
</tr>
<tr>
<td>3. aliendtest02.cern.ch</td>
<td>13737</td>
<td>13236</td>
<td>11876</td>
<td>12789</td>
</tr>
<tr>
<td>4. aliendtest03.cern.ch</td>
<td>12333</td>
<td>10950</td>
<td>11500</td>
<td>12415</td>
</tr>
<tr>
<td>5. aliendtest04.cern.ch</td>
<td>13563</td>
<td>12002</td>
<td>12705</td>
<td>13825</td>
</tr>
<tr>
<td>6. aliendtest05.cern.ch</td>
<td>13215</td>
<td>11679</td>
<td>12445</td>
<td>13299</td>
</tr>
<tr>
<td>7. aliendtest07.cern.ch</td>
<td>13095</td>
<td>12238</td>
<td>13039</td>
<td>14106</td>
</tr>
<tr>
<td>Total</td>
<td>92014</td>
<td>86179</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average latency (us)

<table>
<thead>
<tr>
<th>Series</th>
<th>Last Value</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. aliendb06f.cern.ch</td>
<td>2394</td>
<td>2060</td>
<td>2426</td>
<td>3108</td>
</tr>
<tr>
<td>2. aliendb06g.cern.ch</td>
<td>1974</td>
<td>1864</td>
<td>2176</td>
<td>2534</td>
</tr>
<tr>
<td>3. aliendtest02.cern.ch</td>
<td>12609</td>
<td>1836</td>
<td>2276</td>
<td>2698</td>
</tr>
<tr>
<td>4. aliendtest03.cern.ch</td>
<td>1384</td>
<td>7785</td>
<td>8023</td>
<td>9571</td>
</tr>
<tr>
<td>5. aliendtest04.cern.ch</td>
<td>2349</td>
<td>2242</td>
<td>3008</td>
<td>4582</td>
</tr>
<tr>
<td>6. aliendtest05.cern.ch</td>
<td>2692</td>
<td>1969</td>
<td>3023</td>
<td>3845</td>
</tr>
<tr>
<td>7. aliendtest07.cern.ch</td>
<td>4049</td>
<td>3767</td>
<td>4041</td>
<td>4670</td>
</tr>
<tr>
<td>Total</td>
<td>23914</td>
<td>26176</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stress test: read

Read performance

Machines status

<table>
<thead>
<tr>
<th>Machine</th>
<th>Online</th>
<th>Uptime</th>
<th>Load</th>
<th>Kernel</th>
<th>OS</th>
<th>Machine model</th>
<th>CPU</th>
<th>CPUs</th>
<th>MHz</th>
<th>Disk</th>
<th>Space</th>
<th>CPU utilisation (%)</th>
<th>Memory utilisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. db6f</td>
<td>220:04</td>
<td>18.13</td>
<td>4.40</td>
<td>0-65</td>
<td>16.04</td>
<td>ProLiant DL380 Gen9</td>
<td>Xeon E5-2687w v4 3.00GHz</td>
<td>48</td>
<td>3199</td>
<td></td>
<td></td>
<td></td>
<td>22.13</td>
</tr>
<tr>
<td>3. alientest02</td>
<td>21:23</td>
<td>28.8</td>
<td>4.40</td>
<td>0-97</td>
<td>16.04</td>
<td>ProLiant DL380 Gen8</td>
<td>Xeon E5-2690 v2 2.00GHz</td>
<td>48</td>
<td>3000</td>
<td></td>
<td></td>
<td></td>
<td>30.72</td>
</tr>
<tr>
<td>4. alientest03</td>
<td>17:58</td>
<td>62.26</td>
<td>4.40</td>
<td>0-99</td>
<td>16.04</td>
<td>ProLiant DL380 Gen8</td>
<td>Xeon X5650 2.60GHz</td>
<td>16</td>
<td>2794</td>
<td></td>
<td></td>
<td></td>
<td>56.35</td>
</tr>
<tr>
<td>5. alientest05</td>
<td>1:53</td>
<td>37.64</td>
<td>4.40</td>
<td>0-67</td>
<td>16.04</td>
<td>ProLiant DL380 Gen8</td>
<td>Xeon E5-2697 v2 2.70GHz</td>
<td>48</td>
<td>2699</td>
<td></td>
<td></td>
<td></td>
<td>34.74</td>
</tr>
<tr>
<td>6. alientest06</td>
<td>21:24</td>
<td>59.49</td>
<td>4.40</td>
<td>0-97</td>
<td>15.04</td>
<td>ProLiant DL380 Gen8</td>
<td>Xeon E5-2697 v2 2.70GHz</td>
<td>48</td>
<td>2999</td>
<td></td>
<td></td>
<td></td>
<td>54.8</td>
</tr>
<tr>
<td>7. alientest07</td>
<td>21:26</td>
<td>61.48</td>
<td>4.40</td>
<td>0-97</td>
<td>16.04</td>
<td>ProLiant DL380 Gen6</td>
<td>Xeon X5650 2.60GHz</td>
<td>16</td>
<td>2794</td>
<td></td>
<td></td>
<td></td>
<td>48.34</td>
</tr>
</tbody>
</table>

Total 54d 3:59 40.31 256 40.18 11.56 1.683 0 2.629 0 3.4 40.55 352 GB 74.82 GB 2.548 GB 170.3 GB
Stress test: read

KeyCache hits / second

29 Mar 2017
CEST time
Stress test: write

Write performance

<table>
<thead>
<tr>
<th>Write operations / second</th>
<th>Series</th>
<th>Last value</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>allendb08f.cern.ch</td>
<td>15301</td>
<td>12150</td>
<td>14985</td>
<td>15571</td>
</tr>
<tr>
<td></td>
<td>allendb08g.cern.ch</td>
<td>14243</td>
<td>11282</td>
<td>13934</td>
<td>14425</td>
</tr>
<tr>
<td></td>
<td>alientest02.cern.ch</td>
<td>14106</td>
<td>11193</td>
<td>13951</td>
<td>14336</td>
</tr>
<tr>
<td></td>
<td>alientest03.cern.ch</td>
<td>13814</td>
<td>10900</td>
<td>13377</td>
<td>14029</td>
</tr>
<tr>
<td></td>
<td>alientest05.cern.ch</td>
<td>15230</td>
<td>12097</td>
<td>14901</td>
<td>15471</td>
</tr>
<tr>
<td></td>
<td>alientest06.cern.ch</td>
<td>14858</td>
<td>11751</td>
<td>14417</td>
<td>15103</td>
</tr>
<tr>
<td></td>
<td>alientest07.cern.ch</td>
<td>15625</td>
<td>12342</td>
<td>15203</td>
<td>15923</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>103241</td>
<td>100171</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average latency (us)</th>
<th>Series</th>
<th>Last value</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>allendb08f.cern.ch</td>
<td>1348</td>
<td>570.4</td>
<td>1140</td>
<td>1380</td>
</tr>
<tr>
<td></td>
<td>allendb08g.cern.ch</td>
<td>1321</td>
<td>617.2</td>
<td>1117</td>
<td>1338</td>
</tr>
<tr>
<td></td>
<td>alientest02.cern.ch</td>
<td>1763</td>
<td>942.8</td>
<td>1525</td>
<td>1824</td>
</tr>
<tr>
<td></td>
<td>alientest03.cern.ch</td>
<td>5063</td>
<td>1135</td>
<td>4378</td>
<td>5625</td>
</tr>
<tr>
<td></td>
<td>alientest05.cern.ch</td>
<td>1927</td>
<td>678.7</td>
<td>1489</td>
<td>2030</td>
</tr>
<tr>
<td></td>
<td>alientest06.cern.ch</td>
<td>1619</td>
<td>840.8</td>
<td>1321</td>
<td>1703</td>
</tr>
<tr>
<td></td>
<td>alientest07.cern.ch</td>
<td>5340</td>
<td>1391</td>
<td>5126</td>
<td>5748</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>18414</td>
<td>15900</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stress test: write
Stress test: write no compression
Next steps

- Tune Cassandra
  - Investigate and optimize CPU usage
  - Tune JVM+GC, RowCache/KeyCache sizes…
- Run benchmarks with `cassandra-stress`
  - Comparable to similar clusters of the community?
- Get closer to a production workload
  - Mixed set of select/update/insert/delete as in the current catalogue
- Exercise critical operations
  - Backups
  - Addition/replacement of nodes
3D Crosspoint

- Biggest memory breakthrough in 25 years

- Not very clear yet how it will work, but provides:
  - Higher data volume than RAM
  - Low latency
  - $\frac{1}{2}$ price RAM?

- Persistent RAM
  - Remove slow I/O layers -> In-memory DB?
  - Booking area…
Thanks

Questions?
B slides - CPU debug: cpu sample
B slides - CPU debug: GC
B slides - CPU debug: jstack info

"(ML ThP) [ util.process ] Worker 250448, started: Wed Mar 29 16:57:08 CEST 2017" #1096365 daemon prio=5 os_prio=0 tid=0x00007f50680c0000 nid=0x1547 waiting on condition 0x00007f50632f1000
java.lang.Thread.State: TIMED_WAITING (parking)
at sun.misc.Unsafe.park(Native Method)
  - parking to wait for <0x00000000c8002870> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2078)
at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.poll(ScheduledThreadPoolExecutor.java:1129)
at java.util.concurrent.ScheduledThreadPoolExecutor$Worker.runTask(ScheduledThreadPoolExecutor.java:889)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1127)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
ScyllaDB

- “Next generation Cassandra”

- Difference relies on core implementation

Full report here