DDM Accounting

Mario Lassnig

European Organization for Nuclear Research (CERN)

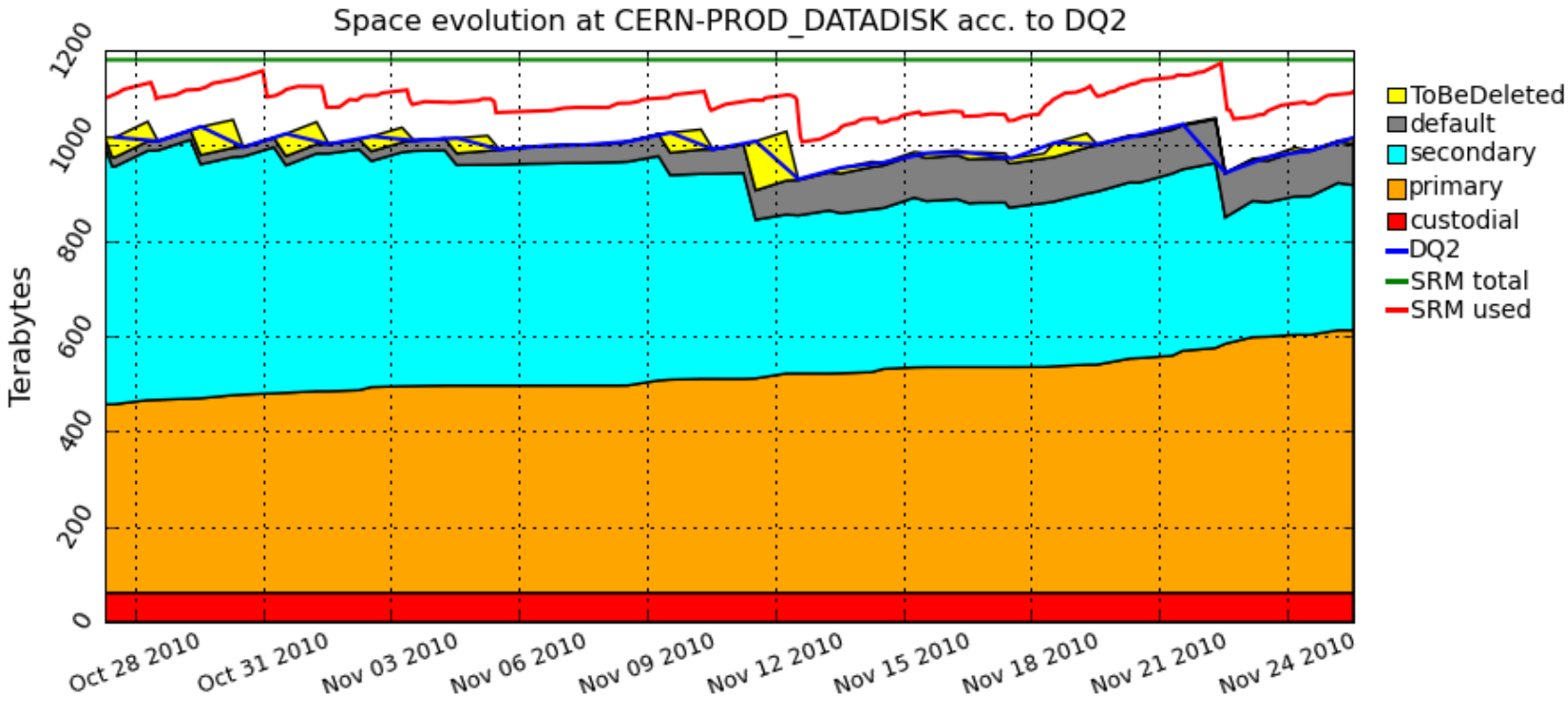
ADC lab

ATLAS Software & Computing Workshop

Outline

1. Current accounting system
2. New accounting system
3. Evaluation
4. Conclusions

Current accounting system



1. Query SRM endpoints for file information (SRM Agent)
2. Organise accounting data in Oracle (DQ2 Agent)
3. Shiny plots created with data retrieved from Oracle (Web)

Current accounting system

- Recently started meta-accounting of the web interface
- Well-established and popular service (unique hits per day)

Where's the problem?

- Continuous requests for new accounting modes
- Requires changes in database schema (i.e. a column per mode)
- Low cardinality of attributes makes indexes useless and results in slow query responses

Architectural change

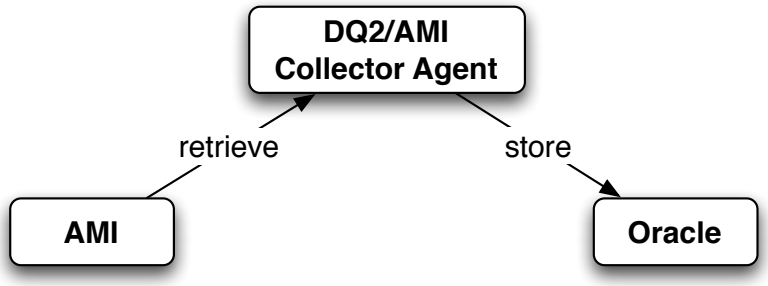
Switch from DSN accounting to metadata accounting

- **Old**: instead of DSN+Location with wildcards
  - "data10.*.ESD.*" + "CERN"
- **New**: query a collection of key:value metadata
  - {'project':'data10','type':'ESD', 'location':'CERN', ...}

Requirements

1. Proper mapping of datasets to metadata
2. New connector service (client, server, backend)
3. Store the history of wanted key:value collections

Proper mapping of datasets to metadata



t_dsnconventions (IOT)

REGEX	CONVENTION
^group09\..*$	group.groupName.[otherFields]
^mc10_7TeV\..*$	project.datasetNumber.....
. . .	

Dataset naming conventions

1. Regularly retrieve metadata descriptions from AMI (Collector)
2. Organise components into regular expression lookup-table
3. Populate the search index with the components

Changes to datasets

- Never update the search index, only insert or delete!
- Transfer deleted entries into history table with timestamp.

Search index

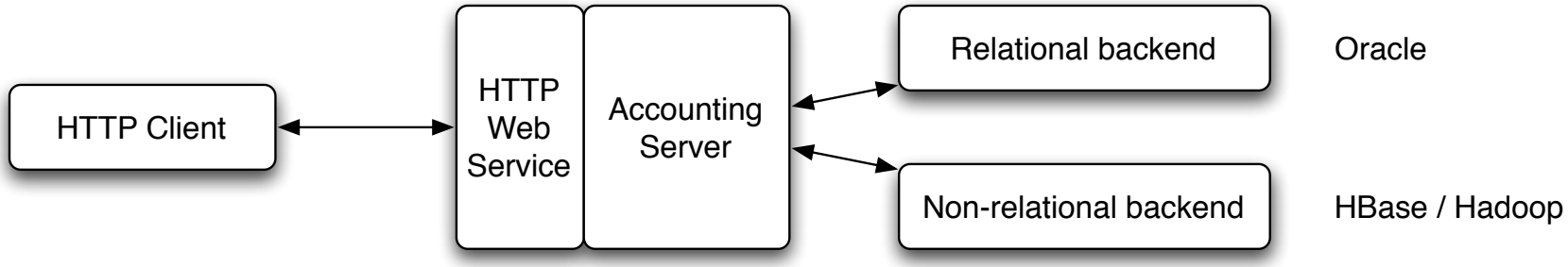
t_kvs (IOT)

DUID	KEY	VALUE	CREATIONDATE
BINARY(16)	project	data08_cosm7	2008-05-31 18:05:28
BINARY(16)	runNumber	00071384	2008-05-31 18:05:28
BINARY(16)	streamName	physics_HLT_Cosmics_NIM0	2008-05-31 18:05:28
BINARY(16)	[otherFields]	Pythia_MSSMPREJUDICE_946_susy	2009-04-17 11:55:05
BINARY(16)	user	user09	2009-04-17 11:55:05
BINARY(16)	userName	phys-susy	2009-04-17 11:55:05
BINARY(16)	[otherFields]	0914133330.764183.lib._000450	2010-09-14 13:33:32
BINARY(16)	user	user	2010-09-14 13:33:32
BINARY(16)	userName	sbehar	2010-09-14 13:33:32
BINARY(16)	[otherFields]	0915155144.954507.lib._003708	2010-09-15 15:51:47
...	...	...	...

Querying the search index

- Constant time lookup to find the required data (regardless the size of the query collection)
- New accounting modes are new rows, no schema changes
- Query the history: just include timing metadata {'fromDate':<timestamp>, 'toDate':<timestamp>}

Accessing the data



New connector service

1. Uses common DQ2 framework
   - HTTP web service
   - HTTP clients
2. Service only has one API call
   - **getUsage** ({key_1:value$_1$, key_2:value$_2$, . . . , key_n:value$_n$} returns a solution dictionary {duid: [#files, #replicas, . . .]}
3. Two independent backends
   - Oracle
   - HBase

Oracle

Design

1. **Objective**: constant time key:value lookup
   - Want to retrieve DUIDs as fast as possible
   - Index-organised: fast lookup
   - Only inner joins: constant-time filter
2. Ignore traditional database principles
   - Normalisation
   - Type safety
   - Redundancy

Key-Value selections

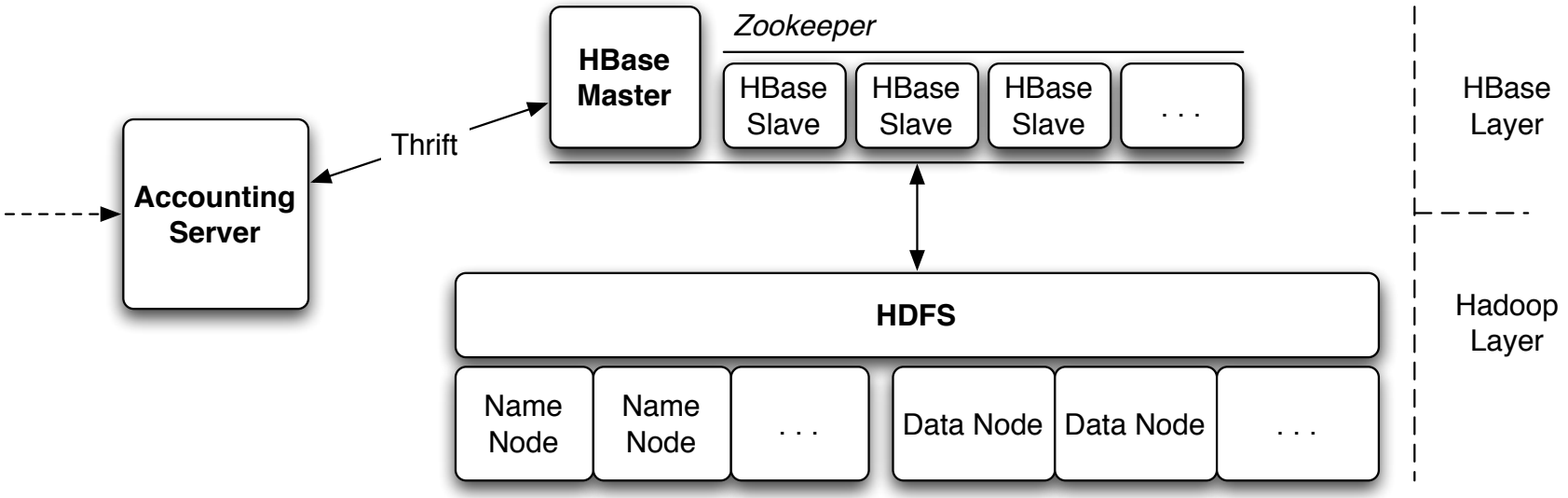
- Why not use dedicated technology tailored for this use case?
- HBase: *distributed key:value store*

HBase

What is it?

- HBase is a *free software*, *non-relational*, *distributed* database, running on a *Hadoop HDFS* filesystem
  - Fault-tolerant way of storing large volumes of sparse data
  - Billions of rows *x* millions of columns on commodity hardware
- In production use by Adobe, Yahoo, Facebook, Microsoft, Twitter, . . .
  - Lots of development manpower
- Cherry-picked advantages
  - No single point of failure
  - Add and remove machines dynamically without shutting down the whole cluster

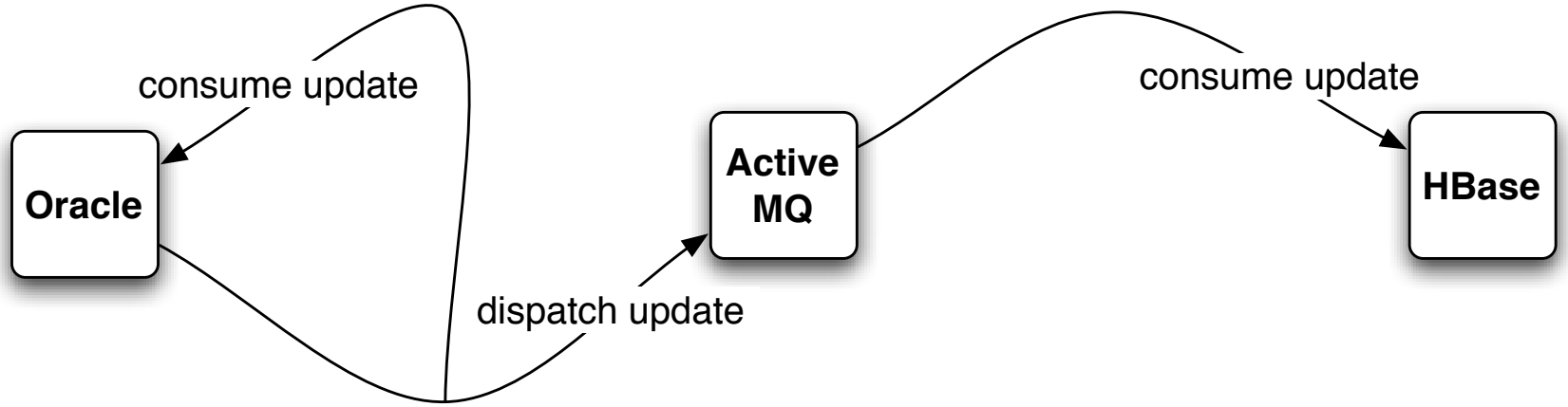
HBase backend



Architecture

- Apache Thrift is a high-performance TCP service API
- Zookeeper keeps the nodes in consistent master/slave states
- HDFS distributes the files in hierarchical namespaces
  - Minimum replication is enforced with eventual consistency
  - Linear scalability across data nodes

Evaluation



Keep the backends synchronised

1. Oracle trigger dispatches change event notification and updates the key:value store tables
   - move old row into *history* table
   - insert new value in *current* table
2. ActiveMQ queues the notification
3. HBase consumes the event notification
   (direct changes in HBase will not be propagated to Oracle)

Evaluation

Zero response round-trip time

Facebook's Thrift way faster cx_Oracle's connection pools

Current accounting server:	$0.0182s \pm 0.0012$
New accounting service (Oracle):	$0.0192s \pm 0.0023$
New accounting service (HBase):	$0.0007s \pm 0.00001$

Querying *data10 ESD* datasets

```
./dq2-accounting-usage project data10% dataType ESD
```

Current accounting server:	$3180s \pm 282$
New accounting service (Oracle):	$84s \pm 5.4$
New accounting service (HBase):	data not migrated yet, demonstrator (1/20th of the data volume) yields $2.2s \pm 1.2$

Conclusions

Summary

- Current accounting system is working well for now
  - But continuous requests for more detailed accounting
  - Current architecture not suitable
- Change the architecture into a metadata accounting service
  - Free-form access to accounting data
  - (Quasi) Real-time!
  - Constant-time lookup, linear time retrieval
    - fast index scan over sorted rows
- Two backends provided, Oracle and HBase
  - Independent but mutual failovers
  - Oracle interface is primary because of existing support
  - Investigate HBase-style model in a real-world use case

Timeline

Oracle

- Mid-December 2010: Implementation ready
- Mid-January 2011: In production

HBase

- Mid-December 2010: Prototype ready
- Mid-January 2011: Implementation ready, Testbed cluster
- Mid-February 2011: Production cluster, In production

More information

- http://bourricot.cern.ch/dq2/accounting/global_view/30/
- http://thrift.apache.org/
- http://hbase.apache.org/