REVIEW

Main development topics
• Inner logic module
• Storage API

Testing
• NoSQL (Cassandra) vs. ATLAS PanDA Archive (Oracle)
HYBRID STORAGE MODEL
Part 1: INTERNAL LOGIC
INTERNAL LOGIC

Storage management inner tasks

- **Synchronization**: provide consistency of SQL and NoSQL parts
- **Inner synchronization**: provide inner data consistency within NoSQL
- [clean SQL from archive records]

**Synchronization main points**

- detect unsynchronized data
- synchronize data
- do not affect the whole system performance
SYNCHRONIZATION

- Step 1. Simple copy.
  Works for tests, not good at all for real life.

- Step 2. Criteria:
  - modification time
  - not copied yet (row count)

- Step 3. Policy:
  - when
  - how much
CRITERIA

✓ Modification time
  Copy data from Date1 to Date2
  Date1: 2014-01-27
  Date2: 2014-01-29 (today)

➢ Not copied yet?
  Nobody would like to check every job, right?
  So check time interval: day or N hrs

Service tables in NoSQL:
  • [datetime – number of jobs]

---

To Date(modification time) | count
--- | ---
2015-01-26 | ...
✓ 2014-01-27 | 1096887
➢ 2014-01-28 | 2210772
➢ 2014-01-29 | 75862

Date | count | state
--- | --- | ---
2015-01-26 | ... | Ok
2014-01-27 | 1096887 | Ok
2014-01-28 | 1023529 | ?
2014-01-29 | 0 | ?

29/01/2015
○ POLICY

• When?
  Daily.

• How much?
  If something went wrong, we don’t want to synchronize too much at a time
  - jobs limit: 5M
  - days limit: 3 days
INNER (NOSQL) SYNCHRONIZATION

☑ Step 1. Simple copy.
   From main table (Jobs) into dependent tables.

☑ Step 2. Criteria:
   • modification time
   • inner consistency broken (count)

☐ Step 3. Policy:
   • when
   • how much
CRITERIA

- **Modification time**
  Consistency interval: day.

- **Inner consistency**
  Service table in NoSQL

<table>
<thead>
<tr>
<th>Table</th>
<th>Date</th>
<th>Key</th>
<th>Value</th>
<th>Count</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jobs</td>
<td>2015-01-27</td>
<td>TaskId</td>
<td>1516868</td>
<td>201</td>
<td>-</td>
</tr>
<tr>
<td>Task</td>
<td>2015-01-27</td>
<td>TaskId</td>
<td>1516868</td>
<td>201</td>
<td>Ok</td>
</tr>
<tr>
<td>Jobs</td>
<td>2015-01-27</td>
<td>TaskId</td>
<td>1516869</td>
<td>723</td>
<td>-</td>
</tr>
<tr>
<td>Task</td>
<td>2015-01-27</td>
<td>TaskId</td>
<td>1516869</td>
<td>512</td>
<td>?</td>
</tr>
<tr>
<td>Jobs</td>
<td>2015-01-27</td>
<td>Cloud</td>
<td>UK</td>
<td>692</td>
<td>-</td>
</tr>
<tr>
<td>Cloud</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

To be fixed
- POLICY (inner synchronization)

  - **When?**
    - *initial synchronization*: with writes into main table
    - *service data update and check*: daily
    - *fixing synchronization*: manual

  - **How much?**
    - *jobs limit*: 10M
    - *days limit*: 3 days
Part 2: STORAGE API
SQL ONLY

SQL

Application

Application

Application

Standard Drivers:
Django Oracle Engine
Django MySQL Engine
cx_Oracle

...
API MODULE

Clients

Application

Application

Application

API

NoSQL

Custom API

Standard driver

NoSQL DB

SQL

Custom API

Standard driver

SQL DB

29/01/2015
Disadvantages of unified API
- System gets less flexible
  (every request is to be implemented in advance)

Advantages
+ Isolation:
  + no need to make an explicit request to every DB (or to determine destination DB by request parameters)
  + no changes in application code if storage was switched from one DB engine to another (e.g. from Cassandra to HBase or from Oracle to MySQL)
API MODULE

General idea

1. Receive requests from external applications
2. Determine location of the requested data (SQL or NoSQL)
3. Transfer the request to a proper database
4. Get a result
5. Return the result
QUERY DISTRIBUTION

send_request

NoSQL
Storage.noSQL
Storage.noSQL.cassandra
  Cassandra driver
Storage.noSQL.mongo
  Mongo driver...

SQL
Storage.SQL

the way it was in application originally

Django Oracle Engine
Django MySQL Engine
  cx_Oracle

29/01/2015
Part 3: TESTS
# Cassandra vs. Oracle

<table>
<thead>
<tr>
<th>Test</th>
<th>Cassandra</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job by PandalID</td>
<td>0.007 s</td>
<td>0.097 s</td>
</tr>
<tr>
<td>Jobs by TaskID</td>
<td>0.012 s</td>
<td>0.063 s</td>
</tr>
<tr>
<td>Jobs by JediTaskID</td>
<td>0.004 s</td>
<td>0.065 s</td>
</tr>
<tr>
<td>Jobs by TaskID with given JobStatus within given interval of ModificationTime</td>
<td>0.008 s</td>
<td>0.069 s</td>
</tr>
<tr>
<td>Jobs by TaskID within given interval of ModificationTime</td>
<td>0.023 s</td>
<td>0.065 s</td>
</tr>
</tbody>
</table>

Measured time is a **query execution time** (without data transfer); Each request was send **100 times** with different query parameters; Presented in table value is a **mean value** for these 100 requests;

---

**Oracle**: 1 year archive; partitions.  
**Cassandra**: 2 weeks archive; request-specific tables.

29/01/2015
THANK YOU
storage.api.requests – determined set of functions:
• JobByIID(caller, parameters, fields, SQLOnly=True)
• TaskJobList(caller, parameters, fields, SQLOnly=True)
• ...

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>caller</td>
<td>&quot;monitor&quot;</td>
</tr>
<tr>
<td>parameters</td>
<td>&quot;{&quot;pandaid&quot;: &lt;pandaid&gt;, ...}&quot;</td>
</tr>
<tr>
<td>fields</td>
<td>[&lt;field1&gt;, &lt;field2&gt;, ...]</td>
</tr>
<tr>
<td>SQLOnly</td>
<td>True</td>
</tr>
</tbody>
</table>
storage.api.routing - to determine SQL or NoSQL:
  • routing(caller, parameters, SQLOnly=False)

Caller:
  • "server" → SQL
  • "monitor" → depends on parameters

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModificationTime</td>
<td>&lt; (today – 3 days)</td>
<td>SQL</td>
</tr>
<tr>
<td>JobStatus</td>
<td>not in &lt;FINISHED&gt;</td>
<td>SQL</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
API: TRANSFER

`storage.api.send_request`

- `send_request(destination, request_type, parameters, fields, waitAll=False)`

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>`storage.SQLdb</td>
</tr>
<tr>
<td>request_type</td>
<td>`JobById</td>
</tr>
<tr>
<td>parameters</td>
<td><code>{&quot;pandaid&quot; = &lt;pandaid&gt;, ...}</code></td>
</tr>
<tr>
<td>fields</td>
<td><code>[&lt;field1&gt;, &lt;field2&gt;, ...]</code></td>
</tr>
<tr>
<td>waitAll</td>
<td>`False</td>
</tr>
</tbody>
</table>