STRUCTURED STORAGE IN ATLAS DISTRIBUTED DATA MANAGEMENT

Mario Lassnig, Vincent Garonne, Angelos Molfetas,
Thomas Beermann, Gancho Dimitrov, Luca Canali, Donal Zang,
on behalf of the ATLAS Collaboration
mario.lassnig@cern.ch
ph-adp-ddm-lab@cern.ch

CHEP'12, New York City, NY

Overview

- Structured storage
  - Concepts
  - Technologies
- ATLAS DDM use cases
  - Storage facility
  - Data-intensive analytics
- Operational experiences
  - Software
  - Hardware
- Conclusions

Structured Storage :: Concepts

- Is this about NoSQL? Yes, but…
  - NoSQL is a buzzword term to annoy RDBMS people
  - Correct CS term: *(distributed) structured storage*
  - Many products support SQL or SQL-derivatives anyway

- So what is NoSQL, pardon, structured storage about?
  - 1. ***Non-relational modelling and storage of data***
    - Use the native data layout of an application
  - 2. ***Linear scalability of data processing***
    - Scalability ≠ Performance

- **Performance:** Capability of a system to provide a certain response time
  - e.g., *generate a valid analysis of a sample within three seconds*

- **Scalability:** Dependency characteristics between resources and performance
  - e.g., *maintain the three seconds when the number of samples increase*

Structured Storage :: Concepts

- Relational database management systems
  - Vertical scalability ("scale up")
  - Few powerful nodes
  - Shared state
  - Explicit partitioning
  - Resistant hardware
  - ACID
  - Implicit queries *(WHAT)*

- Structured storage
  - Horizontal scalability ("scale out")
  - Lots of interconnected low cost nodes
  - Shared nothing architecture
  - Implicit partitioning
  - Reliability in software
  - BASE
  - Explicit data pipeline *(HOW)*

Structured Storage :: Concepts

- Relational database management systems
  - Vertical scalability ("scale up")
  - Few powerful nodes
  - Shared state
  - Explicit partitioning
  - Resistant hardware
  - ACID
  - Implicit queries *(WHAT)*

- Structured storage
  - Horizontal scalability ("scale out")
  - Lots of interconnected low cost nodes
  - Shared nothing architecture
  - Implicit partitioning
  - Reliability in software
  - BASE
  - Explicit data pipeline *(HOW)*

Main problems addressed:

1. There is an upper limit of processing power you can put in a single node
2. Explicit partitioning can be cumbersome
3. Relaxation of ACID properties can be necessary
4. Query plans need information about the data contents

Structured Storage :: Technologies

- Three technologies evaluated
  - **MongoDB** (*10gen, Inc.*)
  - **Cassandra** (*Apache Software Foundation*, formerly *Facebook*)
  - **Hadoop with HBase** (*Apache Software Foundation*, formerly *Yahoo*)
- Many more available, but these were chosen with the following things in mind
  - Large community available and widely installed
  - In production use at several larger companies with respectable data sizes
  - Potential commercial support
- 12 node cluster to evaluate technologies
  - Nodes located in CERN IT data centre
- Nodes managed by Puppet
  - Data centre automation framework
  - Implicit service and configuration definition
  - One-button push update on all nodes

	Cluster configuration
Nodes	12
Architecture	Linux x86_64
CPU Cores	96 (Intel Xeon 2.26 GHz, 8/node)
RAM	288 GB (24/node)
Storage	–
Storage Network	–
Disk	24 SATA (1TB each, 2/node)
Cache	–
Network	1 GigE

Structured Storage :: Technologies :: hadoop

- Hadoop is framework for distributed data processing
  - It is not a database like MongoDB or Cassandra

- Many components
  - **HDFS**: distributed filesystem
  - **MapReduce**: distributed processing of large data sets
  - **HBase**: distributed data base for structured storage
  - **Hive**: SQL frontend and warehouse
  - **Pig**: data-flow language for parallel execution
  - **ZooKeeper**: coordination service
  - … many more

Structured Storage :: Technologies :: Data Models

mongoDB

```
{
 _id: 'Main Account User',
 groups: ['group_a',
          'group_b',
          'group_c'],
 selections: {
              'select_a': 123,
              'select_b': abc
             }
}
```

- Explicit row-key
- Native datatypes
- Everything indexable

Cassandra

```
'Groups' : {
    'Main Account User' : {
        'groups' : ['group_a',
                    'group_b',
                    'group_c']
    }
}

'Selections' : {
    'Main Account User' : {
        'select_a' : 123,
        'select_b' : abc
    }
}
```

- Implicit row-keys
- Data is byte streams
- Column Families group row-keys

APACHE HBASE

```
'Main Account User' : {
    'Groups' : {
        'groups' : ['group_a',
                    'group_b',
                    'group_c']
    },
    'Selections' : {
        'select_a' : 123,
        'select_b' : abc
    }
}
```

- Implicit row-key
- Data is byte streams
- Row-keys group Column Families
- Row-keys are sorted

Structured Storage :: Technologies :: Data Bases

mongoDB

- Master/Slave
  - Smart client implements failover
- Write-ahead log
- Limited MapReduce
  - interleaved
  - bound to single thread
- Keyed binary storage
- Indexes
- Table locking
- Replica sets
- Explicit partitioning

Cassandra

- No single point of failure
  - ring of nodes
  - forwarding of requests
- Write-ahead log
- No MapReduce
  - can use Hadoop
- No file storage
- Bloom filter
- Row locking
- Snapshotting
- Implicit partitioning

APACHE HBASE

- No single point of failure
  - multiple masters
- Write-ahead log
- MapReduce
- File storage
  - Data on HDFS
  - Can be used as a source and sink within Hadoop
- Bloom filter
- Row locking
- HDFS-backed redundancy
- Implicit partitioning

Structured Storage :: Technology Selection

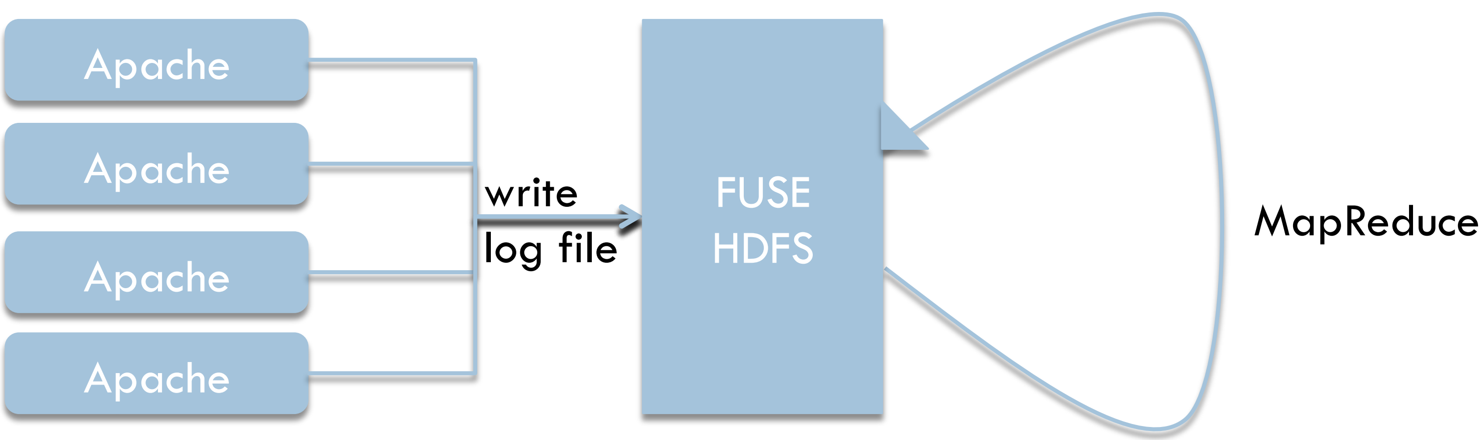
	MongoDB	Cassandra	Hadoop/HBase
Installation/ Configuration	Download, unpack, run	Download, unpack, configure, run	Distribution, Complex config
Buffered read 256	250'000/sec	180'000/sec	150'000/sec
Random read 256	20'000/sec	20'000/sec	20'000/sec
Relaxed write 256	10'000/sec	19'000/sec	9'000/sec
Durable Write 256	2'500/sec	9'000/sec	6'000/sec
Analytics	Limited MapReduce	Hadoop MapReduce	MapReduce, Pig, Hive
Durability support	Full	Full	Full
Native API	Binary JSON	Java	Java
Generic API	None	Thrift	Thrift, REST

Structured Storage :: Technology Selection

	MongoDB	Cassandra	Hadoop/HBase
Installation/ Configuration	Download, unpack, run	Download, unpack, configure, run	Distribution, Complex config
Buffered read 256	250'000/sec	180'000/sec	150'000/sec
Random read 256	20'000/sec	20'000/sec	20'000/sec
Relaxed write 256	10'000/sec	19'000/sec	9'000/sec
Durable Write 256	2'500/sec	9'000/sec	6'000/sec
Analytics	Limited MapReduce	Hadoop MapReduce	MapReduce, Pig, Hive
Durability support	Full	Full	Full
Native API	Binary JSON	Java	Java
Generic API	None	Thrift	Thrift, REST

Use cases :: Log file aggregation

- HDFS is mounted as a POSIX filesystem via FUSE
  - Daily copies of all the ATLAS DDM log files are aggregated in a single place
  - 8 months of logs accumulated, already using 3 TB of space on HDFS
- Python MapReduce jobs analyse the log files
  - Streaming API: read from stdin, write to stdout
- Processing the data takes about 70 minutes
  - Average IO at 70MB/s
  - Potential for 15% performance increase if re-written in pure Java
    - Better read patterns and reducing temporary network usage



Use cases :: Trace mining

- Client interaction with ATLAS DDM generates traces
  - E.g., downloading a dataset/file from a remote site
  - Lots of information (25 attributes), time-based
  - One month of traces uncompressed 80GB, compressed 25GB
    - Can be mapreduced in under 2 minutes
- Implemented in HBase as distributed atomic counters
  - Previously developed in Cassandra
  - At various granularities (minutes, hours, days)
  - Size of HBase tables negligible
  - Average rate at 300 insertions/s
- Migrated from Cassandra within 2 days
  - Almost the same column-based data model
  - Get extra Hadoop benefits for free (mature ecosystem with many tools)
  - The single Cassandra benefit, HA, was implemented in Hadoop recently

Use cases :: DQ2Share

- HTTP cache for dataset/file downloads
  - Downloads via ATLAS DDM tools to HDFS, serves via Apache
  - Get all the features of HDFS for free, i.e., one large reliable disk pool

DQ2Share

ddo.000001.Atlas.Ideal.DBRelease.v0701010 Get dataset and files

Search results: **ddo.000001.Atlas.Ideal.DBRelease.v07010104** Displaying 20 on 84 files

Type	Name	Size of all files (Bytes)	Number of files	Date	In the Cache
Dataset	ddo.000001.Atlas.Ideal.DBRelease.v07010104	1 754 037 441	84	2009-07-02 02:18:11	.tar: ✗

☐ wget 19%...	Type	Name	Size (Bytes)	Checksum	In the Cache
☐	File	07010104._0121364.tar.gz	12 446 534	ad:b05542bb	✗
☐	File	07010104._0120760.tar.gz	12 183 186	ad:8bd6b6d3	✓
☐	File	07010104._0121457.tar.gz 100%	13 654 310	ad:2aee4fcc	✓
☐	File	07010104._0120884.tar.gz	11 927 416	ad:654e6578	✓
☐	File	07010104._0121416.tar.gz	18 179 853	ad:ed7f5d07	✗
☐	File	07010104._0121064.tar.gz	12 591 388	ad:a5464922	✓
☐	File	07010104._0120808.tar.gz	12 181 040	ad:203c2a15	✓
☑	File	07010104._0121198.tar.gz	13 648 670	ad:1afc7773	✓
☑	File	DBRelease-7.1.1.4.tar.gz	693 005 997	ad:3db1645e	✓
☑	File	07010104._0120713.tar.gz	11 958 239	ad:fa3b23a6	✗
☐	File	07010104._0120852.tar.gz	12 246 473	ad:a0d72443	✓
☐	File	07010104._0121412.tar.gz	12 455 380	ad:32a180f0	✗
☐	File	07010104._0121226.tar.gz	12 559 921	ad:d06936f5	✗
☐	File	07010104._0121414.tar.gz	12 461 831	ad:b8e44e10	✗

Use cases :: Wildcard search

- List contents of ATLAS DDM based on a pattern
  - e.g., all data11 datasets (query: *data11**)
  - RDBMS: *Index range scan (~2 seconds, in memory)*
- This becomes more expensive on sub-selections
  - e.g., all data11 datasets with a RAW datatype (query: *data11*RAW**)
  - RDBMS: *Index full scan (~10 seconds, in memory)*
- And worst if only later parts of the pattern are used
  - e.g., all datasets with a RAW datatype (query: *RAW*)
  - RDBMS: *Full table scan (~30 seconds in memory, ~60 seconds on disk)*

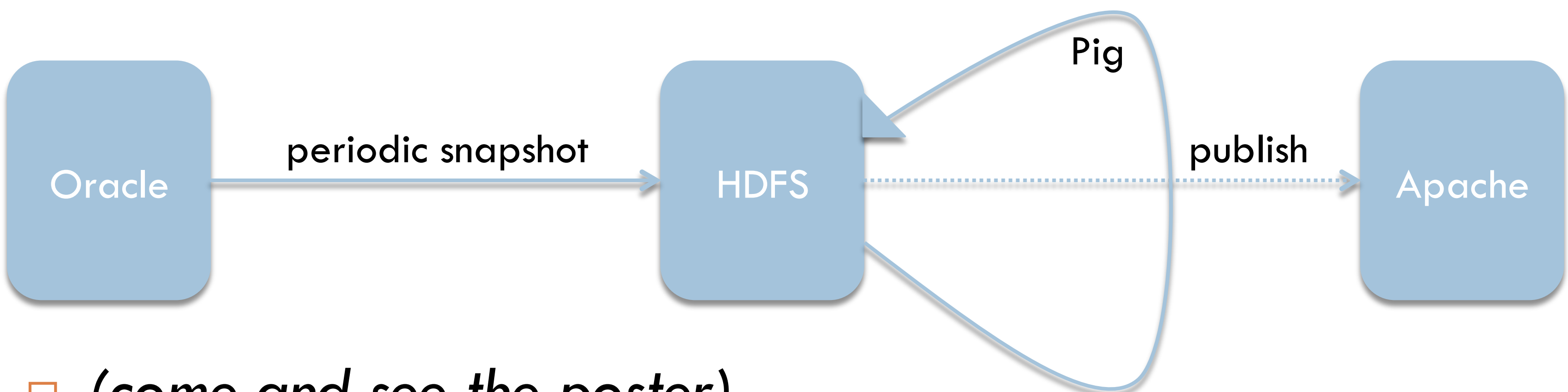
- Asynchronous wildcard search in Hadoop HDFS
  - Periodic dump of the necessary columns from RDBMS to a flat file
  - MapReduce with distributed grep (~*30 seconds*)
  - Prime example for RDBMS offloading

Use cases :: Accounting

- Break down usage of ATLAS data contents
  - Historical free-form meta data queries

```
{site, nbfiles, bytes} := {project=data10*, datatype=ESD, location=CERN*}
```

  - Non-relational periodic summaries
  - A full accounting run takes about 8 minutes
    - Pig data pipeline creates MapReduce jobs
    - 7 GB of input data, 100 MB of output data



- *(come and see the poster)*

Operational experiences :: Software

- MongoDB
  - Easiest to install (*download tarball, unpack, run*)
  - One line of configuration to change to create the cluster
- Cassandra
  - Packages from ASF
  - Straightforward installation and configuration via Puppet/tarball
  - However, nodes need special hardware configuration (two disks for commitlog and data)
- Hadoop
  - Cloudera distribution
    - Tests and packages the Hadoop ecosystem
  - Straightforward installation via Puppet/YUM
  - But the configuration was ... not so obvious
    - Many parameters, extensive documentation, but bad default performance
      - Cluster IO throughput maxing at 30MB/sec, network not saturated
    - But guidelines on how to set parameters properly only exist for large installations
    - Tweaked a lot, but most of the time it got worse and never better
    - Left it defaults *(next slide please...)*

Operational experiences :: Software

- SLC5 ?
  - But the throughput problem didn't come from Hadoop
  - Instead the 8-year-old kernel of SLC5 was the problem
    - No *epoll* (non-blocking-IO) support
- SLC6 !
  - Migrated the whole cluster in-flight to SLC6
    - Original reason for migration was because of a SLC5 kernel bug that broke Puppet
  - Procedure
    - 1. Drain one node (not exactly mandatory)
    - 2. Wipe and reinstall node with SLC6 + puppet template
    - 3. There is no step three (automatic resychronisation of node into cluster)
    - 4. Goto 1
  - Just a few minutes downtime while Hadoop headnode was migrated
    - Could have possibly averted downtime by manually assigning another headnode
    - (Latest Hadoop release can do it automatically now with high-availability headnode)
  - Performance increase of IO remarkable
    - Random read/write performance per node improved by factor 4
    - Cluster IO throughput now maxing at 80MB/sec, network saturated
- Backups
  - Hourly encrypted backups of the HDFS image
  - Cluster state can be restored within 3 minutes (including downloading and unpacking the backup)

Operational experiences :: Hardware

- Disk failure is common and cannot be ignored
- Data centre annual disk replacement rate up to 13% (Google & CMU, 2011)

- Within one year we had
  - 5 disk failures
    - 20% failure rate!
    - Out of which 3 happened at the same time
  - 1 Mainboard failure
    - Together with the disk failure, but another node

- Worst case scenario experienced up to now
  - 4 nodes out of 12 dead within a few minutes
  - Hadoop
    - Reported erroneous nodes
    - Blacklisted them
    - And resynced the remaining ones
  - No manual intervention necessary
  - Nothing was lost

Conclusions

- Structured storage systems are too useful to be ignored
- Hadoop proved to be the correct choice and an excellent platform for our analytical workloads
  - Stable – reliable – fast – easy to work with
  - Survived disastrous hardware failures
- DDM use cases well covered
  - Storage facility (*log aggregation*, *traces*, *web sharing*)
  - Data processing (*trace mining*, *accounting*, *searching*)
- Miscellaneous
  - All three evaluated products provide full durability, and transactions were not missed
  - We see Hadoop complementary to RDBMS, not as a replacement

- Future work
  - WAN replication as Hadoop is location aware
  - Generic RDBMS-to-HBase synchronisation framework
  - Improved data mining framework for generic analytics

STRUCTURED STORAGE IN ATLAS DISTRIBUTED DATA MANAGEMENT

Mario Lassnig, Vincent Garonne, Angelos Molfetas,
Thomas Beermann, Gancho Dimitrov, Luca Canali, Donal Zang,
on behalf of the ATLAS Collaboration
mario.lassnig@cern.ch
ph-adp-ddm-lab@cern.ch

CHEP'12, New York City, NY