Processing of the WLCG monitoring data using NoSQL

J. Andreeva, A. Beche, S. Belov, I. Dzhunov, I. Kadochnikov,
E. Karavakis, P. Saiz, J. Schovancova, D. Tuckett
CERN IT/SDC/MI

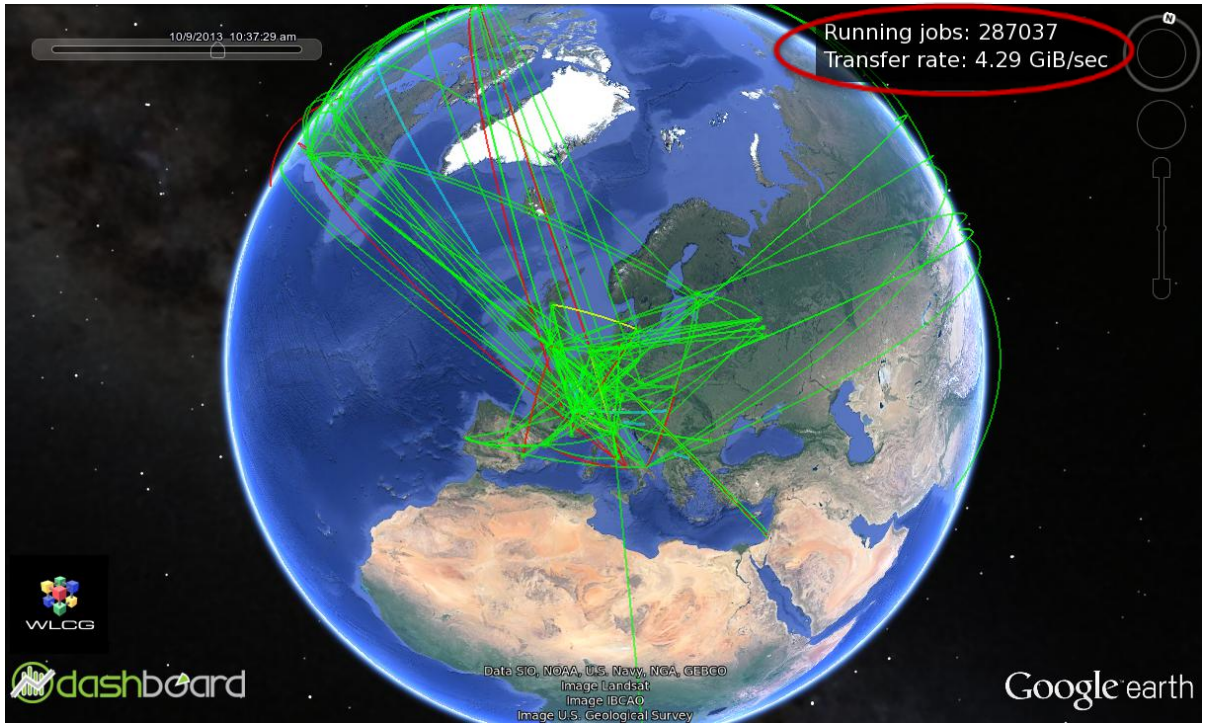
CHEP 2013 - 17/10/2013

Outline

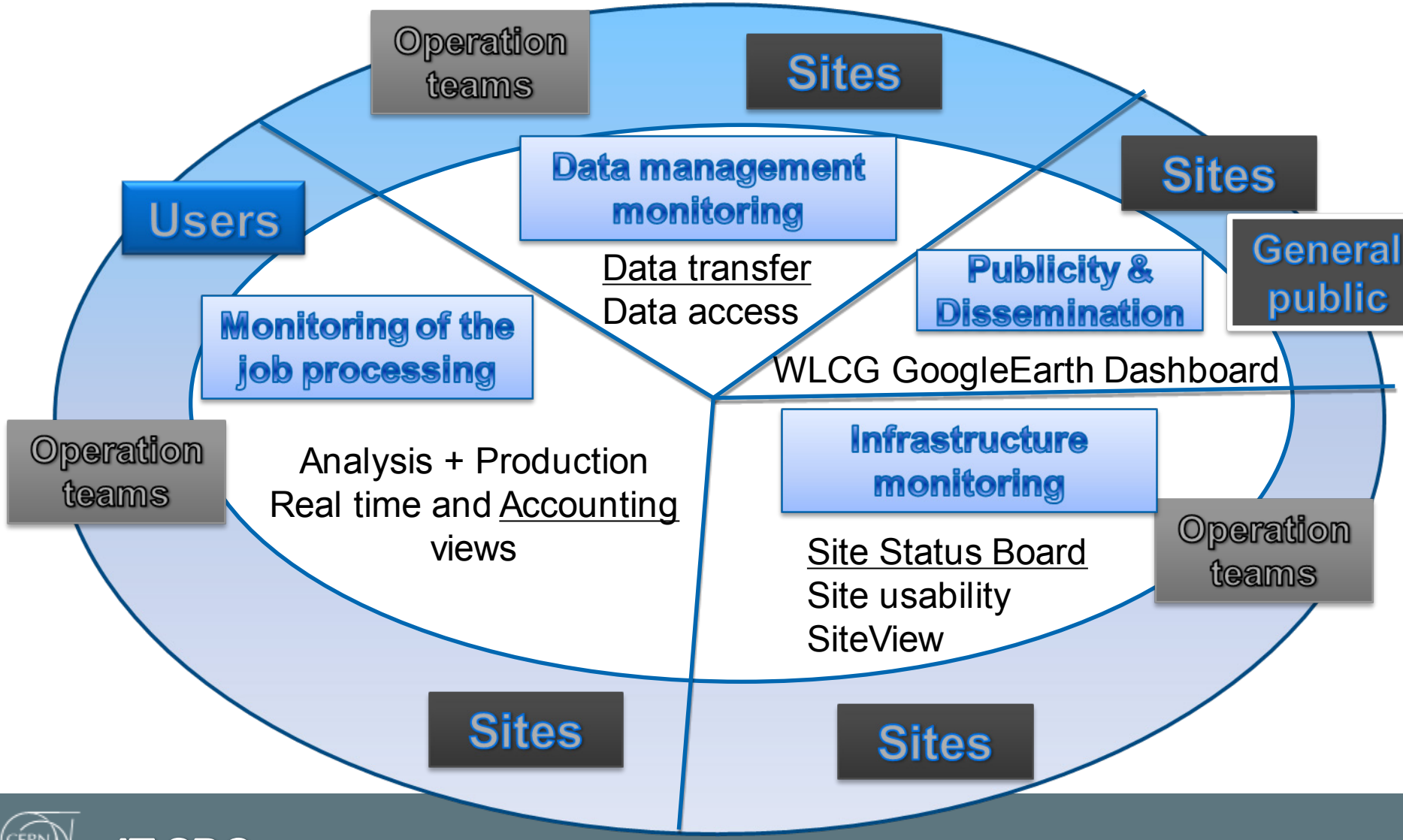
- Monitoring the WLCG
  - Experiment Dashboard
- Challenges
- Evaluation of NoSQL solutions in two use-cases
  - Apps that require grouping by multiple fields
    - Job Accounting
    - WLCG Transfers
  - Apps that group by single field
    - Site Status Board
- Future work
- Conclusion

Monitoring the WLCG

- More than 150 computing centres in nearly 40 countries
- *Reliable monitoring is complicated!*



Experiment Dashboard solutions



Experiment Dashboard solutions

- Python framework for developing Grid Monitoring apps
- Provides common solutions across multiple VOs and middleware
- Heavily used within LHC experiments
  - More than 2.5K unique visitors per month

Challenges

- Amount of data is growing!
  - We need to scale horizontally
- Heterogeneity of data/schema
- Oracle currently used. Whether existing open source solutions can provide better performance and how difficult would it be to migrate?

Evaluation of alt. solutions

- Web UIs are decoupled from data storage technology
- In line with the strategy of the IT department
- Many different technologies to consider as an alternative depending on the schema/use-case:
  - Open source RDBMS
    - MySQL, PostgreSQL, etc ...
  - NoSQL solutions
    - Hadoop / HBase, Elasticsearch, etc ...

- Not a technology benchmark
  - We are comparing *our* Oracle cluster with different storage solutions for *our* use-cases

Cluster specifications

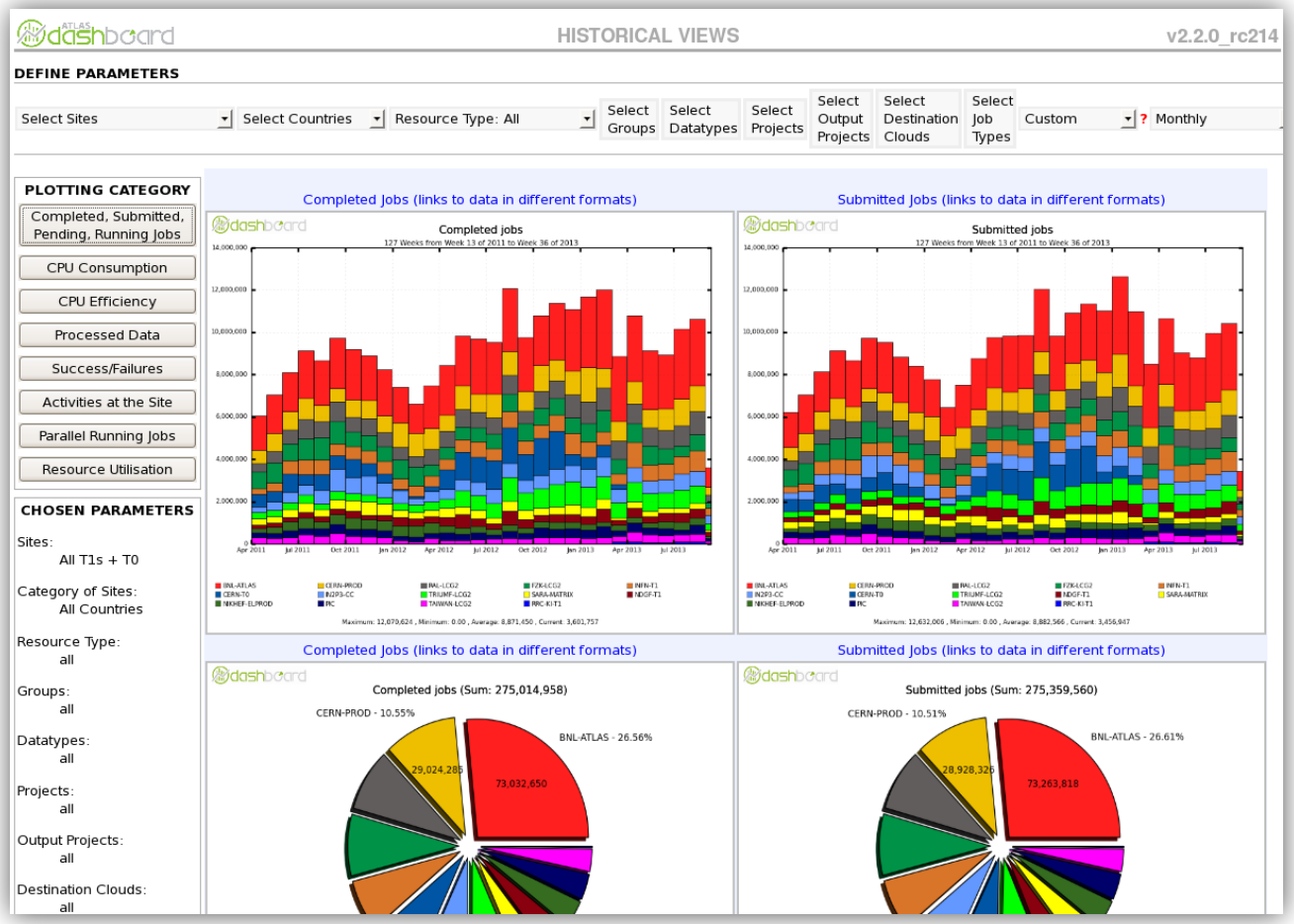
Oracle 11g RAC (Shared)		**5 Physical machines** CPU : 4 cores (8Threads) 2.5GHz RAM: 48GB
Elasticsearch cluster		**6 Virtual machines** CPU : 4 cores 2.3GHz RAM: 8GB
Hadoop cluster		**8 Virtual machines** CPU : 4 x 4 + 4 x 8 cores (2.2GHz) RAM: 4 x 8GB + 4 x 16GB

*Oracle had many users when we ran the test – HBase and Elasticsearch had few users

*Didn't use the 'parallel' execution hint in Oracle

Test Case #1: Job Accounting

- Time series data
- Filtering and grouping by multiple fields



Job Accounting

- Imported 8 million rows (stats from 2010) ~ 2.4 GBs
- HBase key in the form of:
  Date_Site_Activity_InputDataType_Group_Project_DestinationCloud_HighLevelActivity_ResourcesReporting_OutputProject
- *Time series data into HBase are problematic*
  - *they result in monotonically increasing row-keys preventing full leverage of parallelism*
- We *always* query on the time range and data need to be accessed in an ordered way
- One column family, 52 columns

Performance Benchmarking

- Didn't use native Java since our framework is written in Python
- Used HappyBase, a high-level Python HBase specific lib
- Used THRIFT interface instead of REST
  - REST is slower than THRIFT and you cannot use custom filters
  - THRIFT is still slower than a native Java client performing large scans

HBase cluster performance tuning

- Very slow scanning results with the default HBase config parameters (see backup slides)
- Performed various optimisations:
  - *hbase.regionserver.handler.count* to 100 instead of 10
  - *hbase.client.scanner.caching* to 1000 instead of 1
  - *hbase.hregion.memstore.flush.size* to 256 MB instead of 128 MB
  - *hbase.hregion.max.filesize* to 256 MB instead of 1 GB
  - *hfile.block.cache.size* to 0.30% instead of 0.25%
  - *hbase.master.handler.count* to 100 instead of 25
  - *hbase.regionserver.checksum.verify* to true

Job Accounting: Oracle VS HBase

Scan type		Oracle 1st hit (grouping)		Oracle 1st hit (no grouping)		HBase (no grouping)	
Period	Filter	Time in secs	Avg. rows	Time in secs	Avg. rows	Time in secs	Avg. rows
1 day	0	0.031	116	0.61	10K	2.13	10K
1 week	0	0.2	807	4.54	70K	13.49	70K
1 month	0	0.956	3.6K	59.03	337K	88.26	337K
1 day	1	0.013	13	0.019	144	0.206	144
1 week	1	0.018	98	0.074	1K	0.977	1K
1 month	1	0.101	431	0.473	5.4K	2.25	5.4K
1 day	2	0.010	5	0.010	28	0.20	28
1 week	2	0.013	28	0.021	178	0.681	178
1 month	2	0.055	123	0.122	925	1.692	925

Job Accounting in Elasticsearch

- Considered alternatives: Elasticsearch was suggested by CERN AI Monitoring team
- *"flexible and powerful open source, distributed real-time search and analytics engine for the cloud"* (http://www.elasticsearch.org/)
- Features: real time data, real time analytics, distributed, multi-tenancy, high availability, full text search, document oriented, conflict management, schema free, restful api, per-operation persistence, apache 2 open source license, build on top of apache lucene
- Imported same amount of data as in HBase

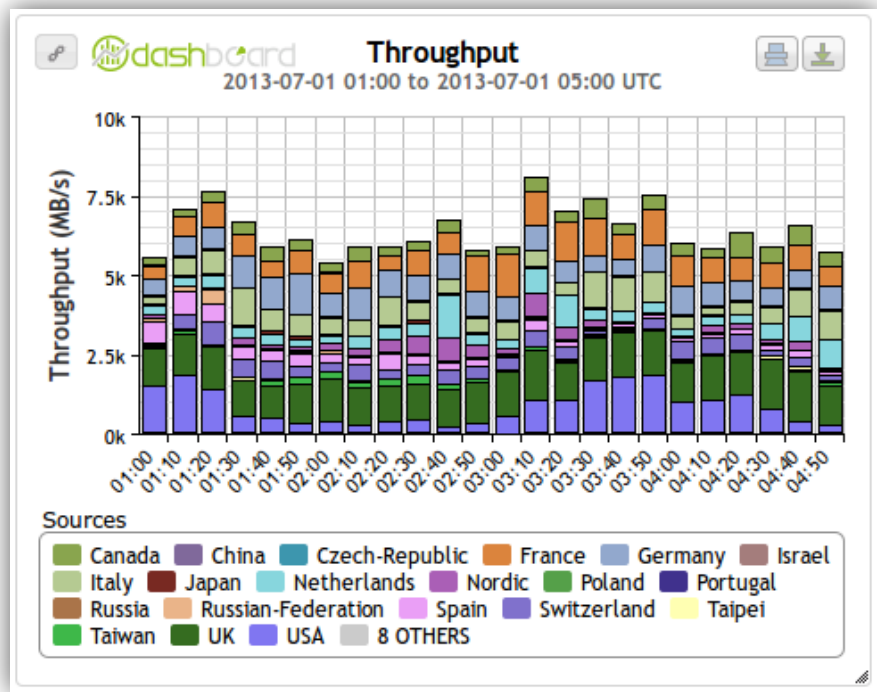
Job Accounting: Oracle VS Elasticsearch

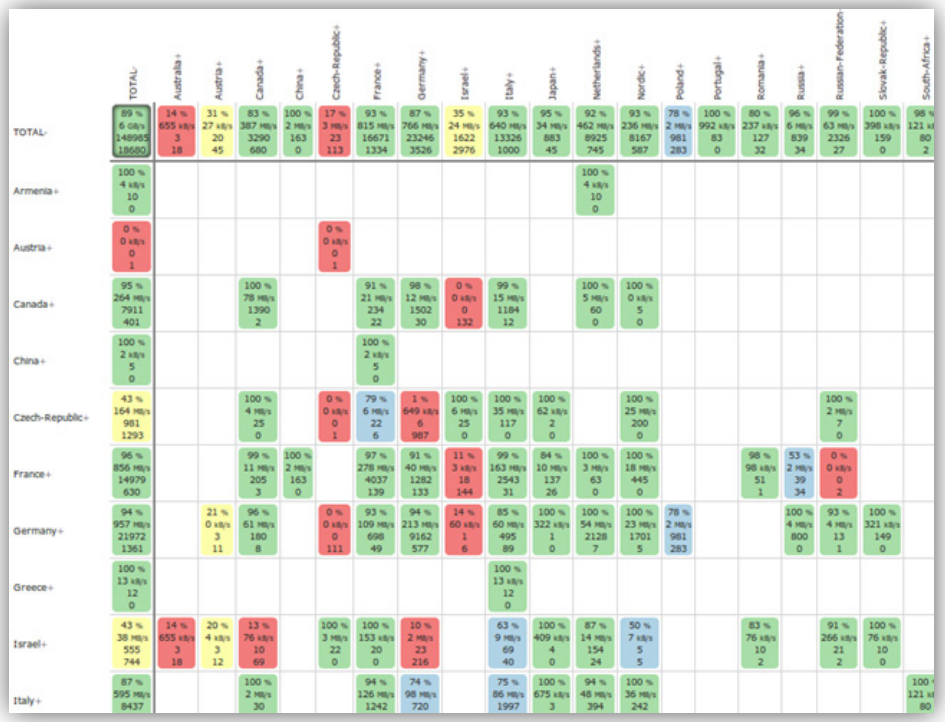
Scan type		Avg. rows	Oracle 1st hit in secs	Elasticsearch in secs
Period	Filter			
1 day	0	116	0.031	0.017
1 week	0	807	0.2	0.118
1 month	0	3.6K	0.956	0.138
2 months	0	7K	2.27	0.160
1 day	1	13	0.013	0.016
1 week	1	98	0.018	0.021
1 month	1	431	0.101	0.056
2 months	1	864	0.16	0.062
1 day	2	5	0.010	0.003
1 week	2	28	0.013	0.004
1 month	2	123	0.055	0.031
2 months	2	259	0.101	0.097

Test Case #2: WLCG Transfers

Plot statistics

- Time series data
- Filtering and grouping by multiple fields





Matrix statistics

- Filtering and grouping by multiple fields

WLCG Transfers

- Considered benchmarking performance on HBase but..

# records	Native JAVA Client	THRIFT Client
68970	0.629 secs	11.04 secs

  - Running on the Hadoop cluster
- Decided to evaluate Elasticsearch
- Imported 1 month (July 2013) of statistics in 10 minute bins from WLCG Transfers Dashboard – 12.8 million rows - 2.9 GB

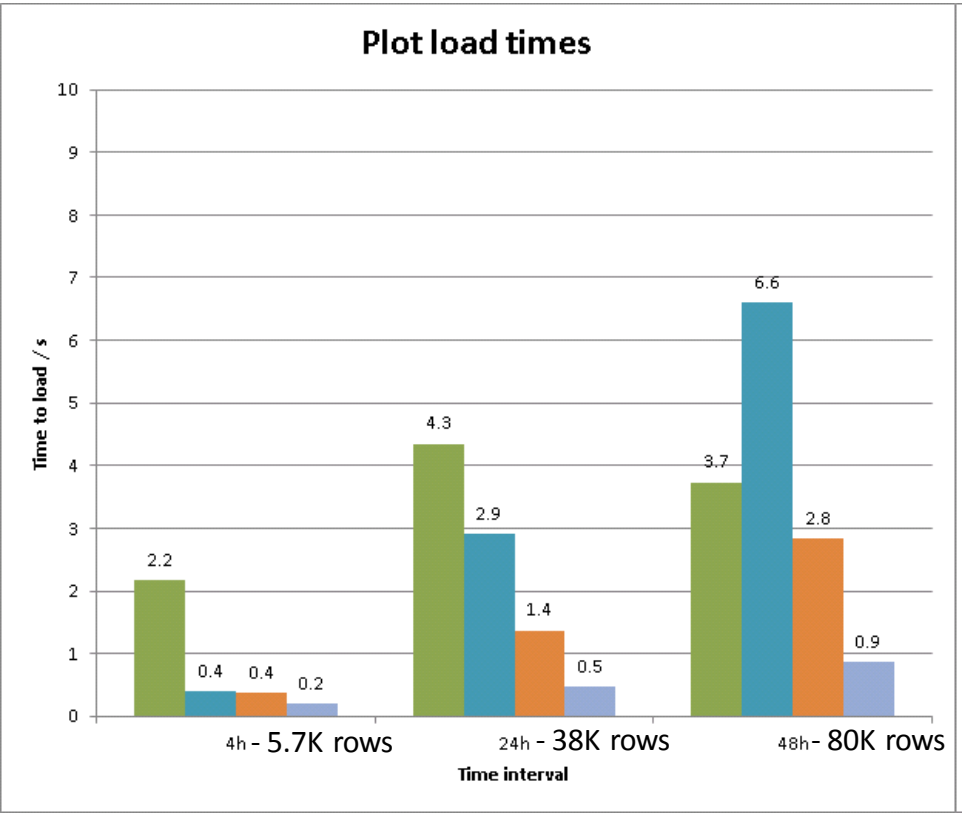
Grouping : Elasticsearch 0.90.3 Limitations

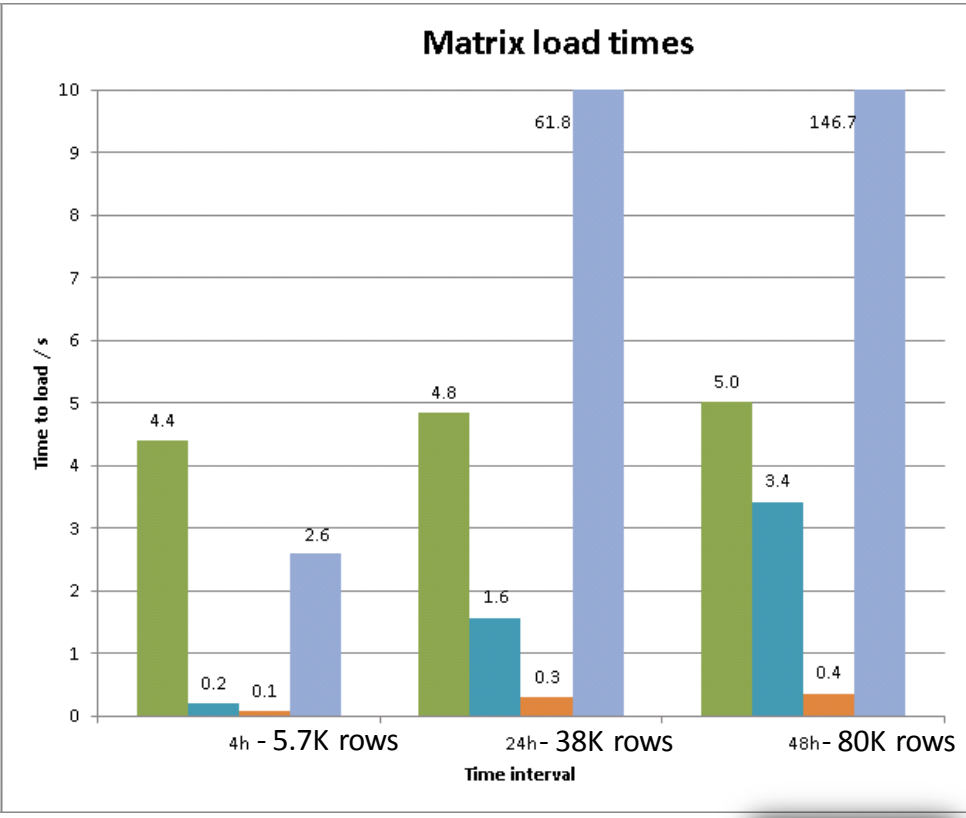
- Currently, grouping by multiple fields for statistical aggregations is not supported
  - *Investigated many workarounds!*
- The future release 1.0 will support grouping by multiple fields

Grouping : Oracle & Elasticsearch Methods

- OG: Oracle Grouping
  - Query using “group by” for user selected grouping fields
- ENG: Elasticsearch No Grouping
  - Query for all data
  - Grouping in the web action
- EIG: Elasticsearch Index Grouping
  - Add single field in index with all possible grouping fields concatenated
- EQG: Elasticsearch Query Grouping
  - Query to list n distinct combinations of selected grouping fields
  - Query n times filtering by distinct combinations

Data Out



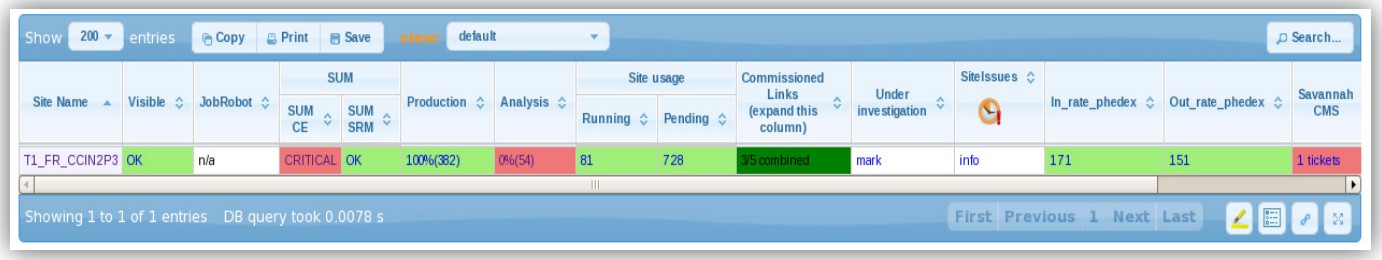


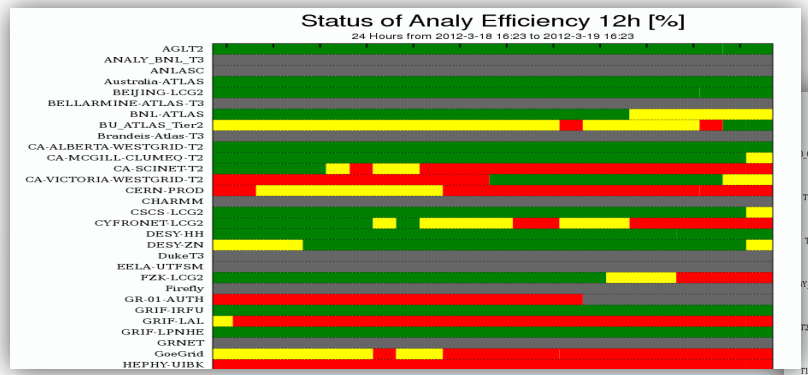
- ENG is much faster than Oracle for small row counts but won't scale
- EIG is faster than Oracle in all cases but inflexible
- EQG is much faster for few distinct grouping values but won't scale

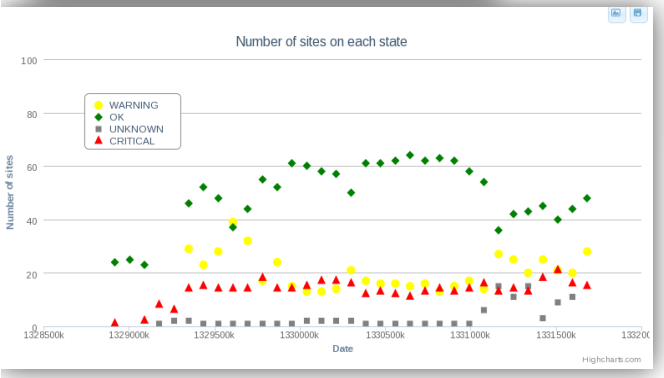
Test Case #3: Site Status Board

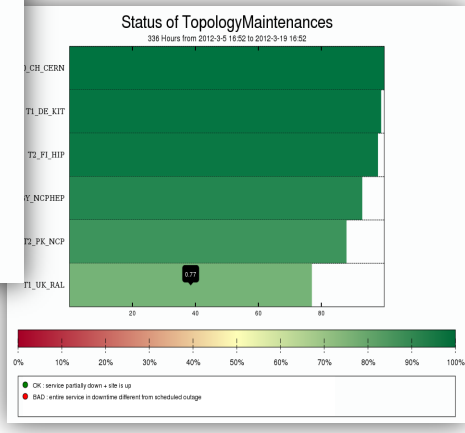
Current status

- Filtering by multiple fields









Historical data

- Filtering by multiple fields
- Grouping by single field

Site Status Board

- Imported a metric with 3 years data - 4M rows

Scan type	Avg. rows	Oracle 1st hit	Elasticsearch
1 day all sites	3K	5.6 secs	0.2 secs
1 week all sites	29K	7.76 secs	0.8 secs
1 month all sites	130K	29 secs	4 secs
3 months all sites	400K	53 secs	16 secs
1 month multiple sites	22K	3.3 secs	0.6 secs

Future work

- HBase
  - Use Coprocessors to aggregate data
  - Use Jython instead of HappyBase
- Elasticsearch
  - Evaluate version 1.0 when available, which will support grouping by multiple fields for statistical aggregations
  - Evaluate on shared physical cluster

Conclusion

- There is no single solution for every use-case!
- HBase
  - Current evaluation showed poor performance with sorted time series data
  - Further investigation planned
- Elasticsearch
  - Faster than Oracle 1st hit
  - Straightforward for use-cases requiring at most a single field grouping
  - Diverse workarounds required for multi-field grouping
- Early results are quite positive! For some WLCG monitoring applications, appropriate solutions were already identified – for others more investigation is required

Backup Slide #1
Job Accounting: Oracle VS HBase without any HBase optimisations

Imported 2.7 million records in HBase ~ 800 MB

Scan type		Oracle 1st hit (grouping)		Oracle 1st hit (no grouping)		HBase (no grouping)	
Period	Filter	Time in secs	Avg. rows	Time in secs	Avg. rows	Time in secs	Avg. rows
1 day	0	0.031	116	0.61	10K	18.93	10K
1 week	0	0.2	807	4.54	70K	150.87	70K
1 month	0	0.956	3.6K	59.03	337K	949.92	337K
1 day	1	0.013	13	0.019	144	0.877	144
1 week	1	0.018	98	0.074	1K	3.62	1K
1 month	1	0.101	431	0.473	5.4K	18.30	5.4K
1 day	2	0.010	5	0.010	28	0.267	28
1 week	2	0.013	28	0.021	178	1.65	178
1 month	2	0.055	123	0.122	925	6.43	925

Backup Slide #2
Job Accounting: Oracle VS HBase without any HBase optimisations

- HBase scales by having regions across many servers
  - default size of a region is 1GB
- Our data was only concentrated on just 3 (replication factor) out of the 8 nodes - nearly the entire cluster was idle!
- *Scans in HBase execute over a single region in a serial manner!*