

# HADOOP: EXPERIENCES IN DDM (AND FUTURE POTENTIAL)

**Mario Lassnig**

on behalf of the DDM Team

[mario.lassnig@cern.ch](mailto:mario.lassnig@cern.ch)

[ph-adp-ddm-lab@cern.ch](mailto:ph-adp-ddm-lab@cern.ch)

# Overview

2

- Structured storage
  - Concepts
  - Technologies
  - Hadoop in detail
- ATLAS DDM use cases
  - Storage facility
  - Data-intensive analytics
- Operational experiences
  - Software
  - Hardware
- Future potential
- Conclusions

# Structured Storage :: Concepts

3

- 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  $\neq$  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

4

- 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

5

- 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

6

- 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
- One-button push update on all nodes

- For comparison, in industry:

- 100 nodes is a small cluster
- 1000 nodes a medium cluster
- 10000+ nodes a large cluster

---

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

---

# Structured Storage :: Technologies :: Data Models

7



```
{
  _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



```
'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



```
'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

8

## 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

## A P A C H E 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

9

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

# Structured Storage :: Technology Selection

10

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

# Structured Storage :: Technology Selection

11

	<b>MongoDB</b>	<b>Cassandra</b>	<b>Hadoop/HBase</b> <span style="background-color: #ADD8E6;">CDH4</span>
<b>Installation/ Configuration</b>	Download, unpack, run	Download, unpack, configure, run	Distribution, Complex config
<b>Buffered read 256</b>	250'000/sec	180'000/sec	180'000/sec
<b>Random read 256</b>	20'000/sec	20'000/sec	20'000/sec
<b>Relaxed write 256</b>	10'000/sec	19'000/sec	16'000/sec
<b>Durable Write 256</b>	2'500/sec	9'000/sec	8'000/sec
<b>Analytics</b>	Limited MapReduce	Hadoop MapReduce	MapReduce, Pig, Hive
<b>Durability support</b>	Full	Full	Full
<b>Native API</b>	Binary JSON	Java	Java
<b>Generic API</b>	None	Thrift	Thrift, REST

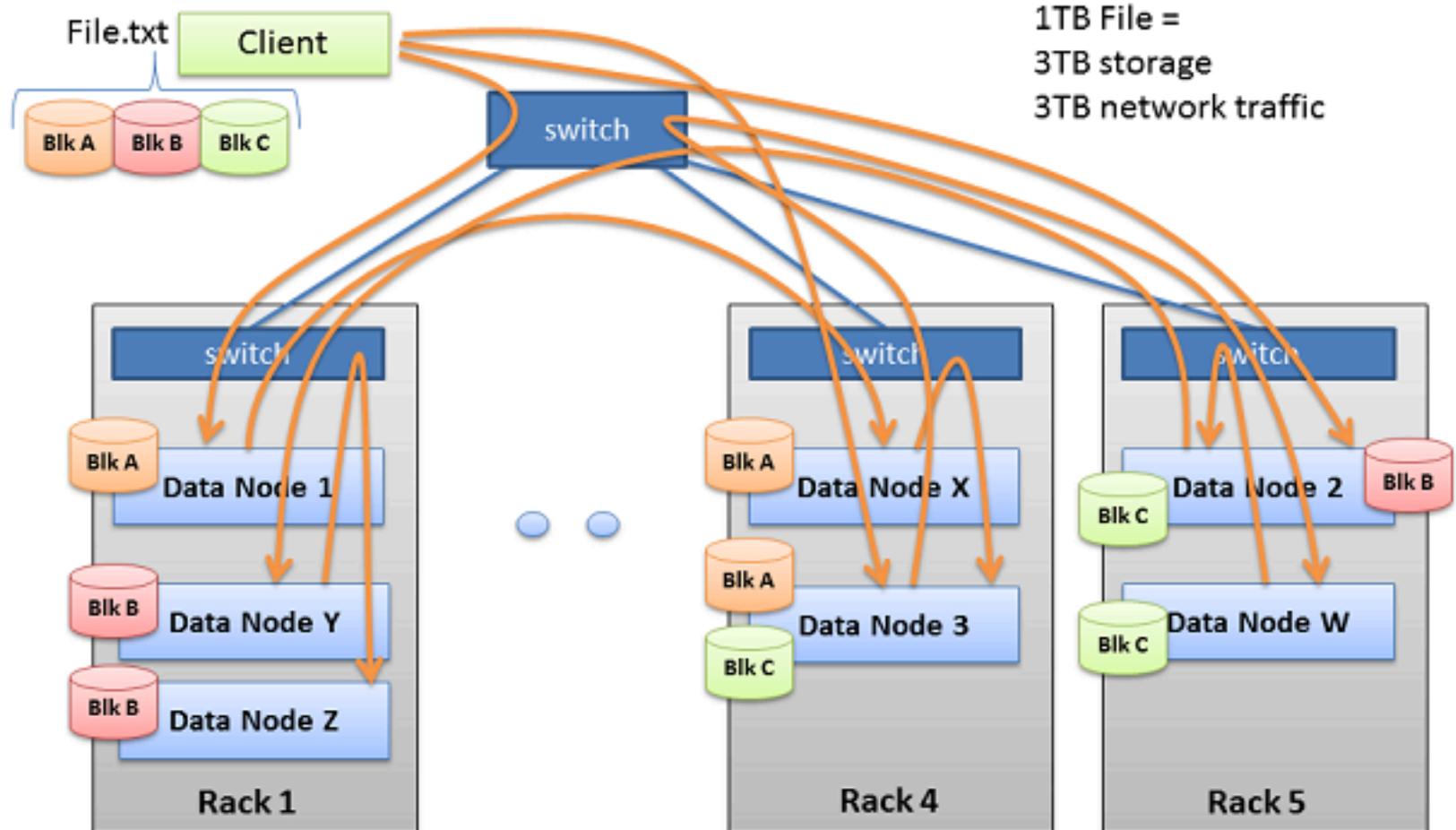
# Structured Storage :: Technologies ::

12

- 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

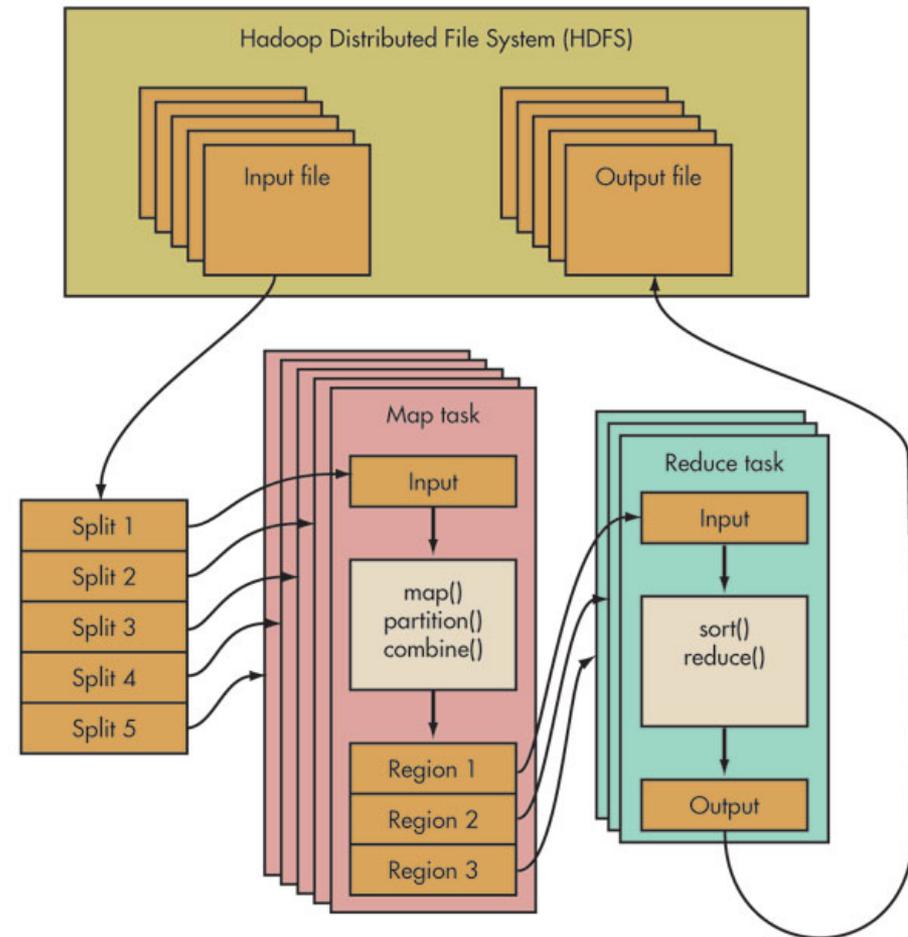
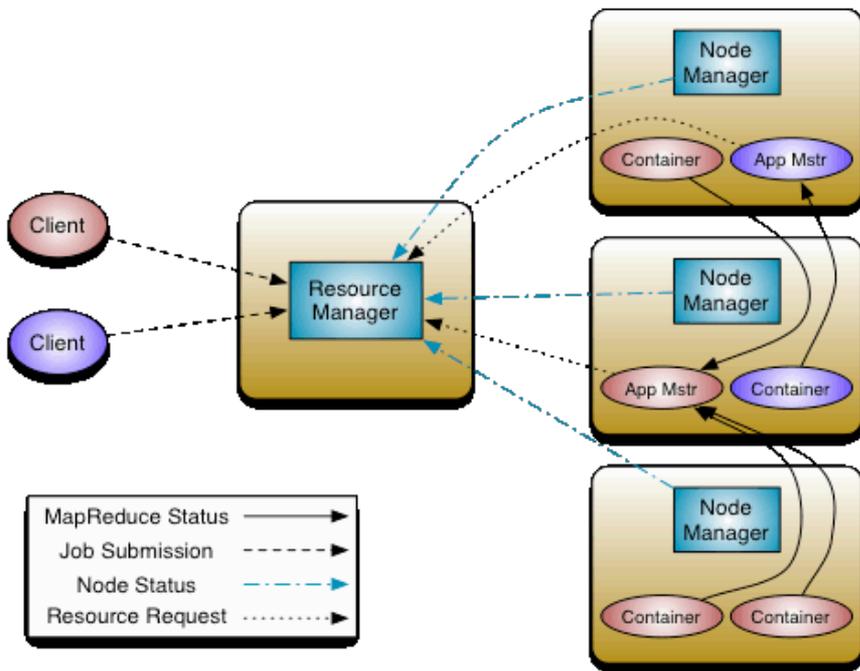
# Hadoop Architecture :: HDFS

13



# Hadoop Architecture :: MapReduce

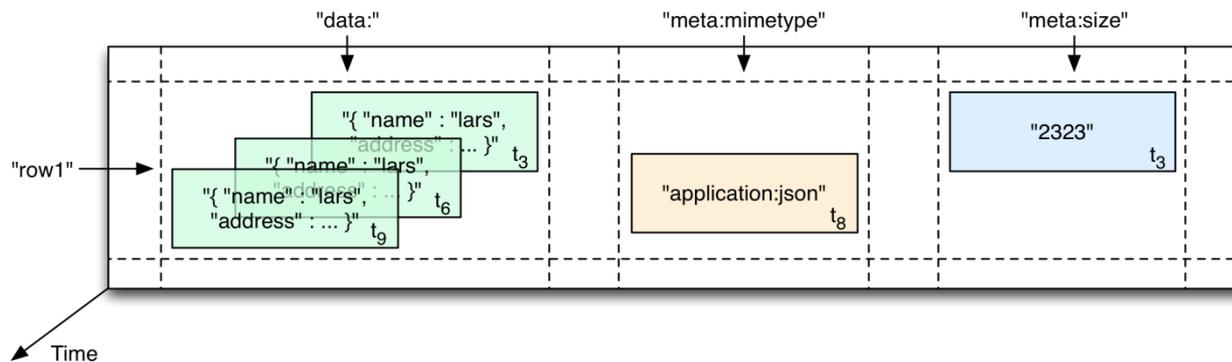
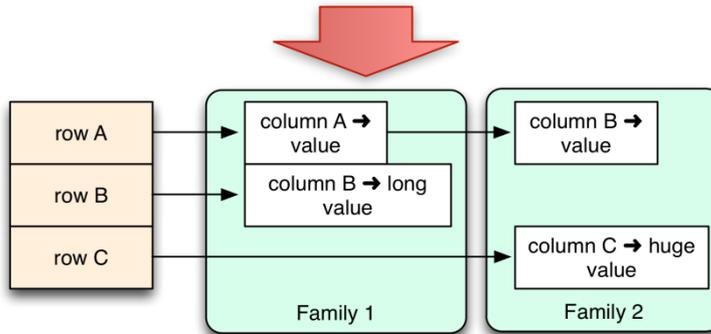
14



# Hadoop Architecture :: HBase

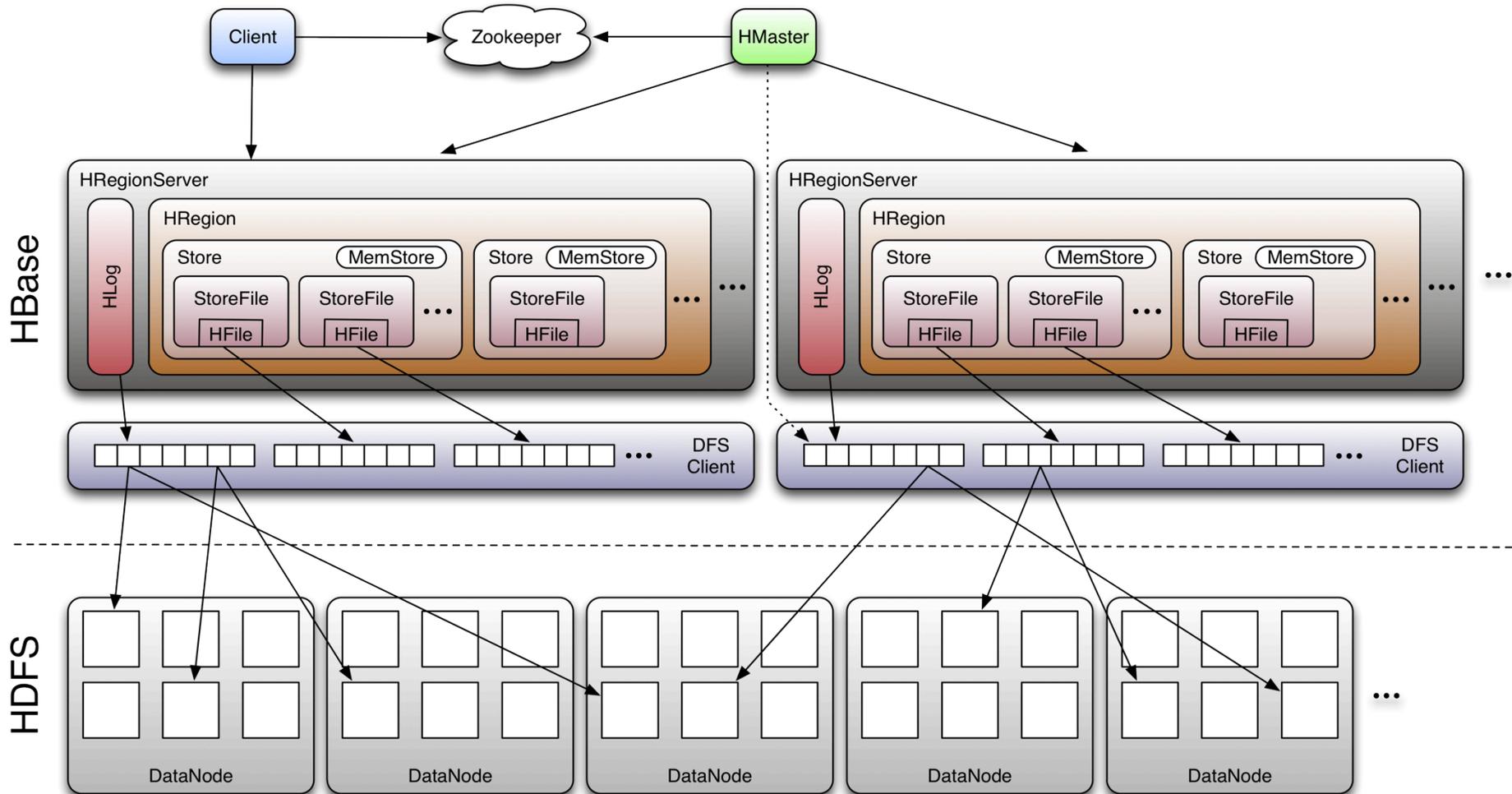
15

	column A (int)	column B (varchar)	column C (boolean)	column D (date)
row A				
row B				
row C			NULL?	
row D				



# Hadoop Architecture :: HBase

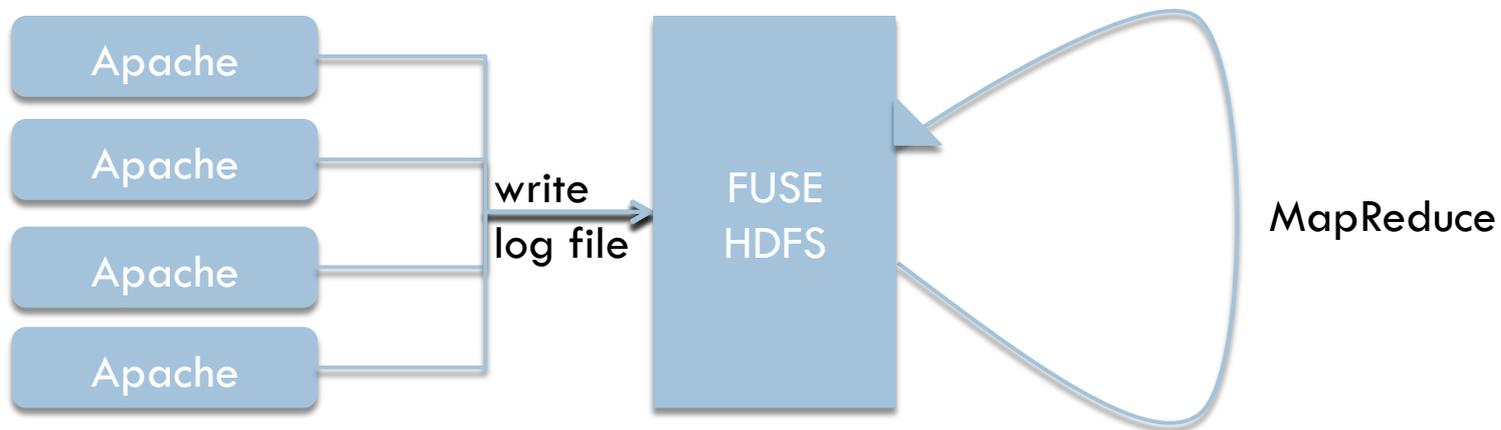
16



# Use cases :: Log file aggregation

17

- HDFS is mounted as a POSIX filesystem via FUSE
  - ▣ Daily copies of all the ATLAS DDM log files are aggregated in a single place
  - ▣ 9 months of logs accumulated, already using 3.5 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

18

- 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

19

- 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



ddo.000001.Atlas.Ideal.DBRelease.v07010104

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	<a href="#">ddo.000001.Atlas.Ideal.DBRelease.v07010104</a>	1 754 037 441	84	2009-07-02 02:18:11	.tar: ❌

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

# Use cases :: Wildcard search

20

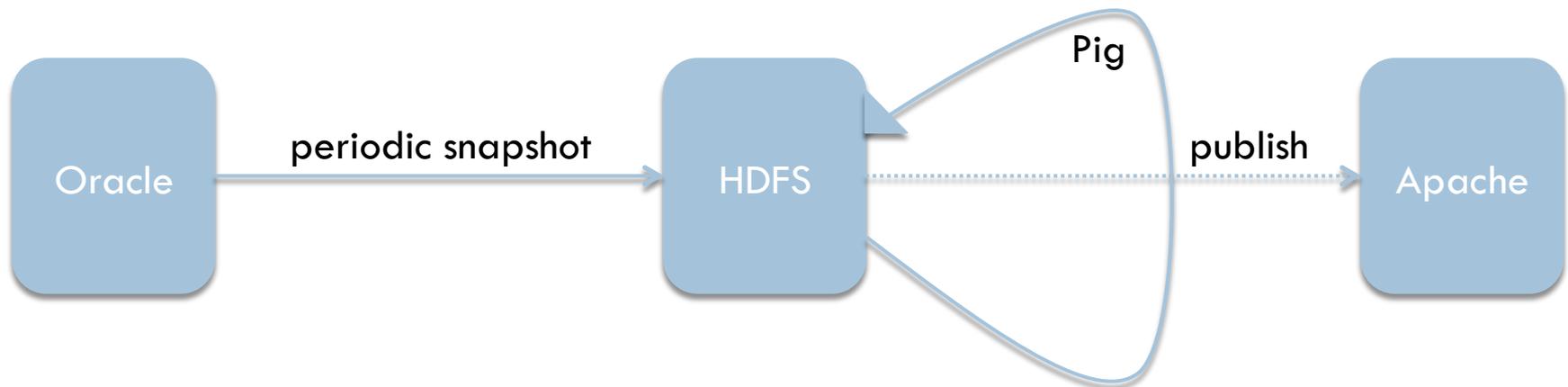
- List contents of ATLAS DDM based on a pattern
  - ▣ e.g., all data11 datasets (query: *data11\**)
  - ▣ RDBMS: *Index range scan (~2 seconds, in memory)*
- This becomes more expensive on sub-selections
  - ▣ e.g., all data11 datasets with a RAW datatype (query: *data11\*RAW\**)
  - ▣ RDBMS: *Index 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

21

- 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, 500 MB of output data



# Operational experiences :: Software

22

- 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

23

- 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 resynchronisation 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)
  - ▣ 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

24

- 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 (but one node is still causing trouble, RAID controller?, cluster not affected)

# Future potential

25

- Storage system
  - exposes data via HTTP(REST), POSIX(FUSE), Thrift, Java
- Distributed processing (HDFS, YARN)
  - If your problem needs to scan through lots of data
  - or: If your problem is embarrassingly parallel (= individual tasks do not share state)
- Free-form storage (HBase)
  - Have tens of billions of rows and need instant lookup?
    - Yes, distributed in-memory tables
  - No automatic secondary indexes though!
- Expressive querying (Hive, Pig)
  - Hive for SQL-aficionados (What do I want?)
  - Pig for data pipeline people (How do I want?)
- Application tie-ins
  - Automated log-file aggregation and monitoring (Flume)
  - RDBMS snapshotting (Sqoop)
  - Data Mining (Mahout)
  - Everything else (stdin/stdout)
- Open framework
  - Exploit, Understand, Contribute

# Future potential :: Operations

26

- Currently, I'm the sole maintainer of our cluster
  - ▣ Patches: ~1 hour (every 8-10 weeks)
    - Verify & Install, Puppetize, Deploy
  - ▣ Release upgrade: ~1 day (every year)
  - ▣ Other than that, it's zero maintenance
- But recently, many people asked to use our cluster!
- Via the Database TEG
  - ▣ Strong recommendation for a hosted Hadoop service
  - ▣ Not aware of any follow-up yet
- Future potential?
  - ▣ Maybe within IT Agile Infrastructure programme, or an IT section (DSS, DB, ?)
  - ▣ A single person can administrate multiple clusters
  - ▣ Some technical training needed (but not much)
  - ▣ Configuration/Deployment, etc... already in place and can be adapted

# Future potential :: Training

27

- Organizing official Hadoop courses at CERN
  - ▣ Administrator (Deploy, Maintain, Monitor, Backup, Replicate)
  - ▣ Developer (HDFS, YARN, HBase, Hive, Pig)
  - ▣ Hands-On!
- Together with HR – Technical Training
- From external certified instructors (Cloudera)
- Most likely in September
  - ▣ To be decided before mid-July
- This will not be a “basic” course (do your homework!)
  - Perhaps possibility to earn Hadoop Certified Administrator/Developer after
- Contact me directly if you are interested before mid-July
  - To keep you updated
  - If you want anything particular to be explained during the courses

# Conclusions

28

- Structured storage systems are too useful to be ignored
- Hadoop proved to be the correct choice and an excellent platform for our analysis 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*)
  - ▣ We see Hadoop complementary to RDBMS, not as a replacement
- Future potential
  - ▣ Storage system, Parallel access to distributed data
  - ▣ Well-maintained, lots of industrial and financial backing
- Future work within DDM using Hadoop
  - ▣ WAN replication (off-data-centre backup), as Hadoop is location aware
  - ▣ Generic RDBMS-to-HBase synchronisation framework
  - ▣ Improved data mining framework for generic analytics
  - ▣ Integrate as Rucio backend

# HADOOP: EXPERIENCES IN DDM (AND FUTURE POTENTIAL)

**Mario Lassnig**

on behalf of the DDM Team

[mario.lassnig@cern.ch](mailto:mario.lassnig@cern.ch)

[ph-adp-ddm-lab@cern.ch](mailto:ph-adp-ddm-lab@cern.ch)