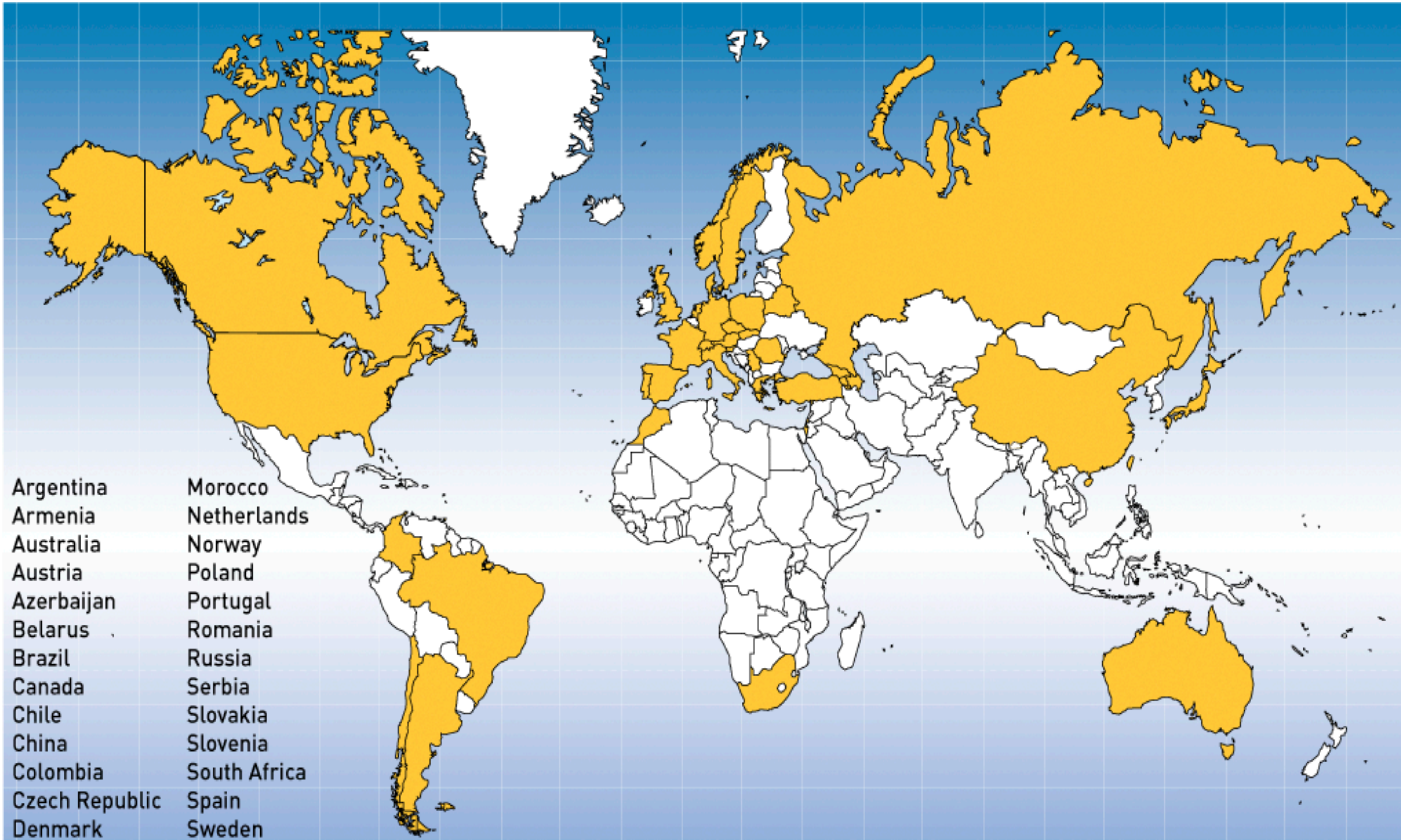
Use of the Hadoop structured storage tools for the ATLAS EventIndex event catalogue

Andrea Favareto
Università degli Studi di Genova & INFN Genova
On behalf of the ATLAS Collaboration

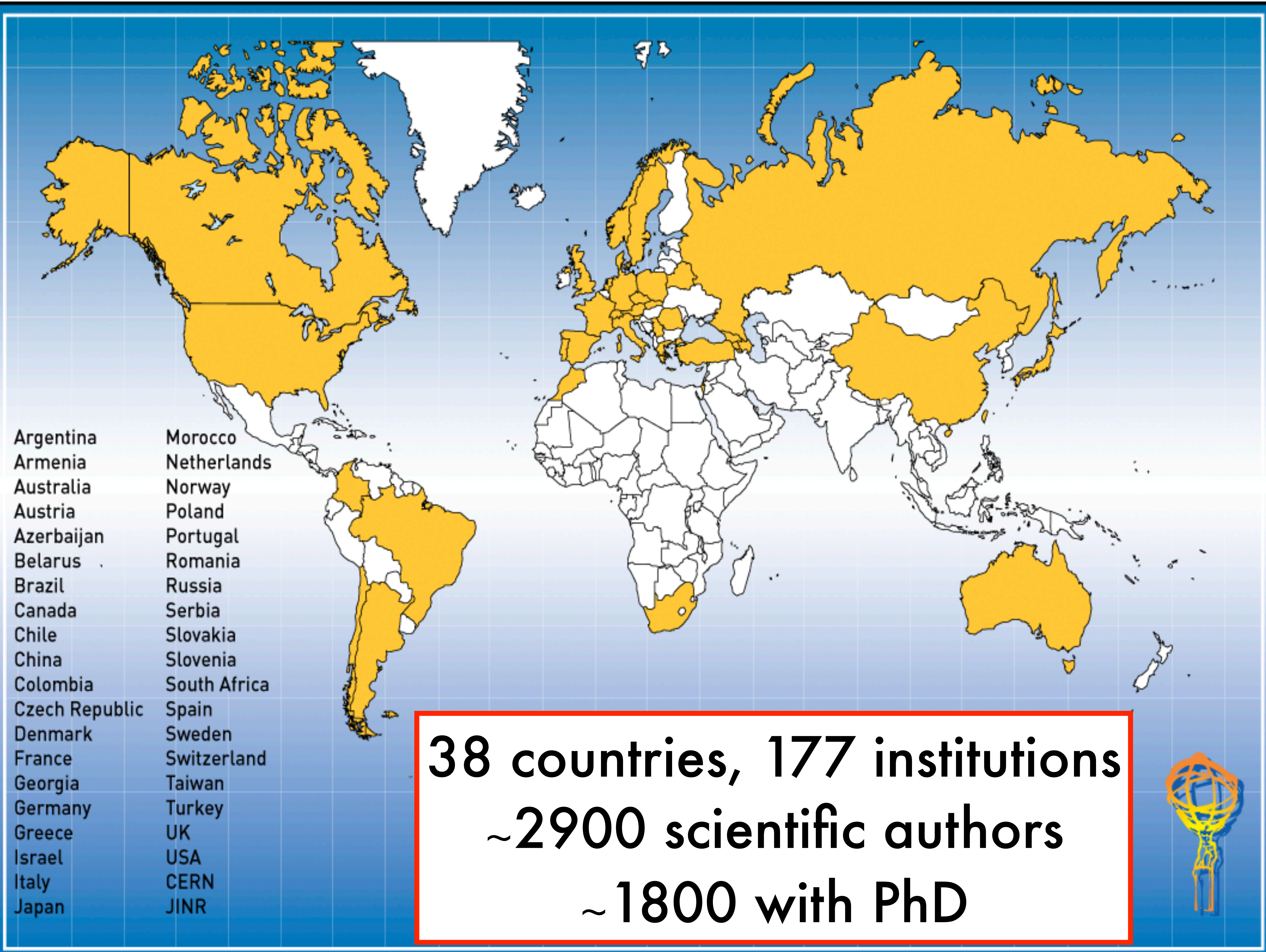
2nd Oct 2015 - NEC'2015



Argentina
Armenia
Australia
Austria
Azerbaijan
Belarus
Brazil
Canada
Chile
China
Colombia
Czech Republic
Denmark
France
Georgia
Germany
Greece
Israel
Italy
Japan
Morocco
Netherlands
Norway
Poland
Portugal
Romania
Russia
Serbia
Slovakia
Slovenia
South Africa
Spain
Sweden
Switzerland
Taiwan
Turkey
UK
USA
CERN
JINR

ATLAS Collaboration

July 2010



Argentina
Armenia
Australia
Austria
Azerbaijan
Belarus
Brazil
Canada
Chile
China
Colombia
Czech Republic
Denmark
France
Georgia
Germany
Greece
Israel
Italy
Japan
Morocco
Netherlands
Norway
Poland
Portugal
Romania
Russia
Serbia
Slovakia
Slovenia
South Africa
Spain
Sweden
Switzerland
Taiwan
Turkey
UK
USA
CERN
JINR

38 countries, 177 institutions
~2900 scientific authors
~1800 with PhD

Challenges of ATLAS distributed computing

- The performance of the Worldwide LHC Computing Grid sites has been outstanding, and is fundamental to ATLAS physics analysis
    - experiments like ATLAS produce large amounts of data
        - 2 billion real and 4 billion simulated events in 2011 and 2012
        - this numbers increase if all reprocessing are taken into account
    - in the current Run2, after the 2013-2014 long shutdown, we are actually facing with:
        - increasing trigger rate to 1kHz (more then twice that for Run1)
        - globally CPU +20%, DISK +15%, TAPE +15% each year

Catalog of events

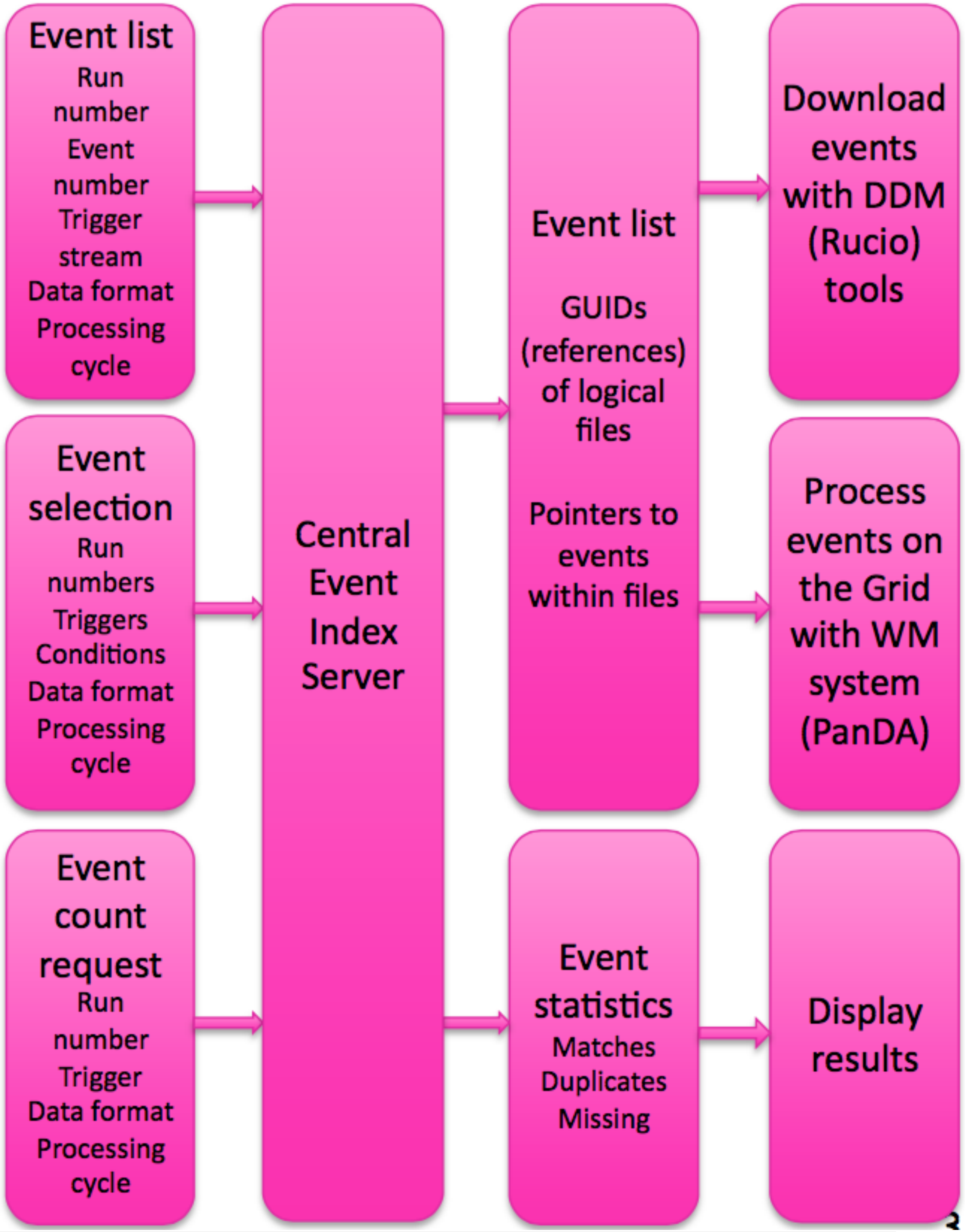
- A catalog of data is needed to meet multiple use cases and search criteria
- A database that contains the reference to the file that includes every event at every stage of processing is necessary to recall selected events from data storage systems
  - in ATLAS an *EventTag* database already exists
    - designed in late 90'
    - Oracle databases with separate tables for each reprocessing cycle. Implementation in Oracle particularly labor-intensive
    - each event is recorded several times, once for each cycle of reconstruction
  - *EventTag* is potentially very useful but very little used, at least in its DB format

The ATLAS EventIndex

- **GOAL**: design a more agile system, using structured storage technologies (NoSQL databases)
  - appropriate to scale for increased data rates (Commodity HW, Fixed cost/unit), scalable
  - cheaper, easier to use and faster for sparse searches over large amounts of data
- The ATLAS EventIndex is a system designed to be a complete catalogue of ATLAS events
  - all events, real and simulated data
  - all processing stages
- Contents
  - event identifiers (run and event numbers, trigger stream, luminosity block etc.)
  - trigger patterns
  - References (pointers) to the events at each processing stage in all permanent files on storage generated by the ATLAS Production System (central productions)

Use cases

- Event picking
  - give me the reference (pointer) to “this” event (or list of events) in “that” format for a giving processing cycle
- Trigger checks and event skimming
  - count, or give me the list of, events passing "this" selection and their references
- Production consistency checks
  - technical checks that processing cycle are complete



Event list
Run number
Event number
Trigger stream
Data format
Processing cycle

Event selection
Run numbers
Triggers
Conditions
Data format
Processing cycle

Event count request
Run number
Trigger
Data format
Processing cycle

Central Event Index Server

Event list
GUIDs (references) of logical files
Pointers to events within files

Event statistics
Matches
Duplicates
Missing

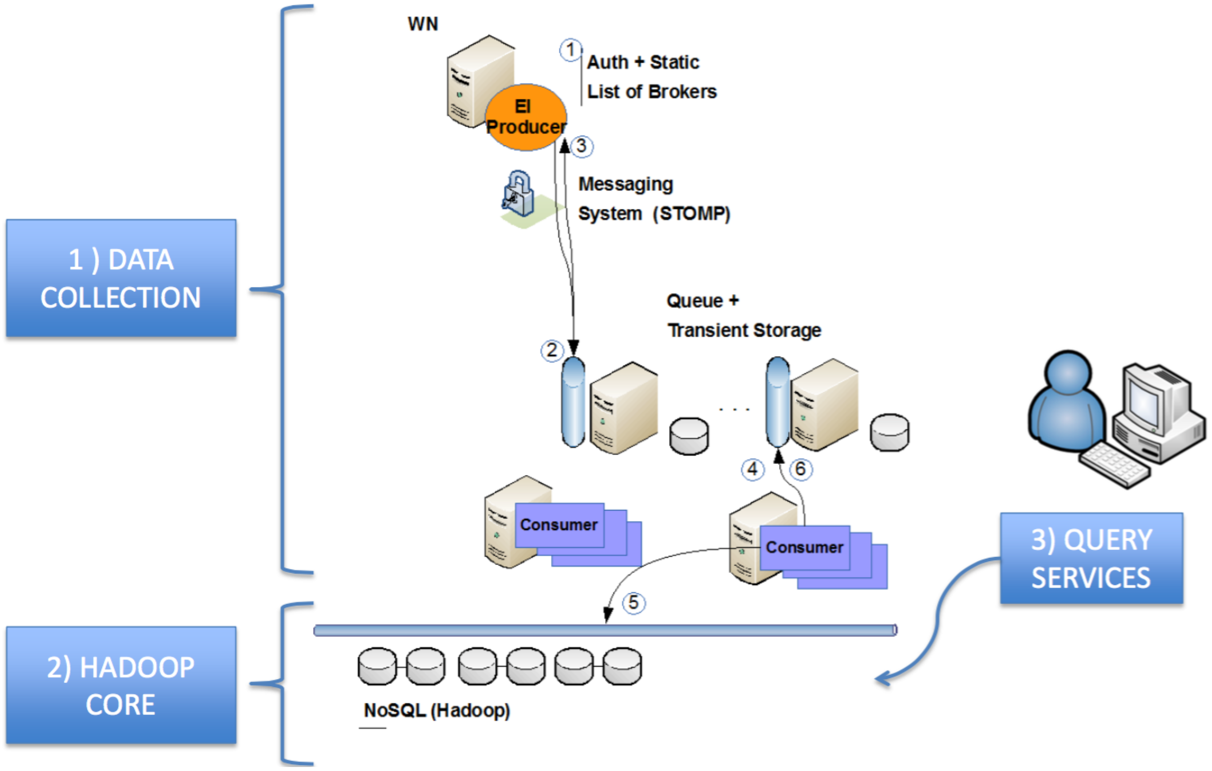
Download events with DDM (Rucio) tools

Process events on the Grid with WM system (PanDA)

Display results

Project breakdown and data flow

- We defined 4 major work areas (or tasks)
  1. data collection
  2. Hadoop core architecture
  3. query services
  4. functional testing and operation; system monitoring



WN

EI Producer

1 Auth + Static List of Brokers

3

Messaging System (STOMP)

Queue + Transient Storage

2

4

6

Consumer

Consumer

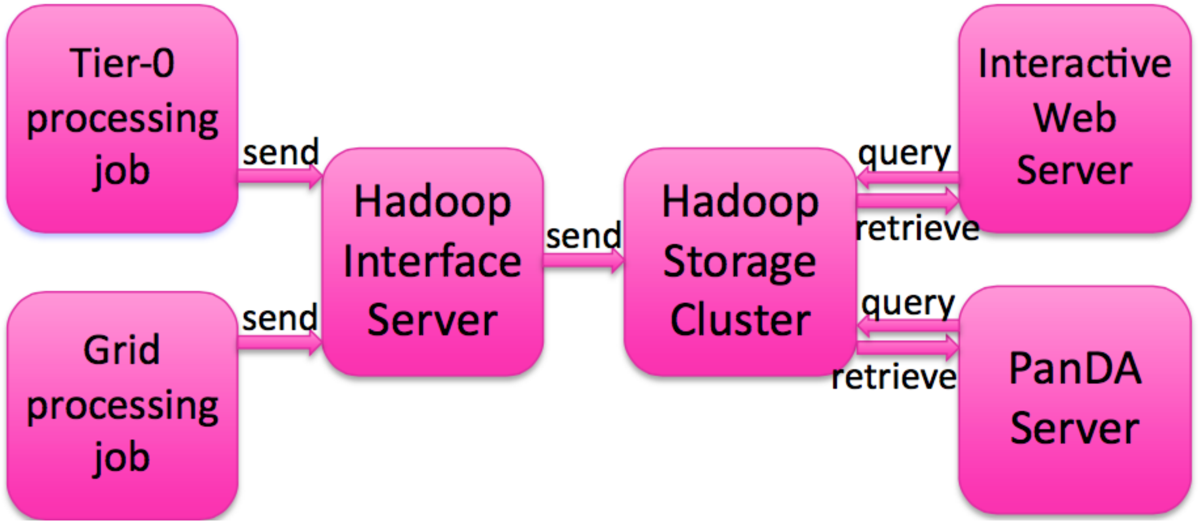
5

NoSQL (Hadoop)

1) DATA COLLECTION

2) HADOOP CORE

3) QUERY SERVICES



Tier-0 processing job

Grid processing job

send

send

Hadoop Interface Server

send

Hadoop Storage Cluster

query

retrieve

query

retrieve

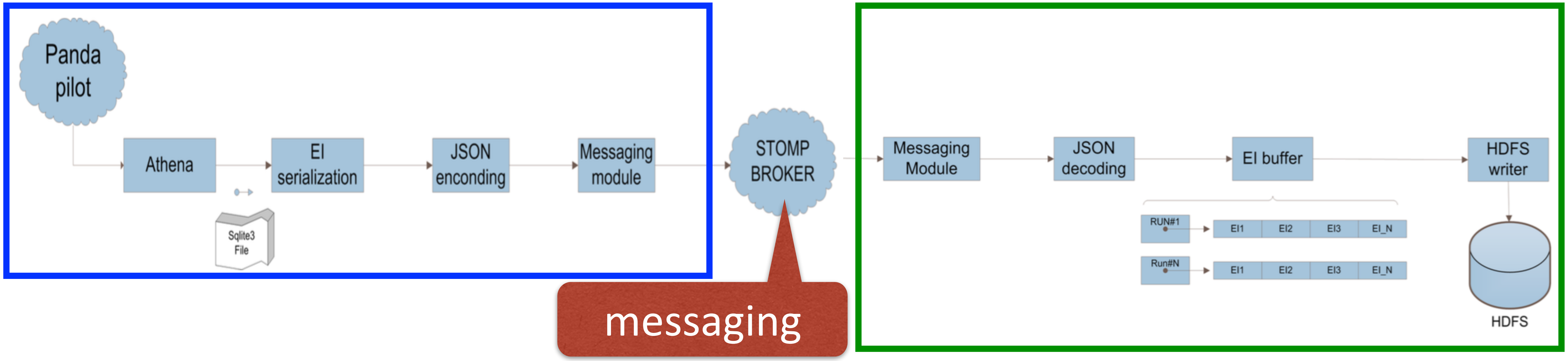
Interactive Web Server

PanDA Server

- Information for the EventIndex is collected from all production jobs running at CERN and on the Grid and transferred to CERN using a messaging system
- This info is reformatted, inserted into the Hadoop storage system and internally catalogued and indexed
- Query services (CLI and GUI) implemented as web services query the EventIndex and retrieve the information

Data collection

- EventIndex Producer: event processing task which can run at Tier-0 (initial reconstruction at CERN) or on Grid sites (downstream processing)
  - send event metadata via ActiveMQ message broker to the Hadoop storage at CERN
- EventIndex Consumer: reads the messages from the message broker
  - organises data into Hadoop MapFile objects
  - does validation task assessing, e.g. dataset completeness
  - flags aborted, obsoleted, invalid data for further action

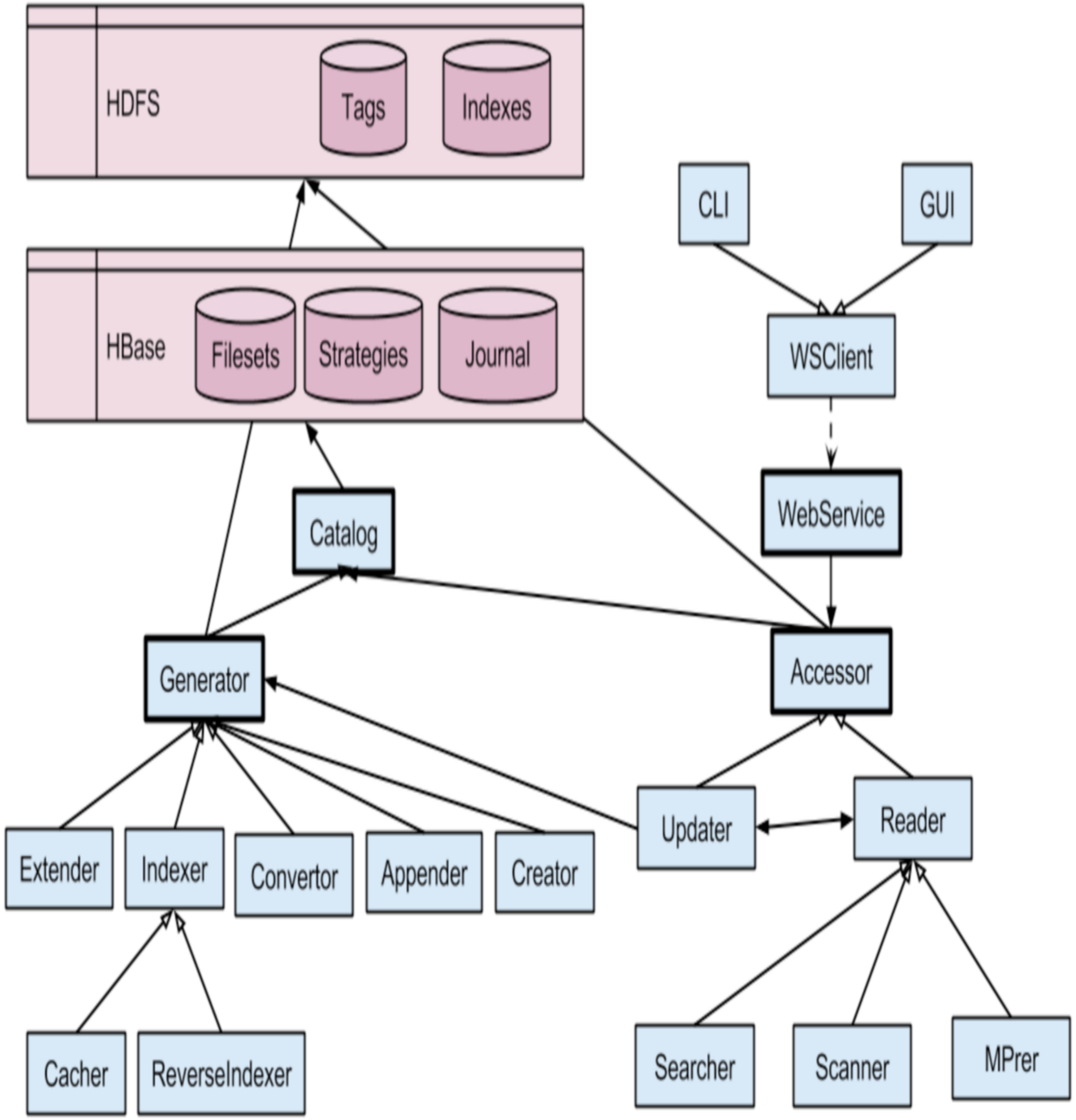


Hadoop core architecture

- **Hadoop** was chosen as the storage technology
  - platform is provided and supported by CERN-IT
  - DDM (Distributed Data Management) project also uses Hadoop
  - plenty of tools to organise the data, index them internally and search them
  - showed satisfactory performance in prototype populated with a year of ATLAS data (1TB of data from EventTag DB in oracle corresponding to full 2011 Tier-0 processing)

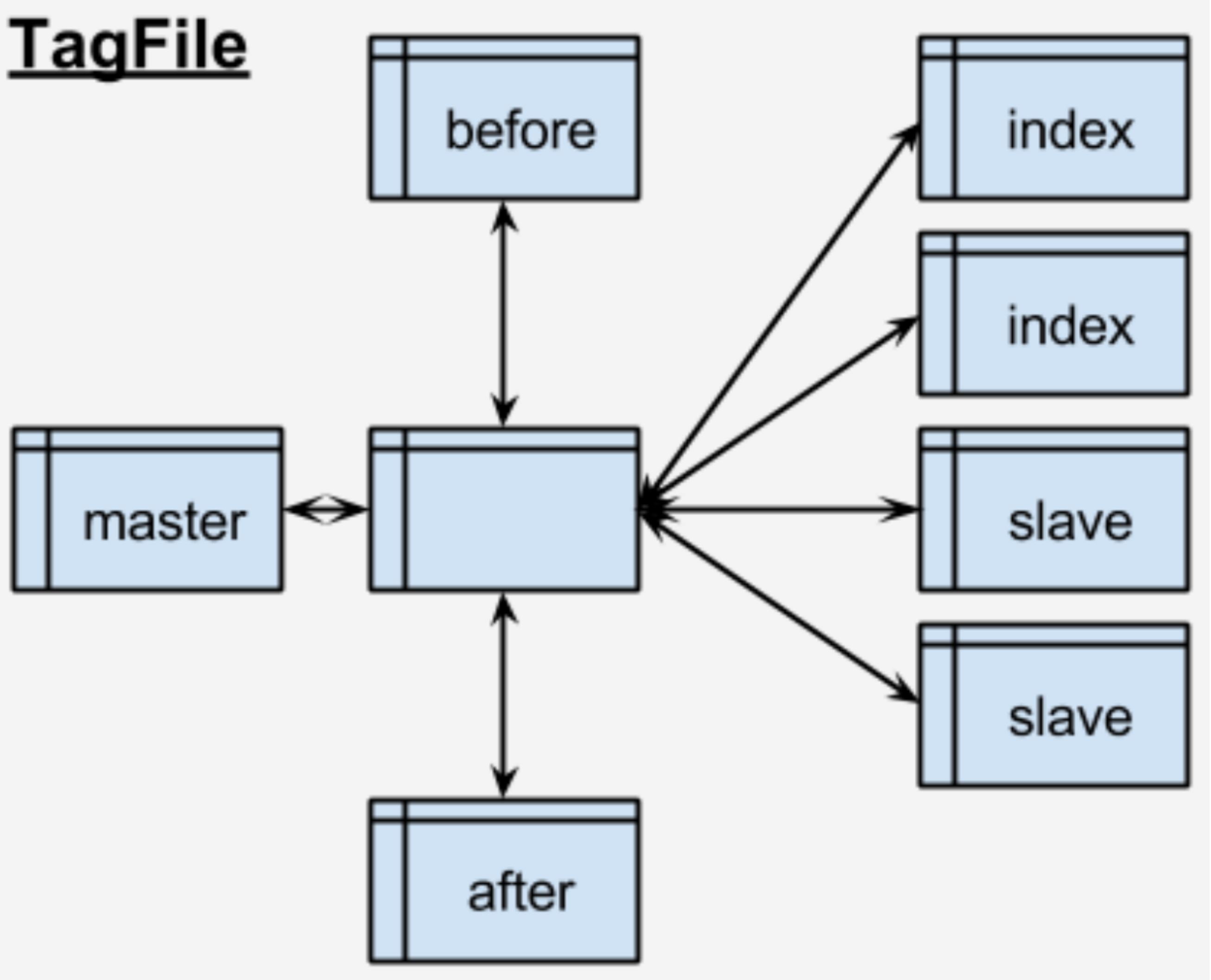
Hadoop core architecture

- Data uploaded from Consumers are stored in mapfiles in HDFS (Hadoop File System). Catalog is stored in Hadoop HBase (metadata about HDFS files)
  - event attributes stored in several groups (constants, variables, references,…)
  - data can be indexed. Index files just have key + references to data files. Some index files created at upload, others added later. Results of queries can be cached to be used



Hadoop core architecture

- Files in HDFS are logically grouped in collections of filesets, representing event collections
- TagFiles can be files or directory of files
- A collection of TagFiles make a TagSet which contains:
  - master record
  - pointers to before/after fillets (vertical partition)
  - slave TagFiles (horizontal partitions)
  - index TagFiles
- Each TagFile is represented by one entry in the HBase table
- The catalog checks the presence and consistency of all components



Query services

- Each search task creates a new TagFile (in the cache)
  - either as a full-content file or as an index
  - this TagFile is registered in the Catalog and can be reused as a basis for future searches
- Access to the data is achieved by a single and simple interface for:
  - upload: copy file into HDFS
  - read: get and search
  - update: add new (vertical) data
  - index: create new indices

Query services

- Search performance enhanced using keyed indexes based on use cases
  - searches based on a key give immediate results (seconds)
  - complex searches use MapReduce and require 1-2 minutes for typical event collections
- Search results can also represent several operations
  - subset selection
  - merge
  - content modification: any data field can be changed/added/removed
  - re-arrangement: in most cases a creation of new index to allow fast key-based search
- Search services via CLI and Web Service GUI

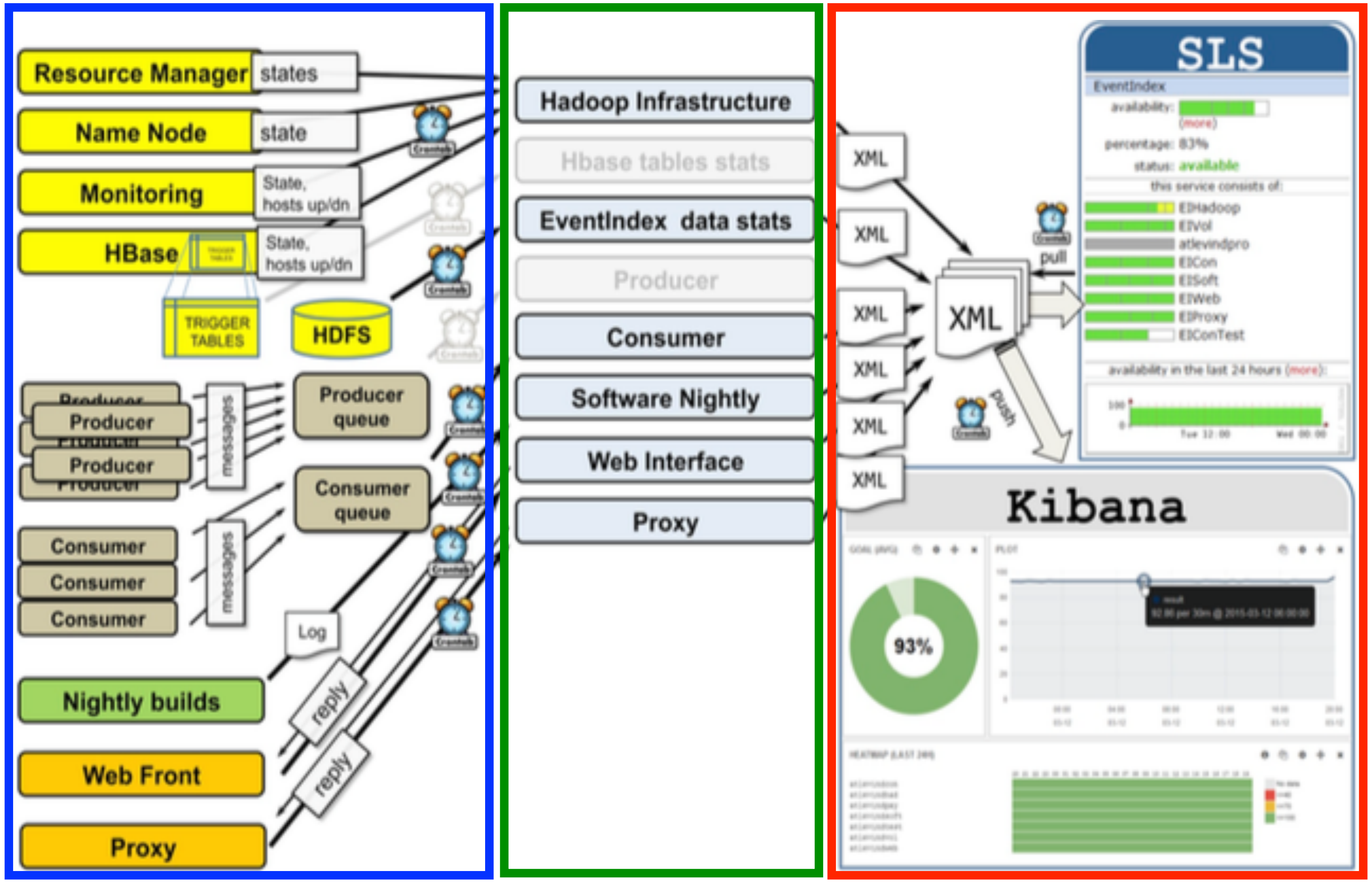
typical performance for count/retrieve operations on Run1 data

Query	Search Base	Retrieved	Time (s)
Get Run/ Event	123492895	1	30
Retrieve all	123492895	123492895	3400
Count all	123492895	0	290
Retrieve with trigger stream & sw version	123492895	939220	142
Count with trigger stream & sw version	123492895	0	130
Retrieve with GUID	123492895	41284	204
Count with GUID	123492895	0	192

- measurements on CERN Hadoop cluster with 18 nodes
- total time depends mainly on the amount of retrieved informations
  - “count” is always much faster than “retrieve” (no output to be written)

System monitoring

- Monitors the health of all servers and processes involved in the chain
  - ActiveMQ brokers and Consumers
  - Hadoop cluster and Web servers
- Contents monitoring under development



servers and processes to be monitored

statistics gatherers

statistics displays

Development status, deployment and operation

- All major components exist and work satisfactorily
  - data collection: Producer transform runs at Tier-0 and on the Grid. Consumer reads data from the ActiveMQ servers, validates them and stores to HDFS
  - Storage system: data organisation in Hadoop and indexing in catalogue; trigger decoding interface
  - Query system: CLI and web interfaces. Also EventLookup for event picking
  - Monitoring: System level monitoring in the new CERN Kibana environment
- Run1 data processed since 1st February. Loaded all first-pass Tier-0 production (5.5 billion events). Latest version of reprocessed data loading almost finished
  - EventIndex data size in Hadoop: ~350 B/event in the EventIndex only for LHC Run1. 2TB of RAW informations (6TB after internal replication in Hadoop)
- Message broker data occupancy kept under control using multiple consumers
- Automatic data reformatting and cataloguing in Hadoop
- Run2 data now flowing automatically and continuously from Tier-0/Grid and available in real time
  - still working on improve system interconnections and monitoring
  - automatic checks of production completeness

Summary and outlook

- The ATLAS experiment needs an EventIndex to catalogue billions of events taken every year
  - TBs are needed to index PBs of event data
- The EventIndex infrastructure is now in operation
  - designed, developed and deployed in 2012-2015
  - Run1 Tier-0 processing data fully indexed, reprocessed data indexing almost finished
  - Run2 new data indexing in real time
- All initial use cases are satisfied with good performance
- Still much work ongoing and to do
  - data validation still semi-automatic
  - robustness w.r.t. network/hardware problems
  - additional internal monitoring
  - performance improvements for common queries