

Monitoring WLCG with the *lambda* architecture

A new processing framework based on Hadoop (and friends)

Luca Magnoni
CERN IT

	TRANSFER			STAGING			DELETION			�			CENTRAL			DEF			IT			DE			CERN+			CA+			TOTAL		
CA+	97 % 757 MB/s	100 % 27 MB/s	99 % 2 GB/s	100 % 166 MB/s	96 % 84 kB/s	98 % 188 MB/s	100 % 7 MB/s	100 % 79 MB/s	96 % 7 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s			
CERN+	98 % 787 MB/s	85 % 177 MB/s	95 % 63 MB/s	97 % 252 kB/s	97 % 704 MB/s	100 % 15 MB/s	100 % 334 kB/s	100 % 9 MB/s	99 % 63 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s			
DE+	99 % 2 GB/s	97 % 103 MB/s	100 % 847 MB/s	92 % 75 MB/s	100 % 2 MB/s	100 % 673 MB/s	97 % 39 MB/s	99 % 63 MB/s	99 % 96 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s			
ES+	99 % 252 MB/s	100 % 0 kB/s	100 % 161 MB/s	90 % 2 MB/s	100 % 78 kB/s	100 % 172 MB/s	99 % 31 MB/s	100 % 13 MB/s	99 % 13 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s			
FR+	100 % 997 MB/s	100 % 845 kB/s	97 % 922 MB/s	99 % 59 MB/s	92 % 11 MB/s	100 % 138 MB/s	95 % 70 MB/s	87 % 4 MB/s	99 % 99 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s			
IT+	97 % 348 MB/s	100 % 37 MB/s	100 % 2 GB/s	98 % 4 MB/s	100 % 344 kB/s	100 % 204 kB/s	100 % 100 MB/s	100 % 2 MB/s	99 % 99 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s	98 % 740 MB/s	99 % 100 MB/s	99 % 100 MB/s			
	99 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %	99 %		



Outline

- Experiment Dashboard as use case
- WLCG Monitoring: challenges
- A new monitoring architecture
- Hadoop at work:
 - Data representation: make your choices
 - Performance results
- Real-Time layer: options and tools



Experiment Dashboard (ED)



- A common monitoring framework developed at CERN
- Multiple Targets:
 - Data activities
 - Job Processing
 - Services and Sites Availability
- Different Perspectives:
 - Users, Experts, Sites

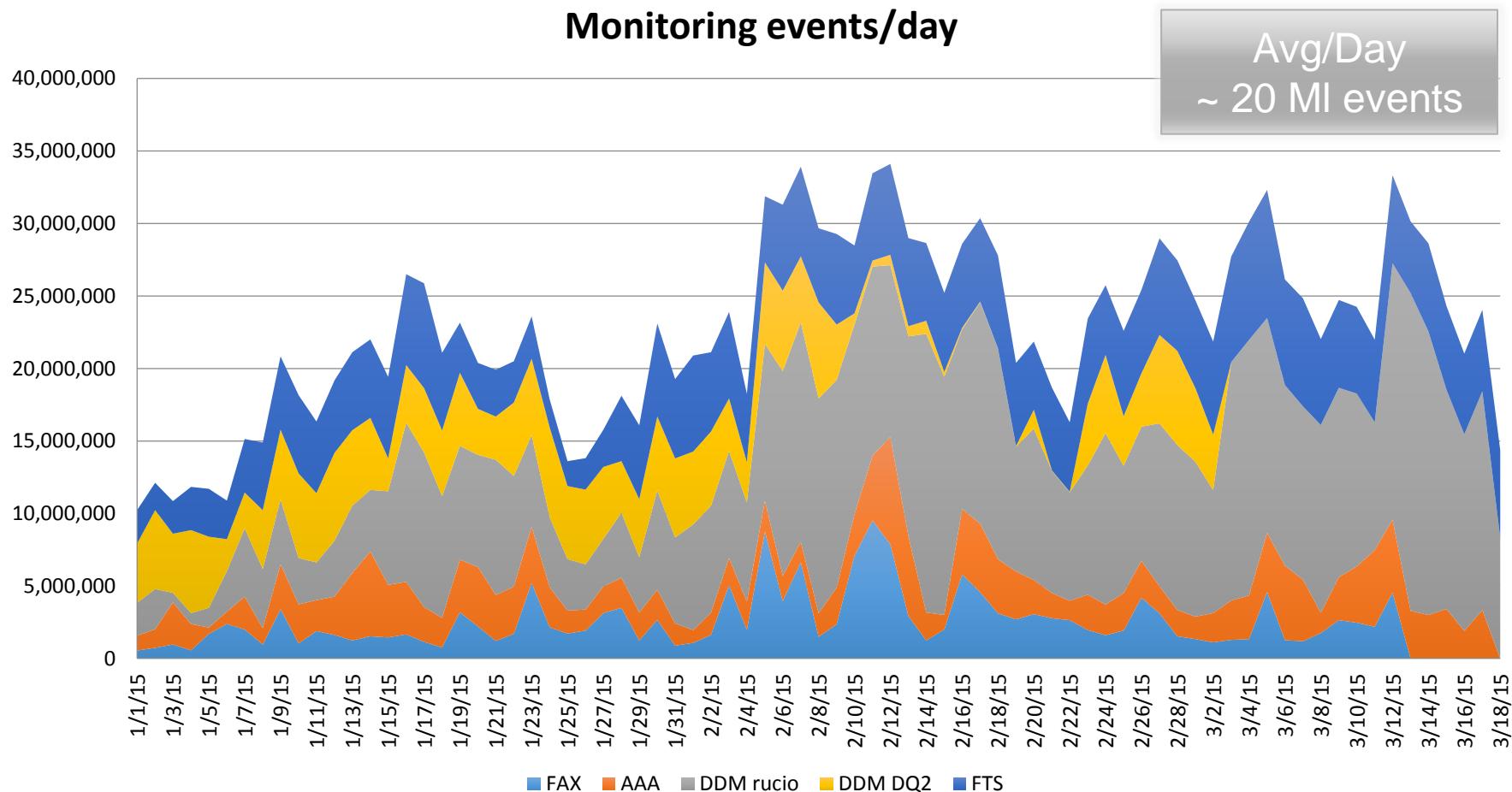
What does it do? *Analytics.*

The screenshot shows the ATLAS DDM Dashboard interface. On the left, a sidebar displays a JSON configuration for a transfer operation, with a large blue arrow pointing from the text towards the dashboard. The main area contains four plots: 'Transfer Efficiency' (a scatter plot showing efficiency by source and destination), 'Transfer Volume' (a stacked bar chart of volume over time), 'Transfer Successes' (a stacked bar chart of successful file transfers), and 'Transfer Failures' (a stacked bar chart of failed file transfers). Each plot includes a legend for sources: CA, CERN, DE, ES, FR, IT, ND, NL, RU, TW, UK, and US.

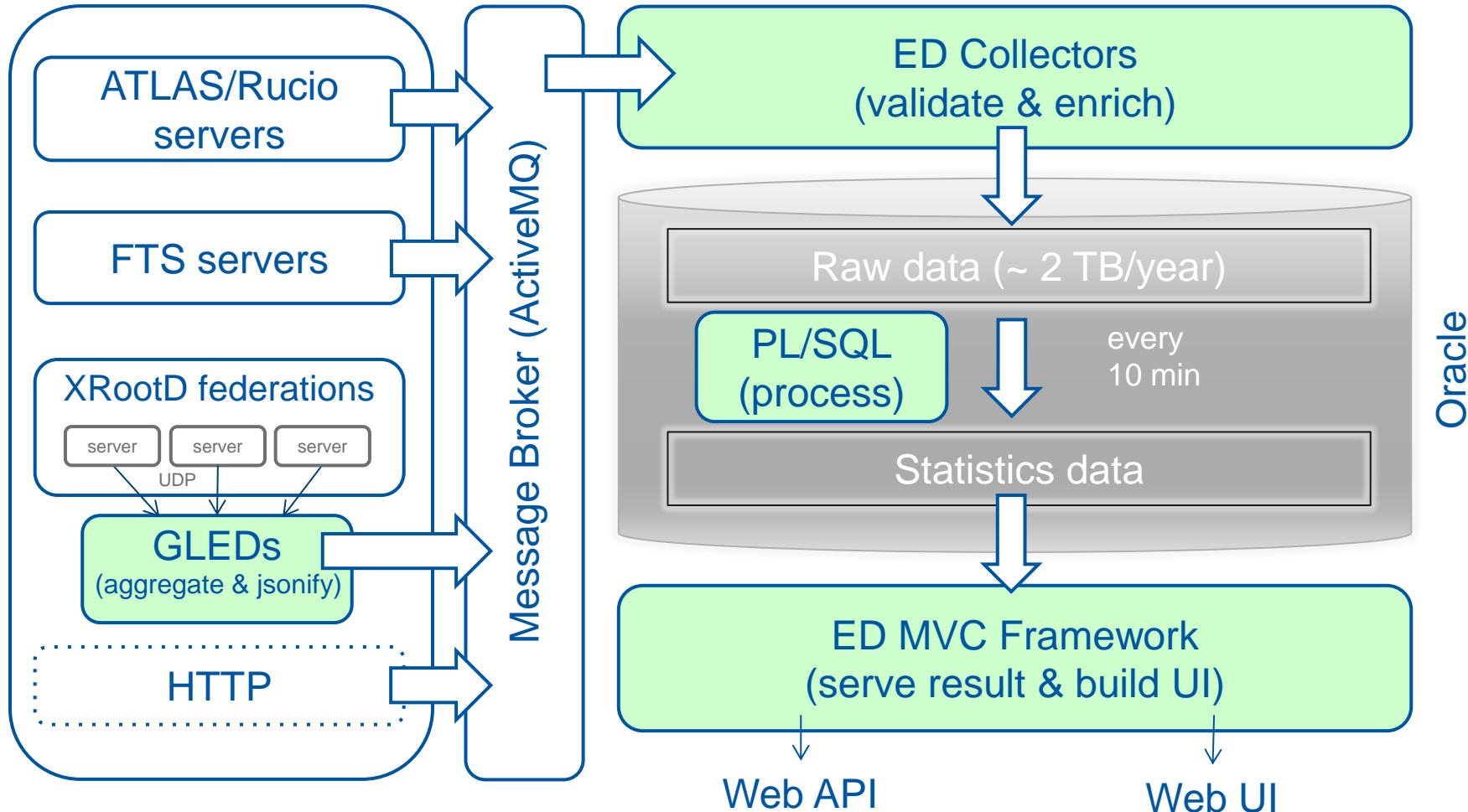
```
{"\"unique_id\": \"30", "\"file_lfn\": \"/store/RAW/castor_tsg_40b", "\"file_size\": \"40", "\"read_bytes\": \"0", "\"read_average\": \"", "\"read_single_operation\": \"", "\"read_single_average\": \"", "\"read_vector_byt", "\"read", "\"read", "\"read_vector_coun", "\"read_vector_count", "\"write_operations", "\"write_average\": \"", "\"read_bytes_at_clo", "\"user_dn\": \"/DC=ca", "Simpson\", \"us", "\"client_domain\": ", "\"user_protocol\": ", "\"app_info\": \"39_1", "https://glidein.cern.ch/39/1318:19342:crab:QCD80:March18_0\",", "\"server_domain\": \"brunel.ac.uk\", \"server_host\": \"dc-grid-pool-a4-03\",", "\"server_site\": \"T2_JP_T2_Kyoto\""}  
{"\"unique_id\": \"30", "\"file_lfn\": \"/store/RAW/castor_tsg_40b", "\"file_size\": \"40", "\"read_bytes\": \"0", "\"read_average\": \"", "\"read_single_operation\": \"", "\"read_single_average\": \"", "\"read_vector_byt", "\"read", "\"read", "\"read_vector_coun", "\"read_vector_count", "\"write_operations", "\"write_average\": \"", "\"read_bytes_at_clo", "\"user_dn\": \"/DC=ca", "Simpson\", \"us", "\"client_domain\": ", "\"user_protocol\": ", "\"app_info\": \"39_1", "https://glidein.cern.ch/39/1318:19342:crab:QCD80:March18_0\",", "\"server_domain\": \"brunel.ac.uk\", \"server_host\": \"dc-grid-pool-a4-03\",", "\"server_site\": \"T2_JP_T2_Kyoto\""}
```

**Collects and processes monitoring data
to build custom visualization**

Monitoring WLCG Data Activities



Current ED Data Pipeline



Oracle

It works, but...

Operational cost

- Custom code/services
- Fragility
- Complexity
 - Too many transformation
 - Difficult to test/validate

XRootD federations

server server server

UDP

GLEDs

(aggregate & jsonify)

Message Broker (A)

Propagation latency

- Reports when file is closed
- Delays data forwarding

ED Collectors (validate & enrich)

Raw data (~ 2 TB/year)

PL/SQL (process)

every
10 min

Processing does not scale well with data volume

- Fluctuations:
 - From few seconds to minutes/each run
- Difficult to improve complexity
 - Spikes > 10 minutes (affects UI)
- Reprocessing is expensive (i.e. days/week)

Oracle

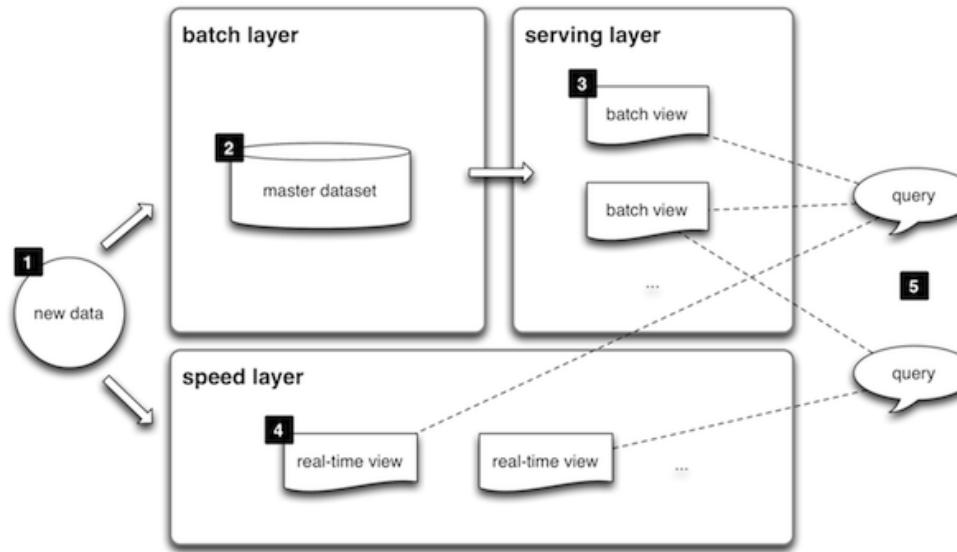
Architecture evolution

“80 percent of the development effort in a big data project goes into data integration and only 20 percent goes toward data analysis.” [Intel ETL White paper]

- Goal:
 - Scalable/Simplified/Mainstream
- Challenges:
 - Collect the raw data, process once and transform only when needed
 - More data (~ x10) to be archived and analyzed
 - Current PL/SQL jobs cannot make it
 - Hadoop/MapReduce seems appropriate, but it's a new processing paradigm and infrastructure

Lambda architecture

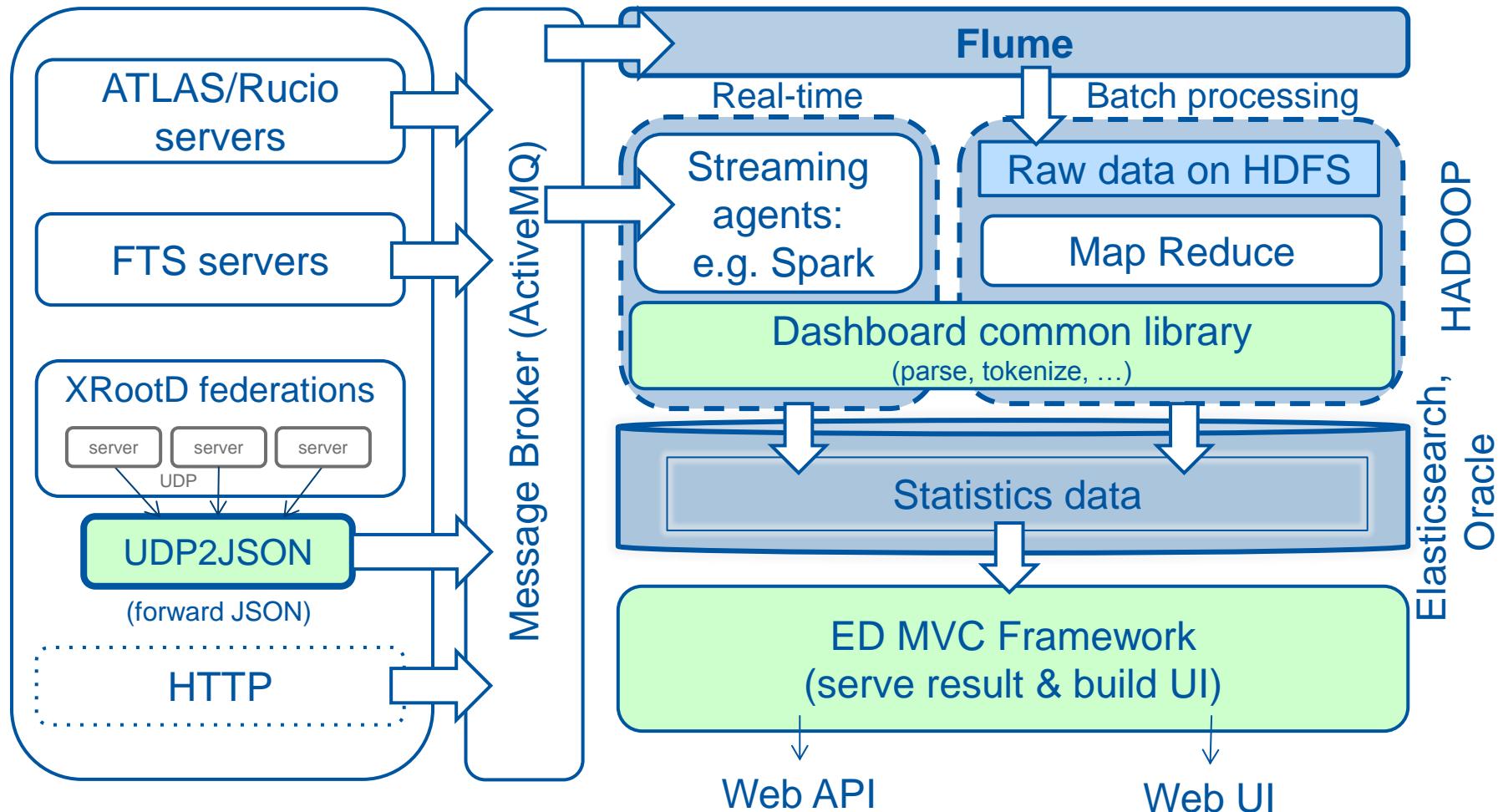
- Taking inspiration from the Lambda architecture
 - Implemented at Twitter
 - Intuitively, *one-technology-does-not-fit-all* idea
 - Batch for slow, reliable and stateless processing
 - Real-Time for fast, complex and incremental computation
 - Serve result from a dedicated serving layer



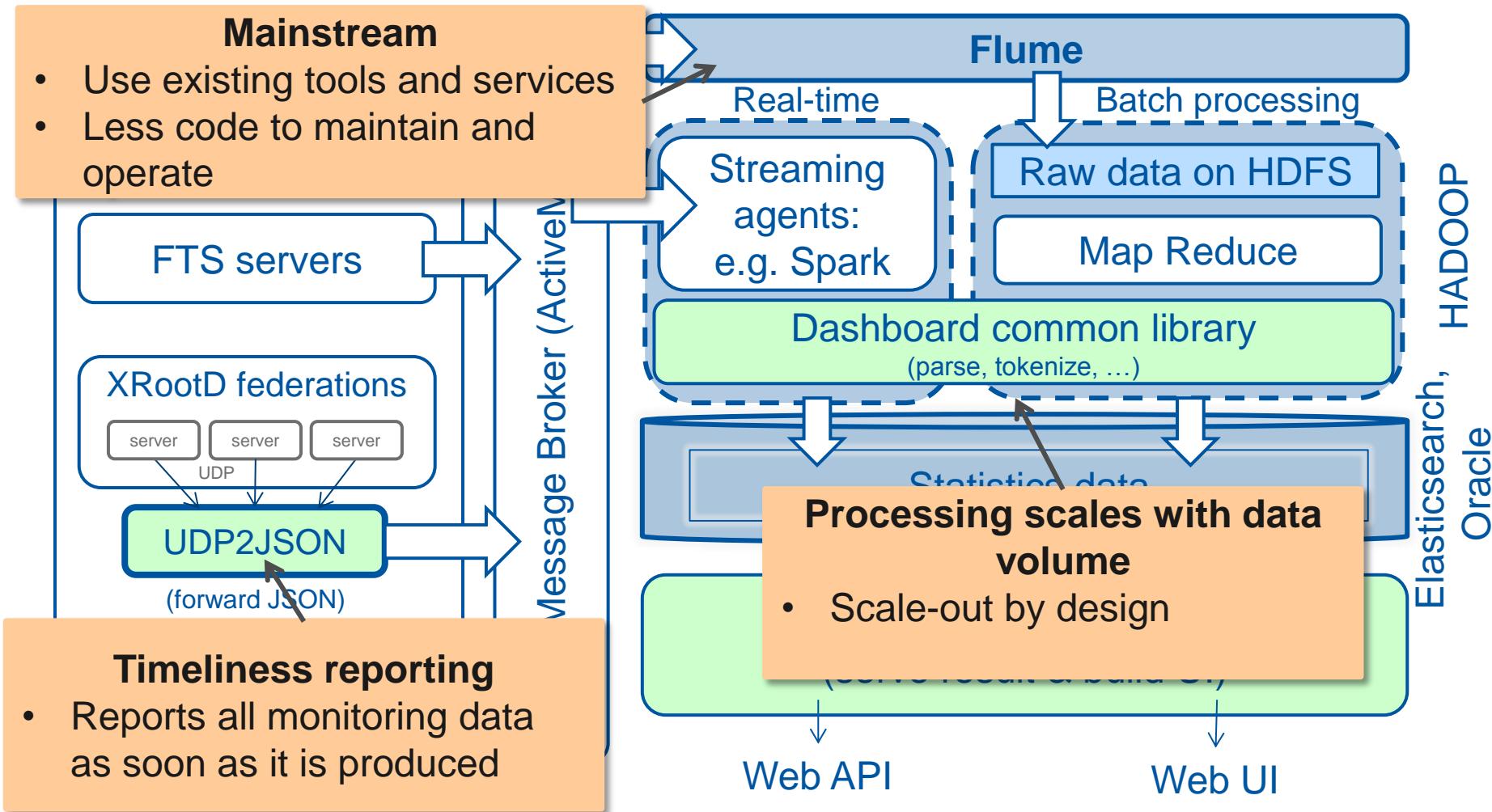
From the book:

Big Data
Principles and best practices of scalable
realtime data system
By Nathan Marz and James Warren
ISBN: 9781617290343

New *lambda-style* architecture



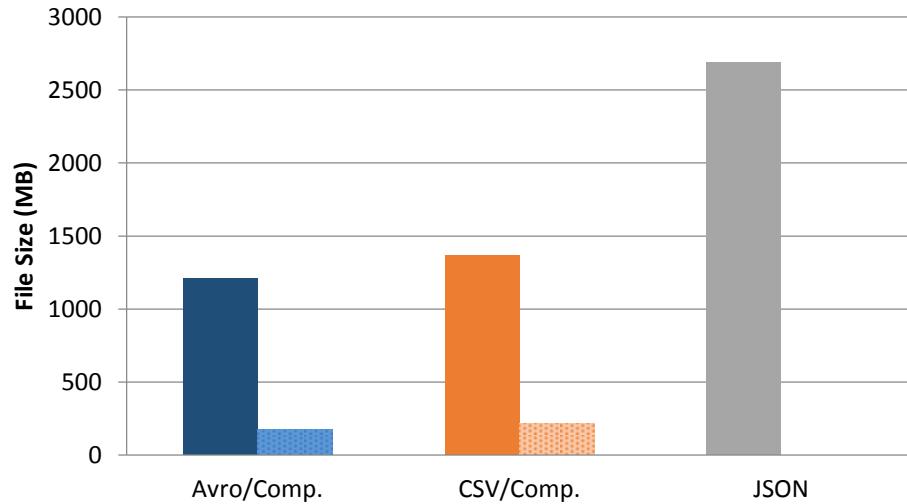
New *lambda-style* architecture



Now, make your choices!

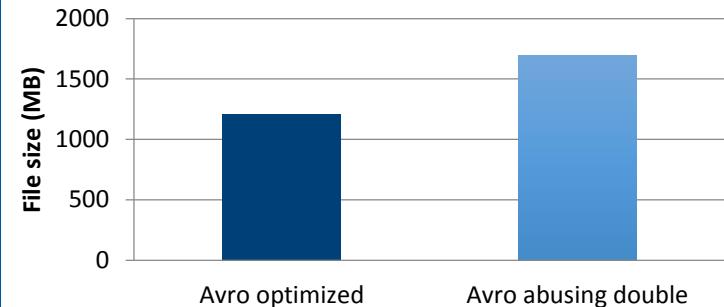
How to store data on HDFS?
Do you need a schema? A serialization lib?

AVRO vs CSV vs JSON for 1 Day FAX data



Is compression a good idea?
Which algorithm?

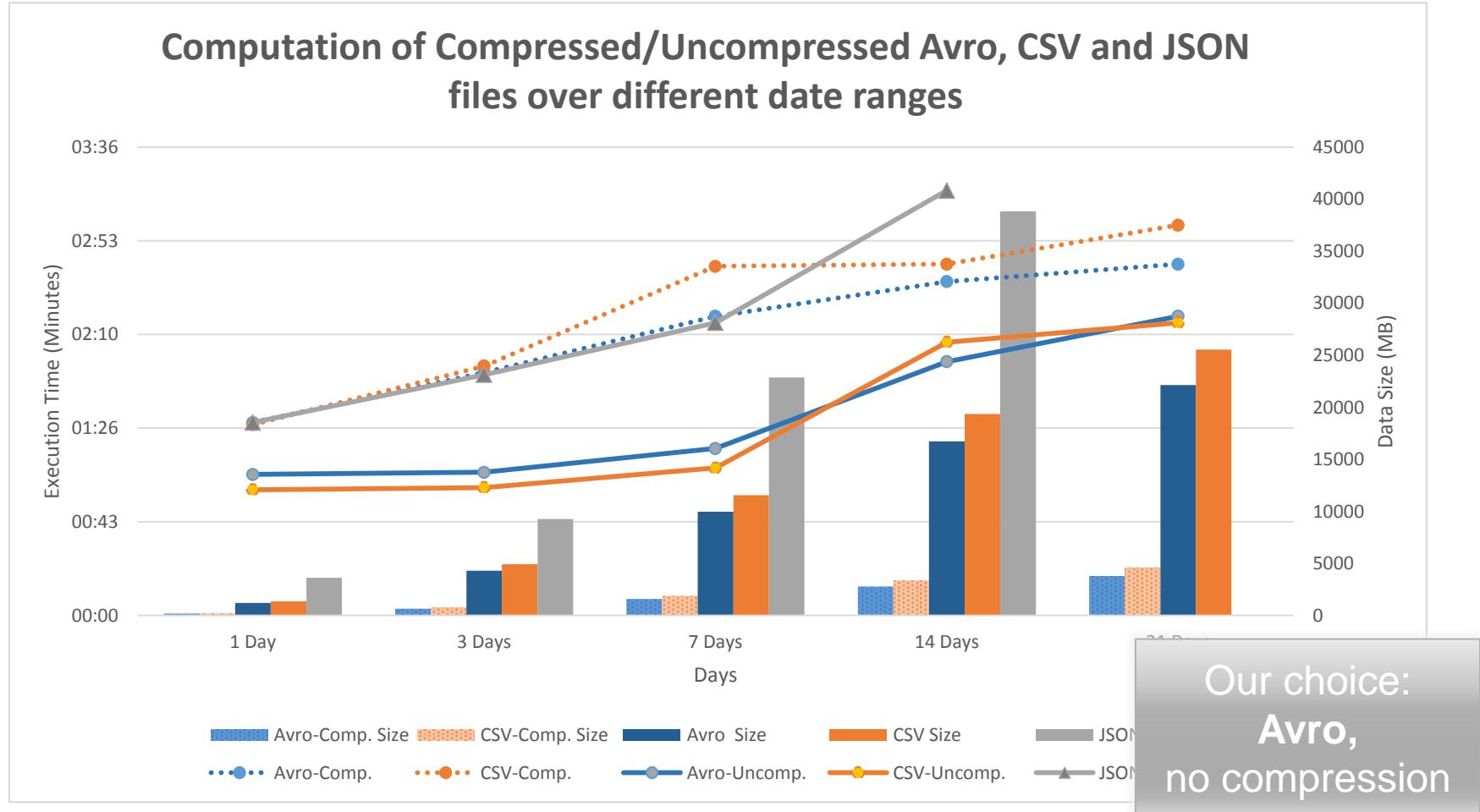
Is your schema good ?



How to partition data?

```
| -- data
|   | -- xrootd
|   |   | -- atlas
|   |   |   | -- 2014
|   |   |   |   | -- 12
|   |   |   |   |   | -- 01
|   |   |   |   |   |   | data.avro
```

Hadoop/MR* at work



* CERN IT-DSS Analytix cluster (Hadoop/CDH 5.1), 8 nodes 32 cores/64GB, 7 nodes 4 cores/8GB

Real-Time layer: more choices!

- Streaming frameworks
 - Distributed (á la Hadoop)
 - Basic processing (filter, join, etc.)
 - Apache Storm, Apache Spark (Streaming), Apache Samza
- In-memory engines
 - Advanced processing (with DSLs like SQL or time interval algebra)
 - Esper, VoltDB
- We are currently investigating:
 - Apache Spark for streaming, potentially with Esper as operator for complex task



Conclusion

- Hadoop ecosystem fits well with WLCG monitoring
 - Scales with data volume
 - For Dashboard, imperative MR easier than SQL
 - Simpler architecture, less custom code and services
- The lambda idea enables “*live-views*” on WLCG
 - Keep all the data
 - Batch + streaming for fast and reliable processing
 - Ongoing investigation on Spark as ~ *uniform* platform
 - MR to Spark for batch trivial, done already (thanks to common lib)
- Plan: towards production
 - Migrate XRootD (FAX, AAA federations) and HTTP dashboards to new architecture by summer 2015
 - Focus then on FTS and DDM dashboard migration

