
CERN

CERN IT Monitoring: migration to *big data* technologies

Luca Magnoni, for the MONIT team

CERN IT Monitoring: what we do

- Provide a common infrastructure to Measure, Collect, Transport, Visualize, Process and Alarm
  - Metrics and Logs
  - for CERN Data Centres, IT and WLCG Services
- http://cern.ch/monitdocs

Some Monitoring Numbers

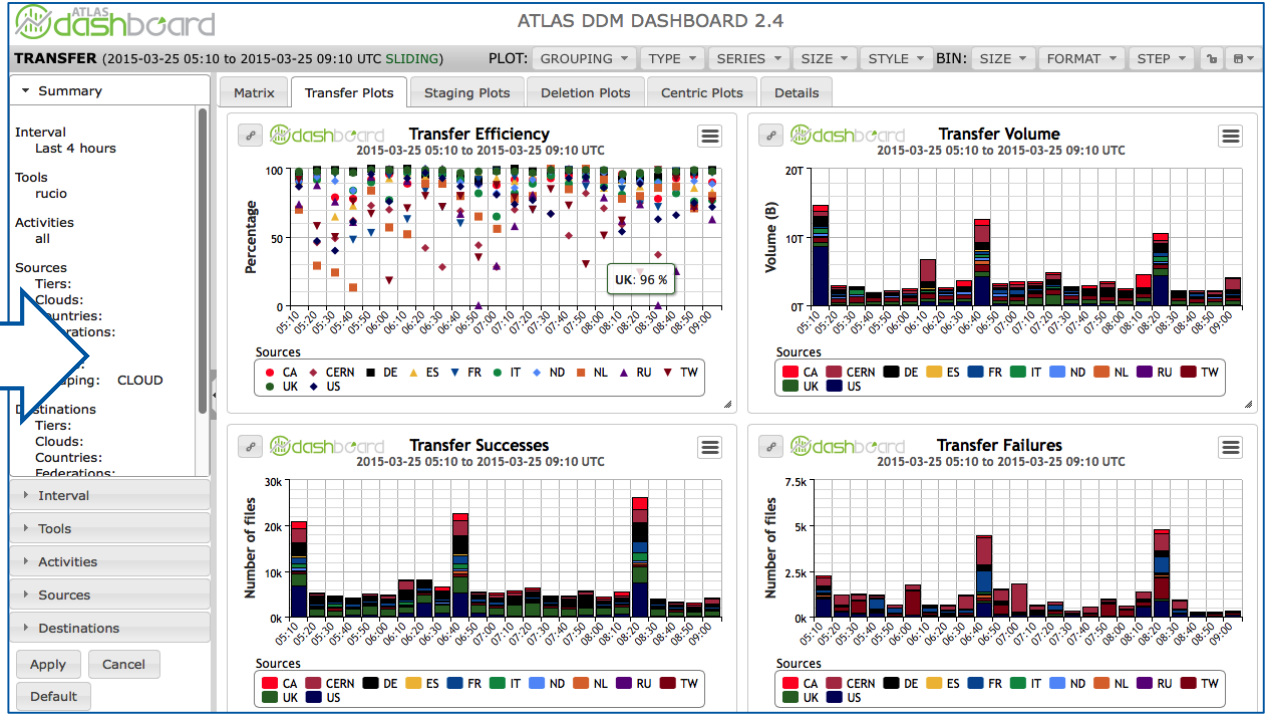
- ~ 100 Data Producers
- ~ 3 TBs / day
- ~ 80 KHz average rate, spikey workload
- > 100 user dashboards

Migration stories

- Old monitoring tools and services are (being) moved to the new common infrastructure
  - Data centre monitoring from Lemon to Collectd
  - System and Service Logs integration
  - WLCG/Experiments Dashboards replacement
    - *relational* part was here

Dashboard example

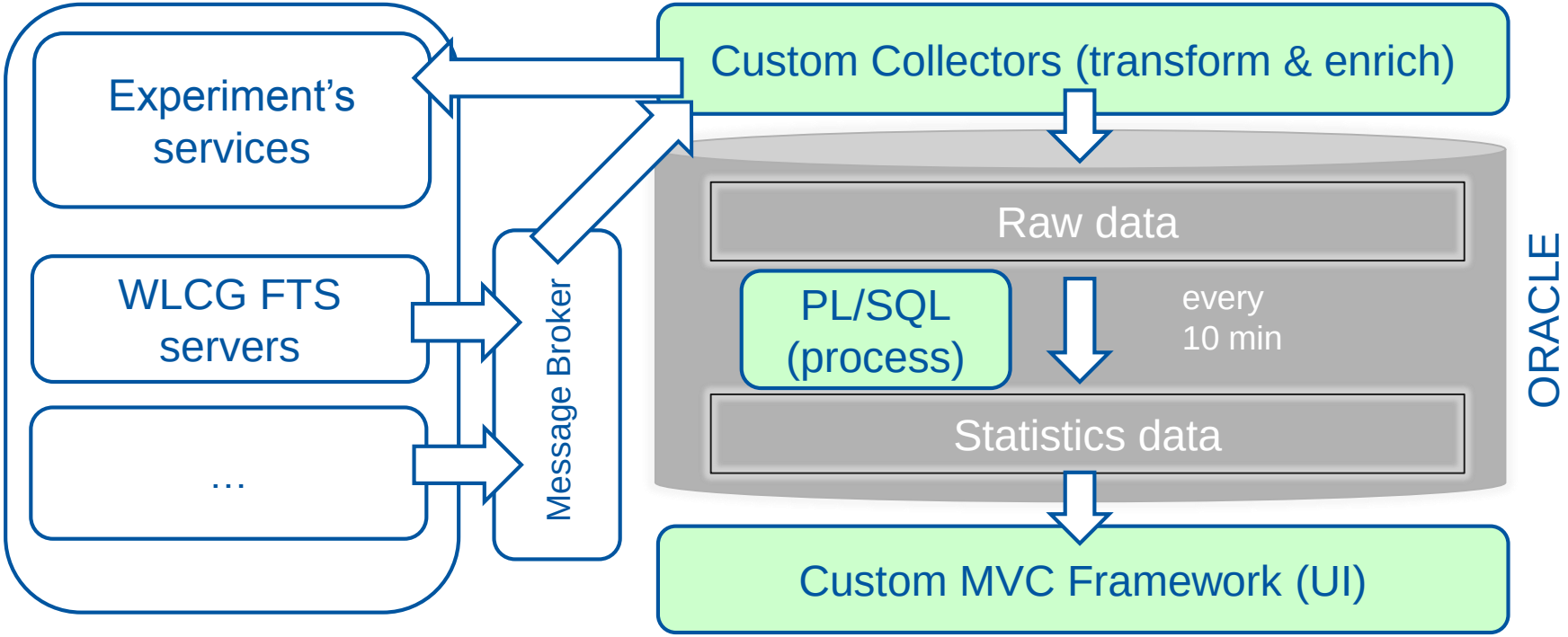
{\"unique_id\":\"30fbed9e-975b-11e4-9717-5b82e4a9beef-4ef6f2e\", \"file_lfn\":\"/store/mc/Fall13dr/QCD_Pt-80to120_Tune4C_13TeV_pythia8/GEN-SIM-RAW/castor_tsg_PU40bx25_POSTLS162_V2-v1/20000/6C4FDD71-1884-E311-9FC2-90E6BA0D09A2.root\", \"file_size\":\"4034966171\", \"start_time\":\"1426860046\", \"end_time\":\"1426863860\", \"read_bytes\":\"0\", \"read_operations\":\"0\", \"read_min\":\"0\", \"read_max\":\"0\", \"read_average\":\"0.000000\", \"read_sigma\":\"0.000000\", \"read_single_bytes\":\"0\", \"read_single_operations\":\"0\", \"read_single_min\":\"0\",



Dashboard workflow

- Data gathering
  - from experiment's DB, message-brokers, HTTP endpoints, etc.
- Validation and Transformation
  - formatting, filtering, extraction, enrichment
- Processing
  - statistics computation, time-based aggregations
- Visualization
  - custom web dashboards

Old Oracle-based solution

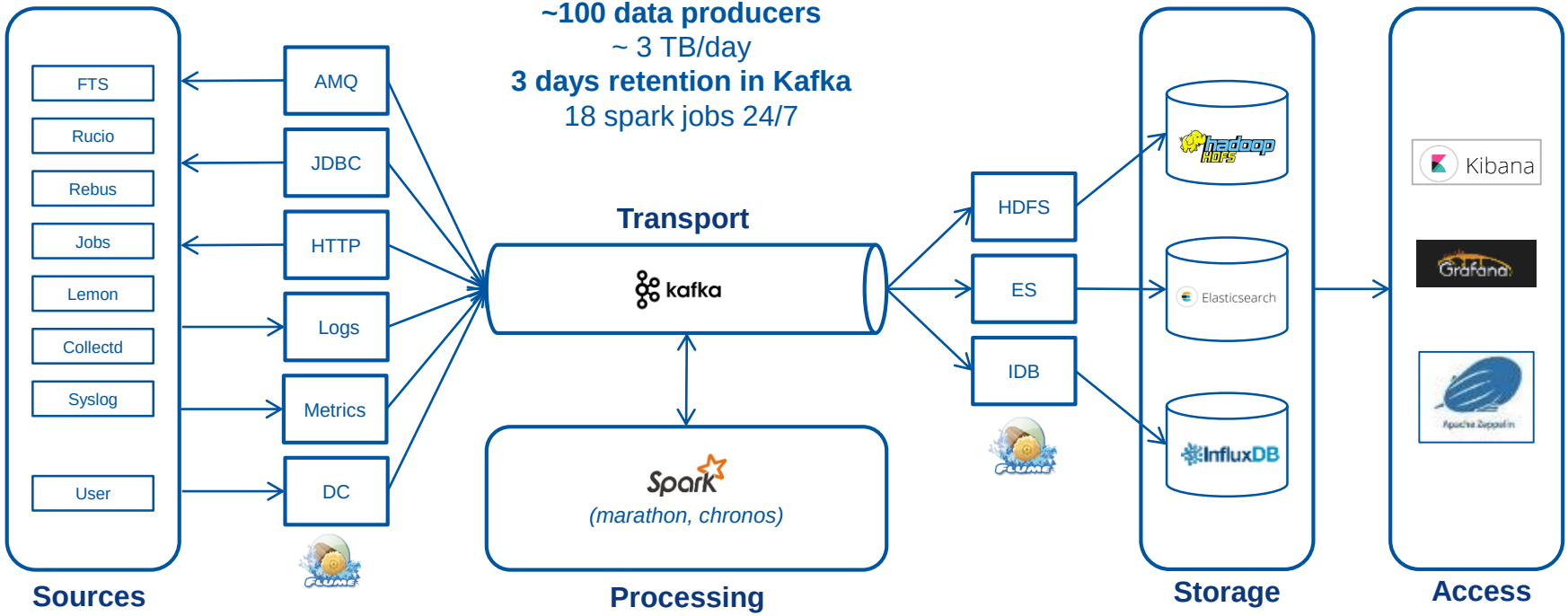


A common architecture

Common Technologies

- **Collectd** for measuring
- **Flume** as collection agent
- **Kafka** as transport layer
- **Spark** as processing framework
- **HDFS** as *cold* storage
- **Elasticsearch** and **InfluxDB** as *hot* storage
- **Kibana, Grafana, Zeppelin** to explore and visualize

The MONIT Architecture

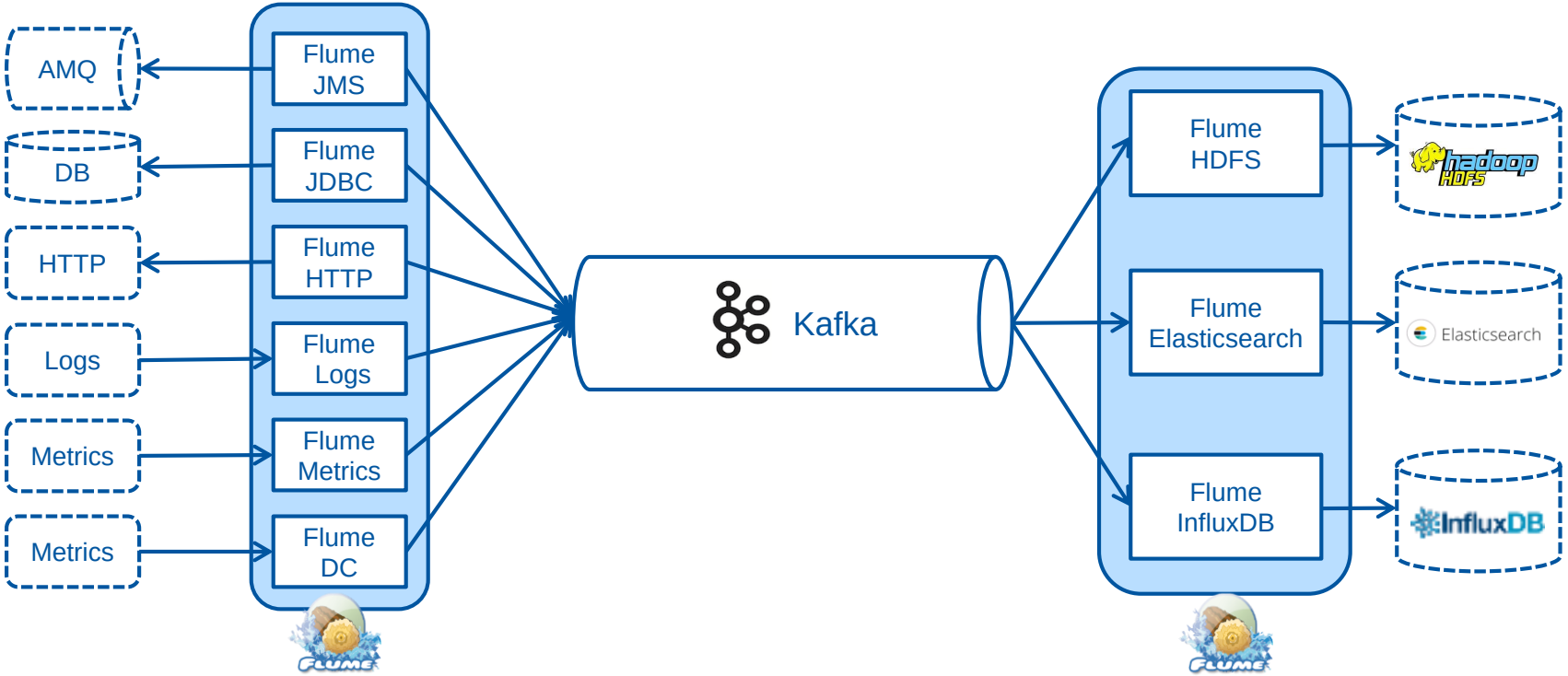


Data Collection and Transport

Apache Flume as collector agent

- One tool, many input/output options
- Push and pull models
- Guaranteed delivery (transactions)
- Horizontal scalability
- Support data interceptor/morphlines
  - ensures common data format

Apache Flume as collector agent

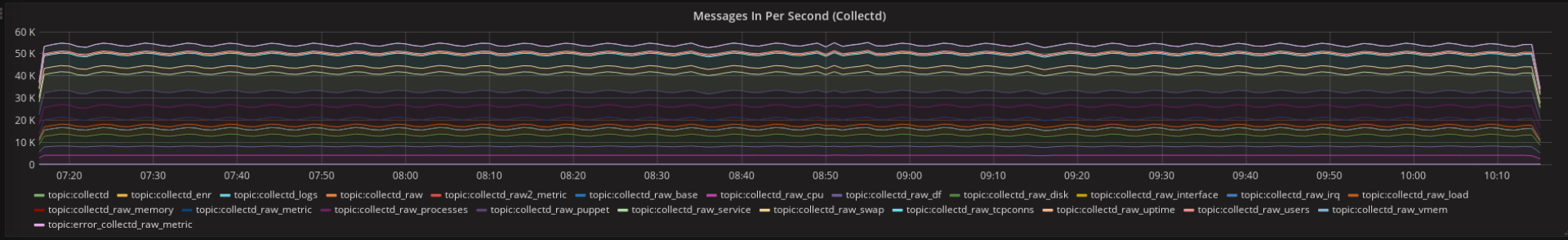


Apache Kafka

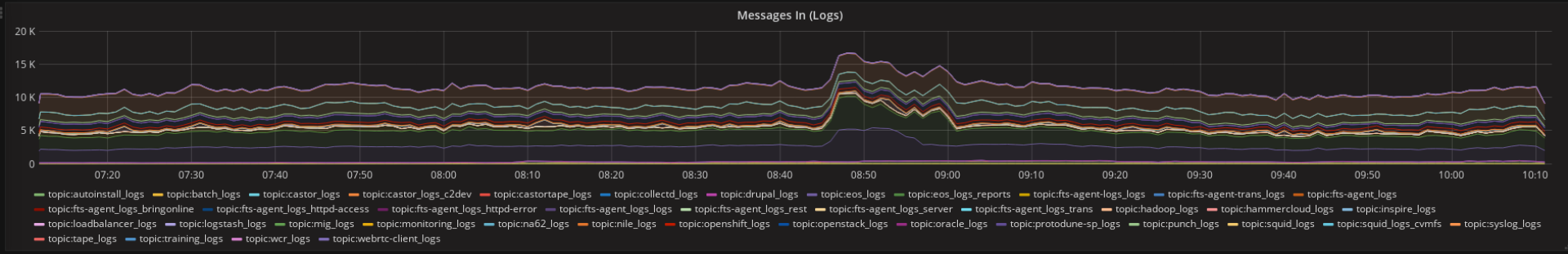
- Fault-Tolerant / Distributed / High-Throughput messaging-like system
- Decouple producers and consumers
- Reliable data buffer (72 hours)
  - proved useful in many situations
- Solid core of the the transport layer

Data Range: one_week | Environment: production | Hostname: All

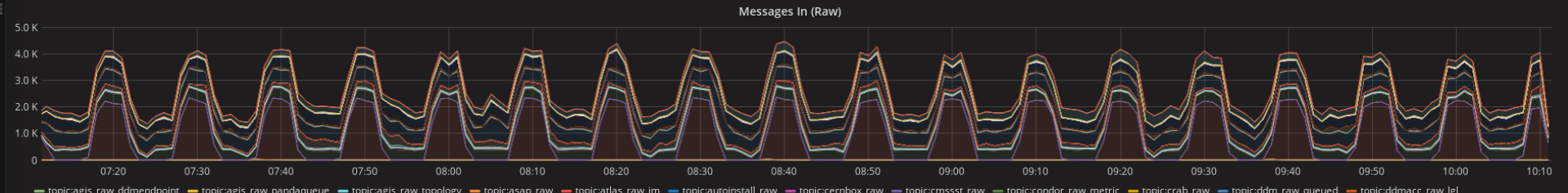
Topics (Raw Collectd)



Topics (Raw Logs)



Topics (Raw Metrics)



A note on data access latency

- HDFS has access *latency*
  - i.e. no access to *fresh* data
- Kafka enables on-the-fly access to all monitoring information
- Plays a key role in serving data for the processing layer

Processing

The need **for** data processing

- **Data enrichment**
  - Enrich monitoring metrics with data from multiple sources (i.e. join)
- **Data transformation**
  - Compute status of systems/services based on other metrics
  - Data aggregation over time or other dimensions (e.g. compute a cumulative metric for a set of machines hosting the same service)
- **Data correlation**
  - Detect anomalies and failures correlating data from multiple sources (e.g. datacentre topology-aware alarms)

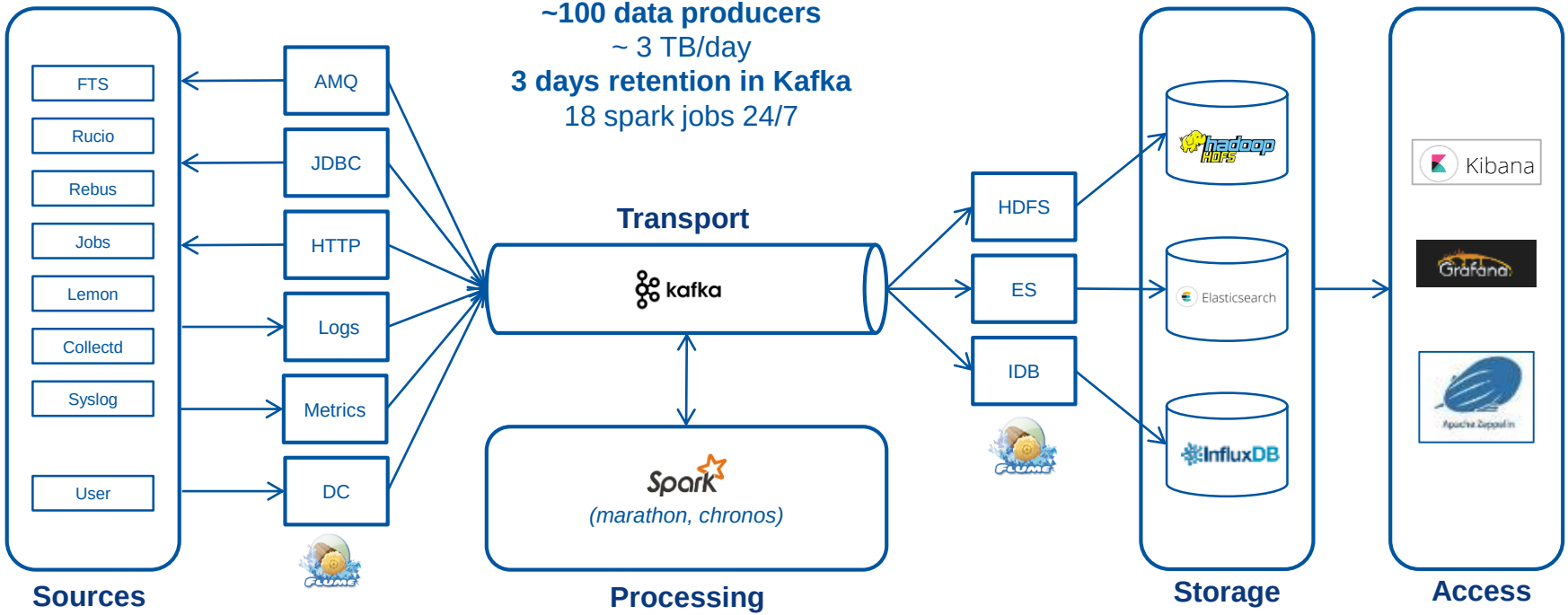
The needs **of** data processing

- Reliable and scalable job execution (Spark)
- Job orchestration (Mesos/Marathon/Hadoop)
- Lightweight deployment (Docker)

Apache Spark

- *Modern* distributed processing framework
  - It runs on Hadoop/YARN, Mesos or standalone clusters
- Evolves the MapReduce paradigm
  - rich directives
  - promotes in-memory/iterative computation
- Supports Batch and Stream processing

Apache Spark for Monitoring



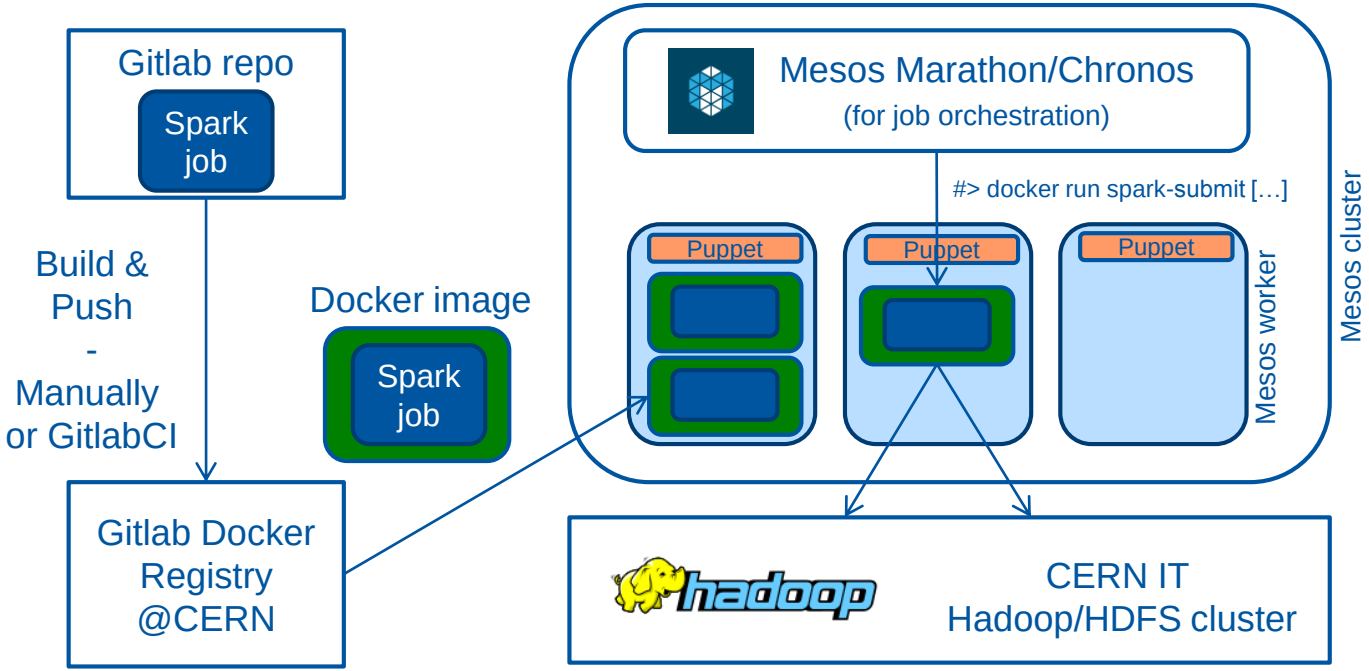
A note on Stream & Batch analysis

- Different processing workflows
  - fast low-latency streaming / slow high-volume batch
  - typically on different frameworks too
- It's a *big data* difference
  - DB is both "hot" and "cold" access
- From user's perspective, it can be inconvenient
  - code duplication
  - things should "just work the same" on fresh and historical data

Spark Structured Streaming

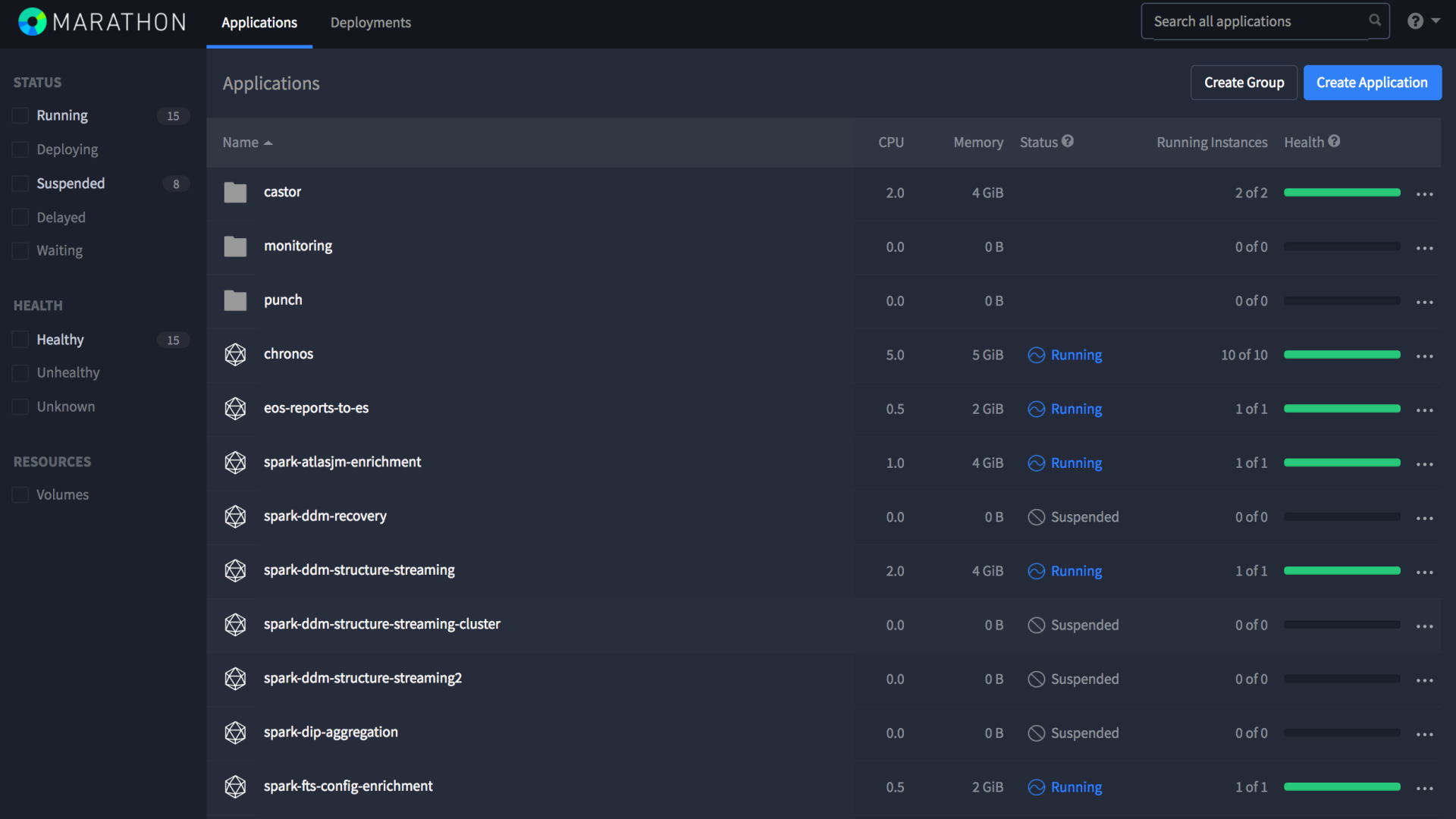
- Promoted stable in Spark 2.2.0
- Dataframe/Dataset can be both static and streaming
  - processing logic/code is the same
- Major simplification
  - many built-in features, resulting in simpler jobs
- In practice, allows the same job to process the same way data from Kafka and HDFS

Monitoring Processing Platform



Job Deployment and Orchestration

- Mesos cluster
  - Distributed and fault-tolerant execution of c*ommands* on workers
- Marathon for long-living processes (e.g. streaming jobs)
  - Start/stop/restart/scale a process
  - Useful web UI for operation/monitoring
- Chronos for recurrent execution (e.g. batch jobs)
  - Support job DAGs (e.g. jobs triggered by the completion of other jobs)
- Native support for containers (e.g. Docker)
  - *command* is executed launching a container from an image
- Gitlab CI pipeline on merges:
  - Build Software build / Build Docker image and push to gitlab registry
- Technology independent solution (e.g. support Spark and other)


MARATHON
Applications
Deployments
Search all applications
STATUS
Running 15
Deploying
Suspended 8
Delayed
Waiting
HEALTH
Healthy 15
Unhealthy
Unknown
RESOURCES
Volumes
Applications
Create Group
Create Application
Name
CPU
Memory
Status
Running Instances
Health
castor 2.0 4 GiB 2 of 2
monitoring 0.0 0 B 0 of 0
punch 0.0 0 B 0 of 0
chronos 5.0 5 GiB Running 10 of 10
eos-reports-to-es 0.5 2 GiB Running 1 of 1
spark-atlasjm-enrichment 1.0 4 GiB Running 1 of 1
spark-ddm-recovery 0.0 0 B Suspended 0 of 0
spark-ddm-structure-streaming 2.0 4 GiB Running 1 of 1
spark-ddm-structure-streaming-cluster 0.0 0 B Suspended 0 of 0
spark-ddm-structure-streaming2 0.0 0 B Suspended 0 of 0
spark-dip-aggregation 0.0 0 B Suspended 0 of 0
spark-fts-config-enrichment 0.5 2 GiB Running 1 of 1

spark-ddm-structure-streaming

Running (1 of 1 instances)

1 Healthy (100%) 0 Unhealthy 0 Unknown

Scale Application | Restart | ⚙

Instances | Configuration | Debug

Current Version - 16/11/2017, 10:57:36

Refresh | Edit

ID /spark-ddm-structure-streaming

Command /usr/lib/spark/bin/spark-submit --driver-class-path /monit/spark-ddm-aggregation/spark-ddm-aggregation-assembly-1.1.jar --driver-java-options "-XX:+UseG1GC" --conf spark.streaming.unpersist=true --conf park.serializer="org.apache.spark.serializer.KryoSerializer" --principal monitops@CERN.CH --driver-memory 3g --keytab /etc/monit/monitops.keytab --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.1.1 --class ch.cern.monitoring.DDMAggregationApplication --master local[*] --conf spark.ui.port=$PORT0 /monit/spark-ddm-aggregation/spark-ddm-aggregation-assembly-1.1.jar --time-window "1 minute" --watermark "7 hours" --output-topic ddm_agg_transfer --checkpoint hdfs://analytix/project/monitoring/checkpoint/spark-ddm-aggregation --input-brokers-url "monit-kafka.cern.ch:9092" --output-brokers-url "monit-kafka.cern.ch:9092" --starting-offset latest --output-mode update --log-level INFO

Constraints Unspecified

Dependencies Unspecified

Labels Unspecified

Resource Roles *

Container


```
{
  "type": "DOCKER",
  "volumes": [
    {
      "containerPath": "/etc/hadoop/conf",
      "hostPath": "/etc/hadoop/conf",
      "mode": "RO"
    },
    {
      "containerPath": "/usr/lib/spark",
      "hostPath": "/usr/lib/spark2",
```



```
    "docker": {
      "image": "gitlab-registry.cern.ch/monitoring/spark-ddm:stable",
      "network": "HOST",
      "portMappings": [],
      "privileged": false,
      "parameters": [
        {
          "key": "rm",
          "value": "true"
        }
      ],
      "forcePullImage": true
    }
```

User model: ~ *server-less*

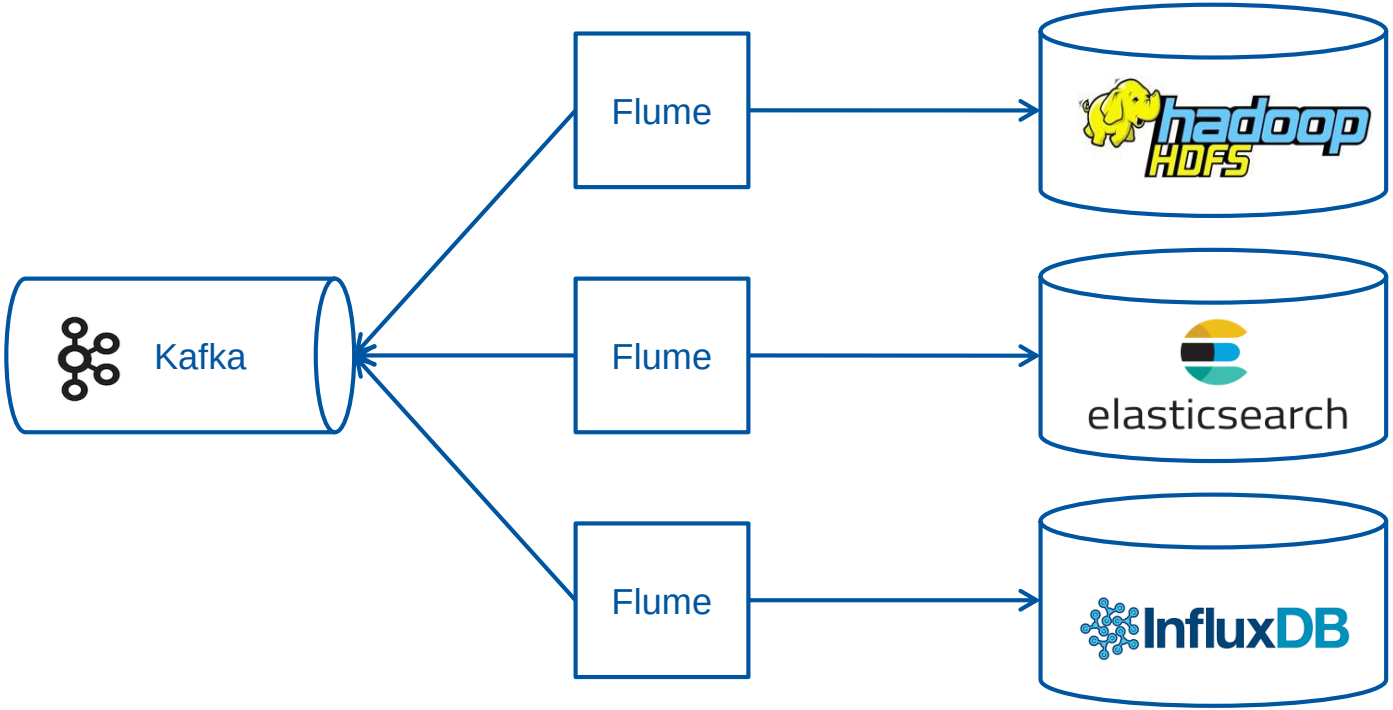
- User care only for the processing logic
  - PL/SQL is a ~ AWS Lambda …
- Monitoring infrastructure provides a fault-tolerant and fully-orchestrated processing environment
  - *Docker for job encapsulation*
  - *Mesos for orchestration*
  - *CERN IT Hadoop for execution*

A note on data processing

- 18 running jobs
  - 14 streaming (24/7), 4 batch (~ daily)
- 4 developed by users
- User-contract defined by monitoring data schema
- Kafka-only interaction proved a good choice
- Prefer idempotent operations
  - Use document ID (or time) to allow deduplication

Storage

Storage


Kafka
Flume
Flume
Flume
hadoop
HDFS
elasticsearch
InfluxDB

Storage

- HDFS for long-term archive
  - Data kept ~ forever (limited to resources)
- Elasticsearch (ES) for data exploration and discovery
  - Data kept for 1 month
- InfluxDB for time-series dashboards
  - Automatic down-sampling, aggregated data kept for ~ 5 years

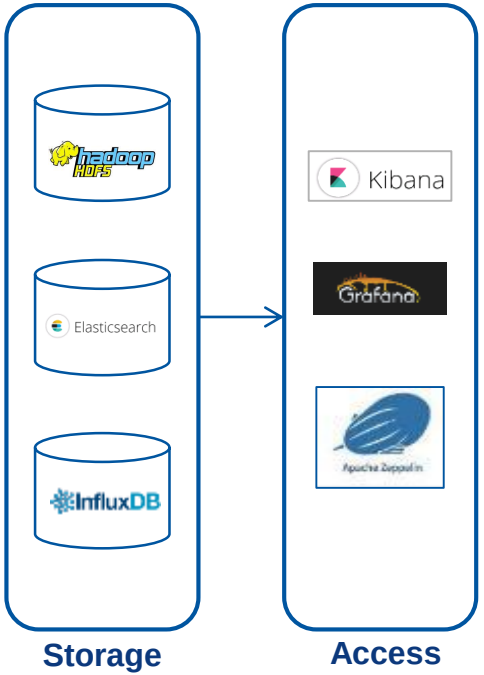
Storage workflow

- All data in HDFS
  - `/project/monitoring/archive/*/*/*/2017/11/21/…`
  - Compressed JSON (daily compaction in 512 MB files)
  - Parquet for Collectd data
- Selected data sets in InfluxDB and/or ES
  - Using common monitoring schema metadata to route where data is written
  - InfluxDB: from IT DBOD service, several instances
    - More on InfluxDB for Monitoring @ DBOD Workshop
  - ES: two instances from IT Central ES service

Visualization

Technologies

- Grafana for user dashboards
- Kibana for data exploration
- Zeppelin for interactive notebooks



Grafana

- Open-source platform for dashboards
- Support multiple backends
  - e.g. Elasticsearch and InfluxDB
- Advanced visualization features
  - Template / Ad-hoc filters / Autocompletion
  - Advanced query syntax
  - Alarms

monit-grafana.cern.ch

- CERN SSO integrated
- Access to all MONIT data
- Possibility to create custom views mixing metrics/sources
  - e.g. service and data centre monitoring
- Users have control
  - Organizations with roles (Editor, View, …)
  - Used by WLCG experiments, service managers, etc.

ATLAS DDM Dashboard (Latest data)

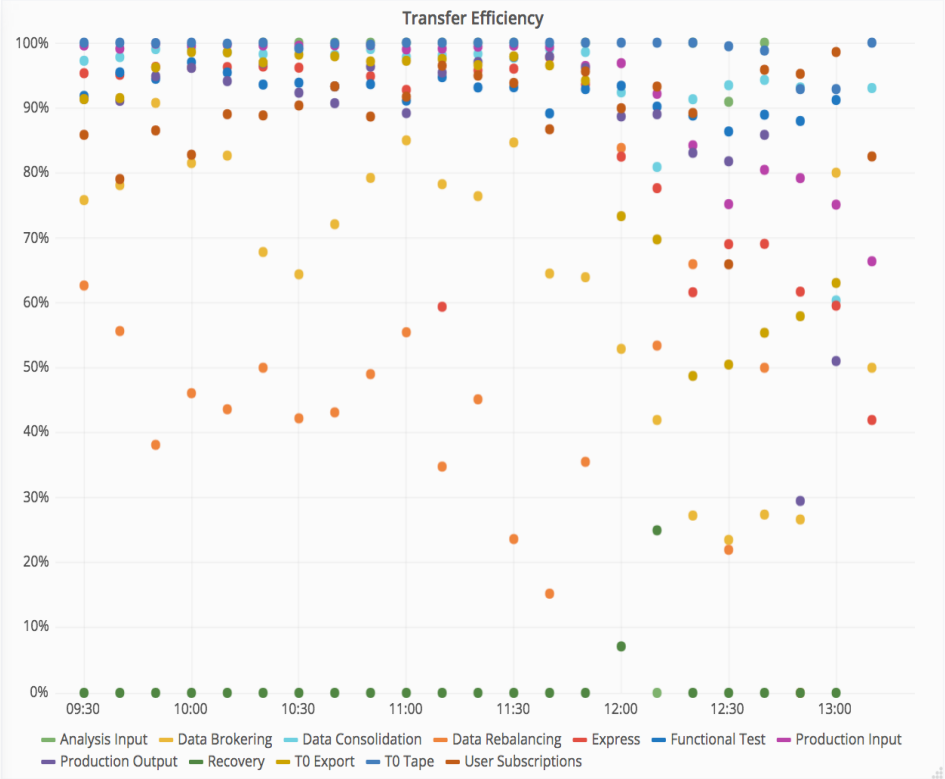
Zoom Out | Last 4 hours UTC

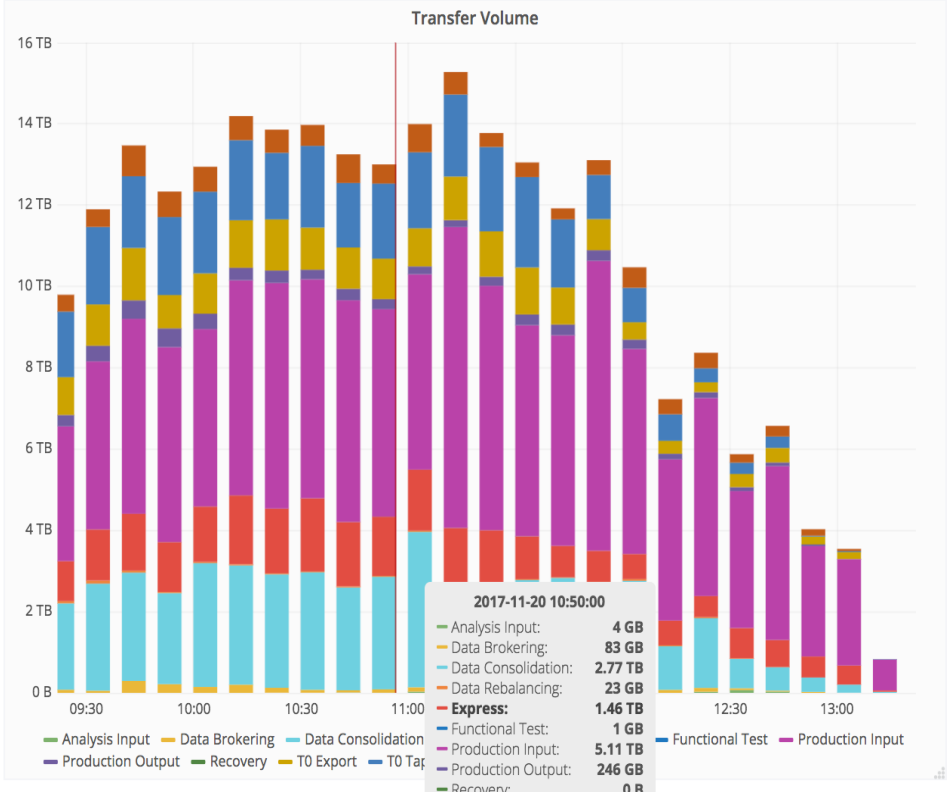
Group by: activity | Binning: auto

Activity: Analysis Input + Data Brokering + Data Consolidation + Data Rebalancing + Deletion + Express + Functional Test + Production Input + Production Output + Recovery + T0 Export + T0 Tape + User Subscriptions + default + Staging | Source country: All

Source site: All | Destination country: All | Destination site: All | Filters: +

Transfers





Wrap Up

On CERN IT Hadoop

- Very positive feedback on the service
  - Prompt support, collaboration and expertise
  - More *batch* use cases are coming from monitoring users
- Whish List
  - Faster software-release cycle (e.g. Spark) ?
  - Cluster monitoring may be useful for users
  - More visual analytics / Tableau-like software?

Summary

- *Big data* technologies offer a number of new ways to gather, process, store data
  - Build a stack, take the best from each
- Mainstream technologies evolve fast
  - Stay at speed, profit from community
- CERN IT monitoring relies on several of those technologies for its production workflow

Reference and Contacts

- Docs: cern.ch/monitdocs
- Support: cern.ch/monit-support


CERN