The ATLAS EventIndex: An Event Catalogue for Experiments Collecting Large Amounts of Data

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

28th Mar 2014 – ISGC 2014, Taipei

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

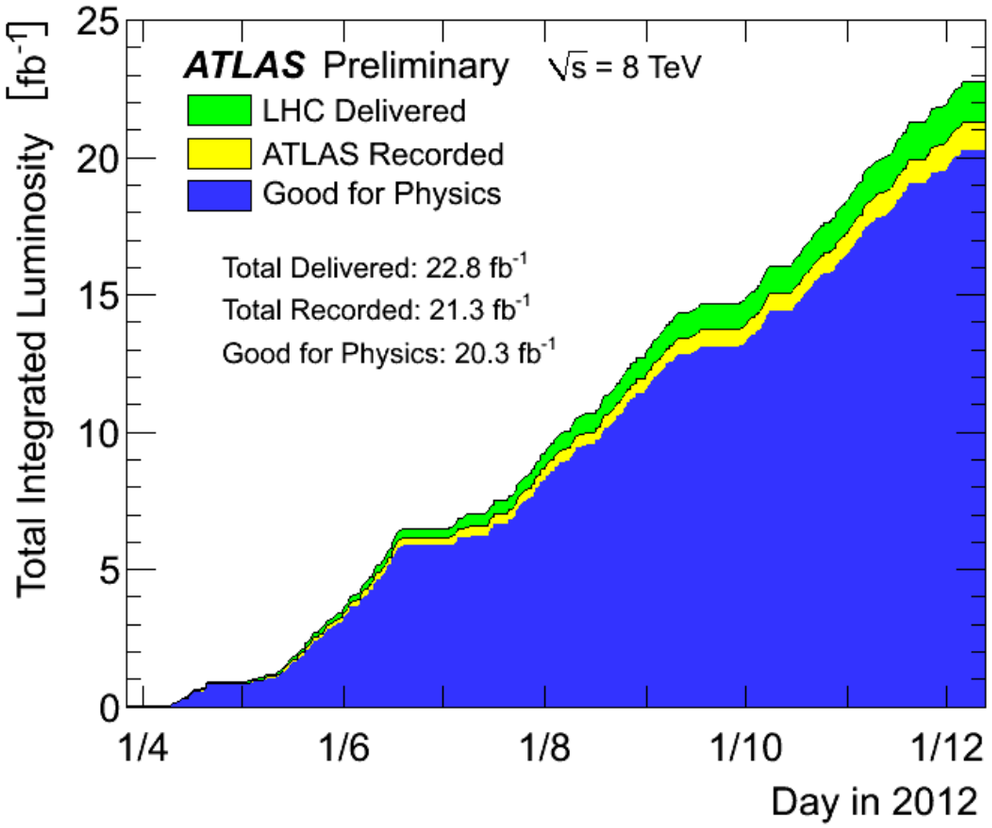
July 2010

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

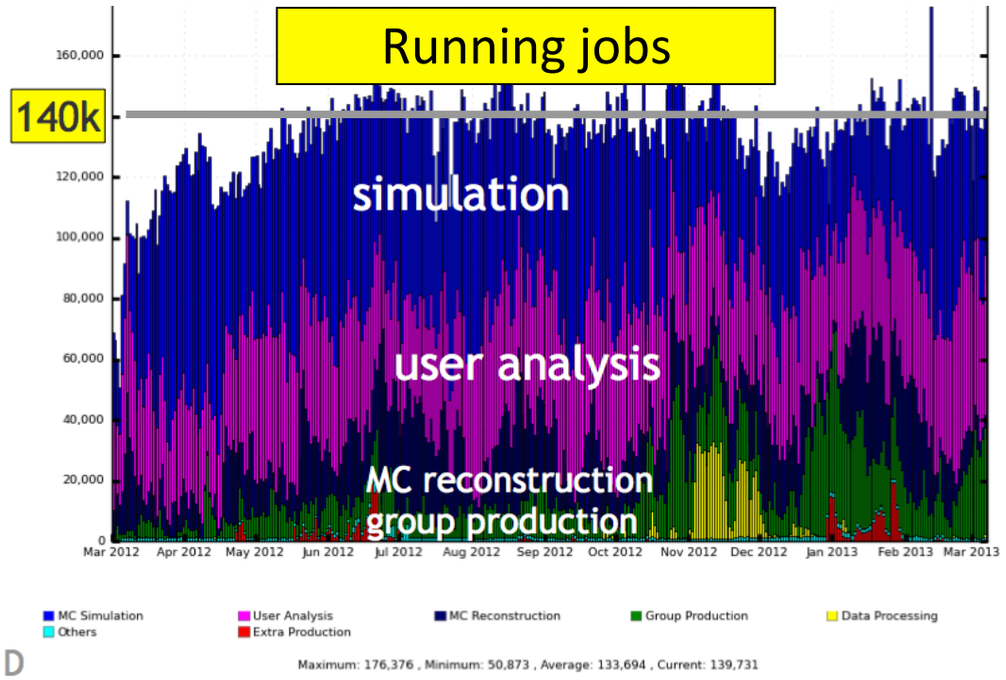
38 countries, 177 Institutions
~2900 scientific authors
~1800 with PhD

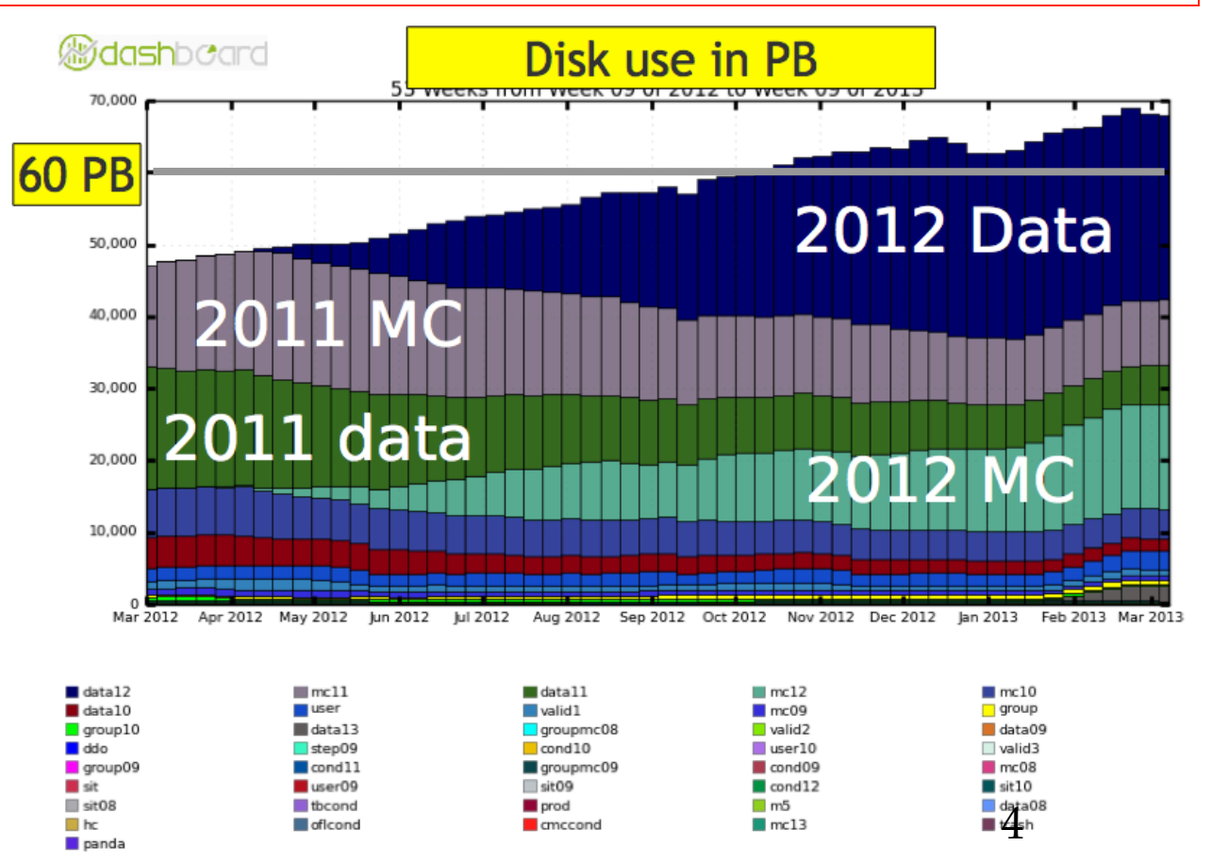
July 2010

ATLAS Distributed Computing



- The performance of the Worldwide LHC Computing Grid sites has been outstanding, and is fundamental to ATLAS physics analysis
- We have benefited from CPU resources beyond pledges which have enhanced the speed and breadth of our physics program
- We have heavily and continuously optimised our use of the resources during the last year - "everything is full"





Introduction

- Modern scientific experiments collect very large amount of data
  - ATLAS: e.g. 2 billion real and 4 billion simulated events in 2011 and 2012
- A catalog of data is needed according to different point of view
  - 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 the late 1990's
    - 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
    - main use: events skimming

EventIndex

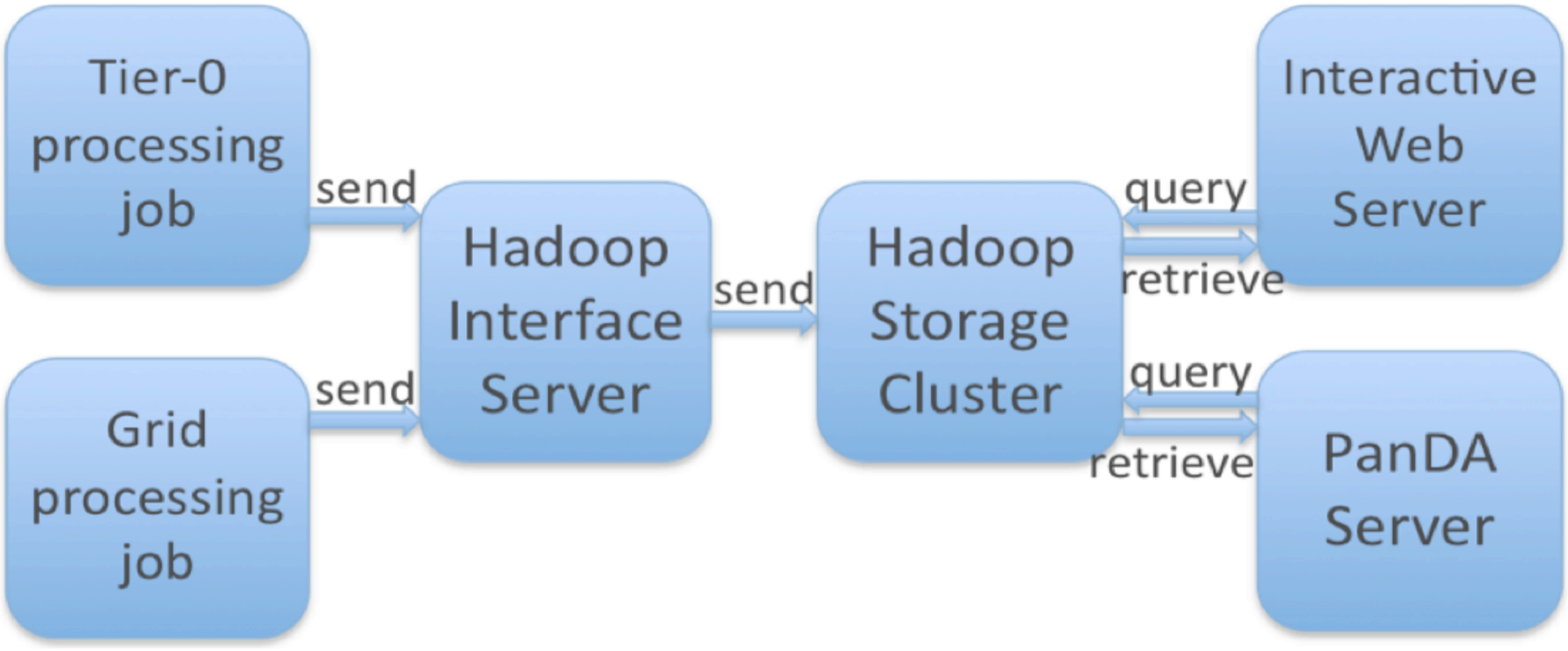
- **GOAL**: design a more agile system ("NoSQL" databases) that keeps the functionality of skimming by keeping only the pointers (currently GUIDs) to the files where the event in question can be found at every stage of processing and eliminating other variables of lower importance
- ***EventIndex***:
  - a *complete* catalog of ATLAS events
    - all events, real and simulated data
    - all processing stages
  - contents
    - event identifiers
    - online trigger pattern and hit counts
    - references (pointers) to the events at each processing stage (RAW, ESD, AOD, NTUP) in **all** permanent files on storage

Use cases

- old use cases
  - event picking
    - give me the reference (pointer) to “this” event in “that” format for a given processing cycle
  - event skimming
    - give me the list of events passing “this” selection and their references
  - production consistency checks
    - technical checks that processing cycles are complete
- new use case
  - event service
    - give me the references for “this” range of events

Data Flow

- Information for the EventIndex is collected from all production jobs running at CERN and on the Grid and transfered 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



EventIndex project timescales

- Run2 should start at the beginning of 2015
  - *2013*: tests of data formats, schemas, performance of upload, search and retrieve data on a reduced dataset (~1 TB)
  - *2013*: implementation of the chosen solution on the CERN Hadoop cluster; adaptation or development of external services
  - *first half 2014*: commissioning of the new system; upload of all existing data; performance optimisation
  - *second half 2014*: discontinuation of Oracle TagDB; commissioning with new cosmic-ray data
- this is a success-oriented schedule
  - it assumes that the Hadoop/HBase back-end is better than Oracle and scales to 100 TB of payload information by 2017 (end of Run2)
  - also that there will be enough manpower and hardware resources for the new system

EventIndex project breakdown

- Definition of 4 major work areas (or tasks)

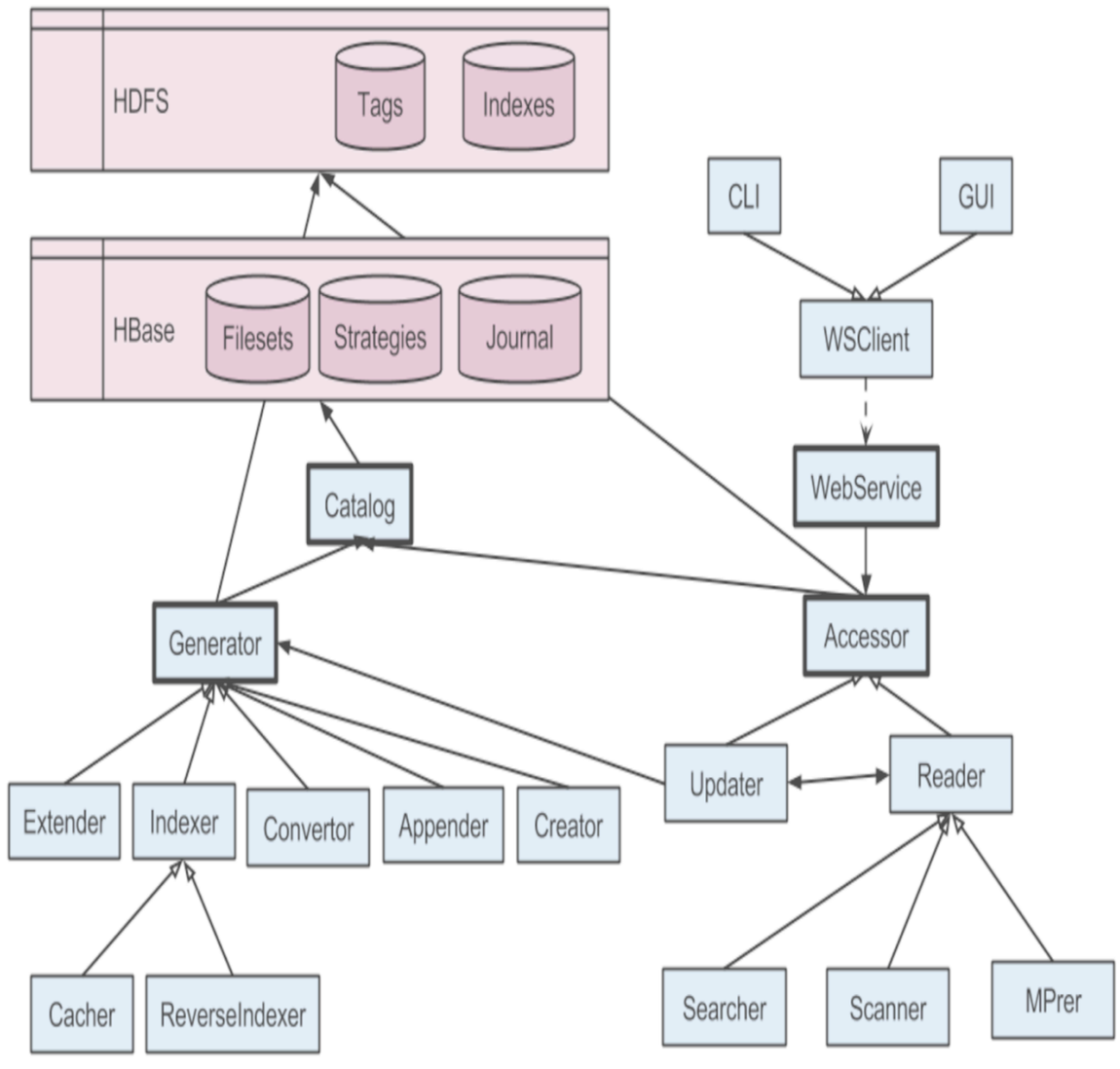
1. **core architecture**
   - design and prototyping, then development and deployment of the core infrastructure to implement the EventIndex using Hadoop technology; timescale: June 2014
2. **data collection and storage**
   - design and prototyping, then development, deployment and operation of the infrastructure to collect from Grid and non-Grid jobs the information needed by the EventIndex, and transfer and upload it to the EventIndex DB; timescale: autumn 2014
3. **query services**
   - adaptation of the web services and APIs now used to query the Tag DB to the EventIndex using Hadoop technology; timescale: June 2014
4. **functional testing and operation**
   - end-to-end tests of the completeness, functionality and performance of the EventIndex and related services. Checks of the operation procedures and setting up monitoring tools; timescale: end 2014

Core architecture (1)

- Design and prototyping, then development and deployment of the core infrastructure to implement the EventIndex using Hadoop technology
- Got access to a test Hadoop cluster managed by CERN-IT-DSS group
- Uploaded ~1 TB of data from TAGs (full 2011 Tier-0 processing) for initial test in different formats
  - Plain CSV files
  - HBase (Hadoop's DB format)
  - Different binary format
- After testing decided to store data in HDFS in Map format (index in memory, data on disk) and data catalog in HBase tables – currently implementing

Core architecture (2)

- data are stored in Hadoop map files
  - event attributes stored in several groups (constants, variables, references,…)
  - data can be indexed (e.g. using inverted indices for trigger info). 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



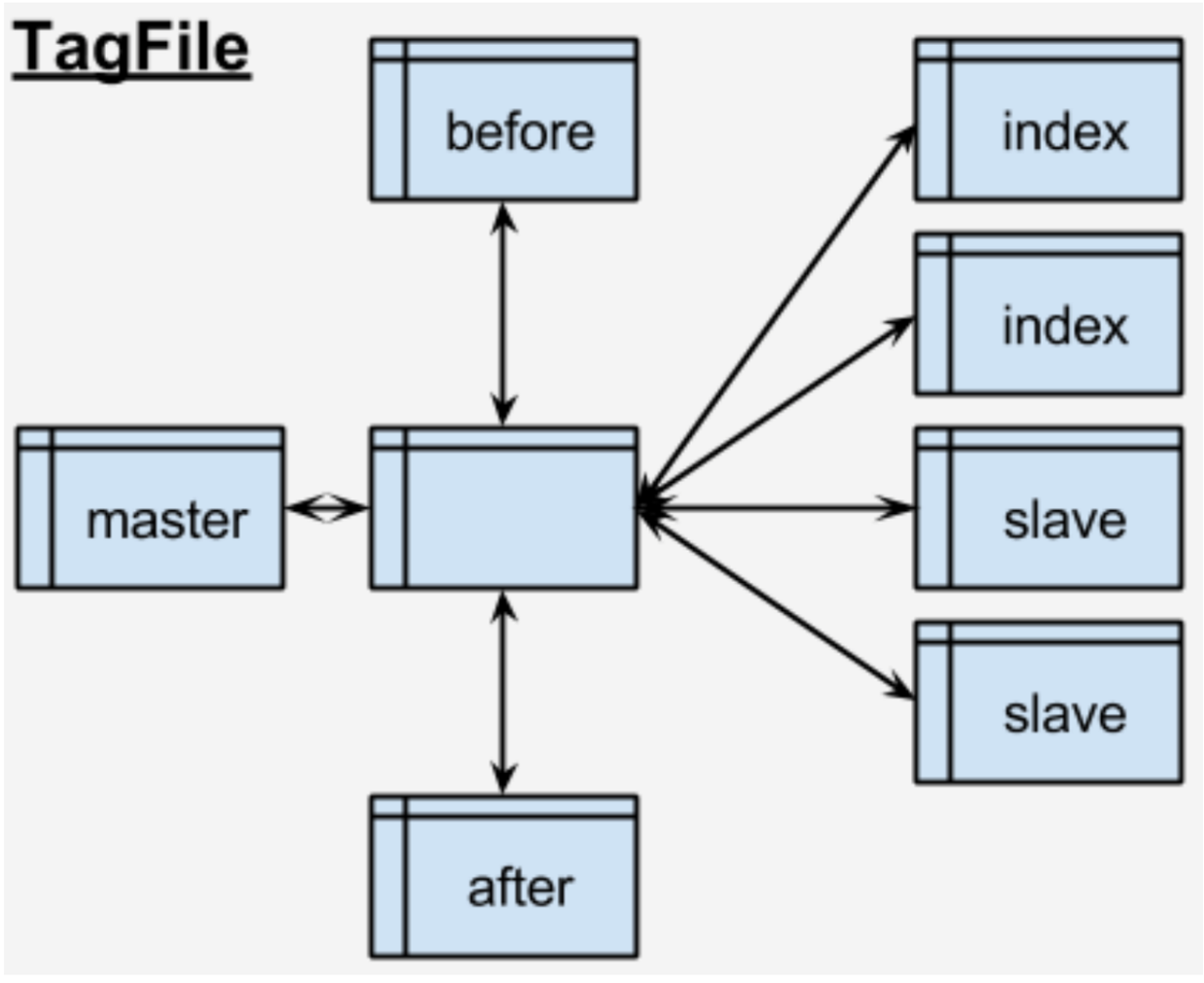
Core architecture (3)

- searches can be performed in two steps:
  1. get reference from index file
  2. using that ref., get data
- Searches can be performed using keys (immediate results), or on data files
- full searches can be done with full scans or using MapReduce jobs
- access to the data is achieved by a single and simple interface for:
  - upload: copy file into HDFS
  - read: get & search
  - update: add new (vertical) data
  - index: create new indices

- Some use cases are:
  1. COUNT/RETRIEVE events W/O FILTER (which simply means to count all the events in the dataset without any particular request)
  2. COUNT/RETRIEVE events W/ trigger AND/ OR W/ run number
  3. COUNT/RETRIEVE events FOR EACH run number W/ OR W/O trigger and vice versa
  4. RETRIEVE GUID W/ run number AND W/ trigger

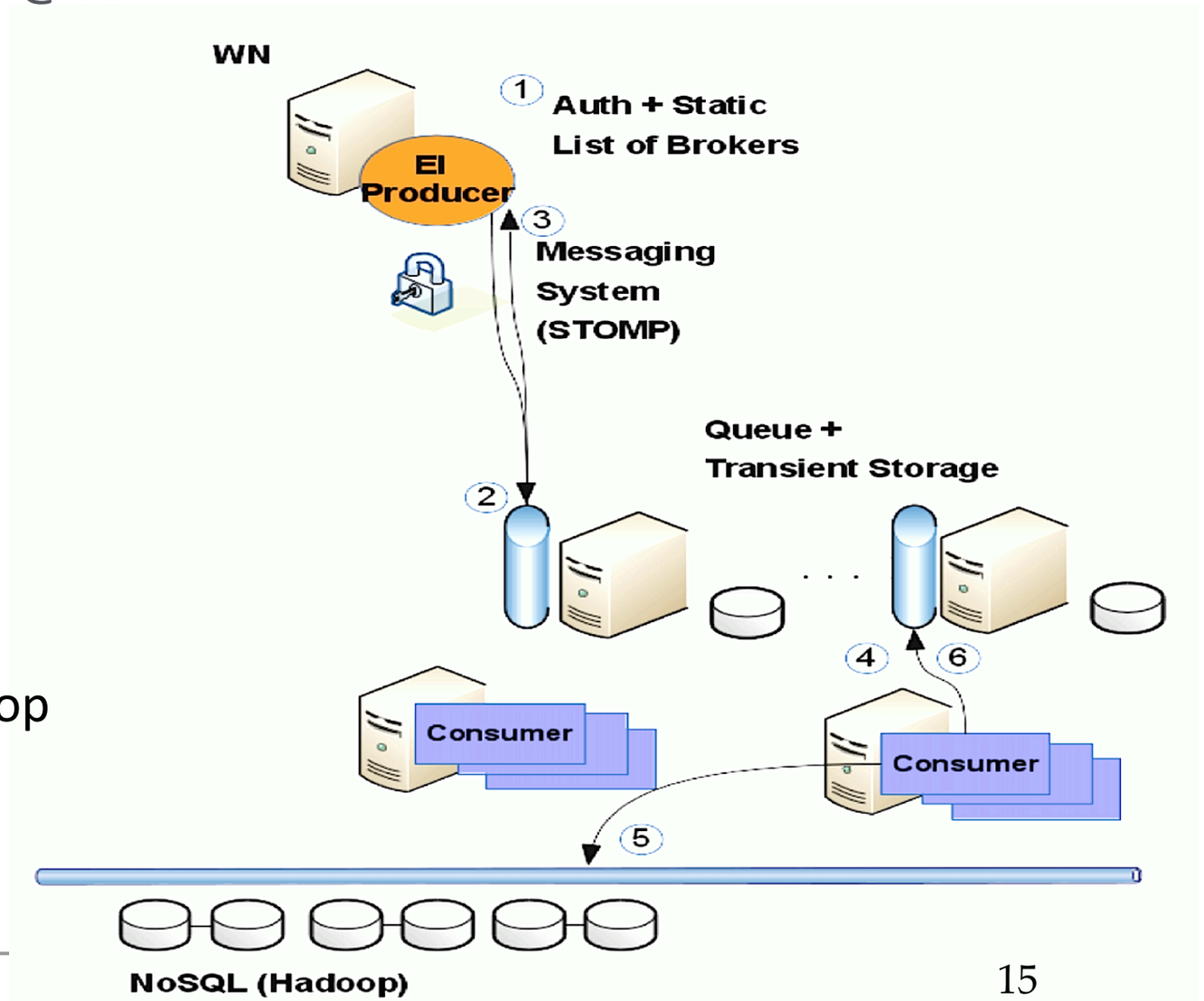
Core architecture (4)

- The files in the Hadoop file system (HDFS) are stored in collections of “filesets”.
  - each collection represents an event collection (grouping all events that have been processed by the same software version)
- TagFiles can be files or directories of files
- A collection of TagFiles makes a TagSet
- Each TagSet has:
  - a master record
  - pointers to before/after filesets (vertical partitions)
  - slave TagFiles (horizontal partitions)
  - index TagFiles
- The catalogue checks the presence and consistency of all components
- Each TagFile is represented by one entry in the HBase table.



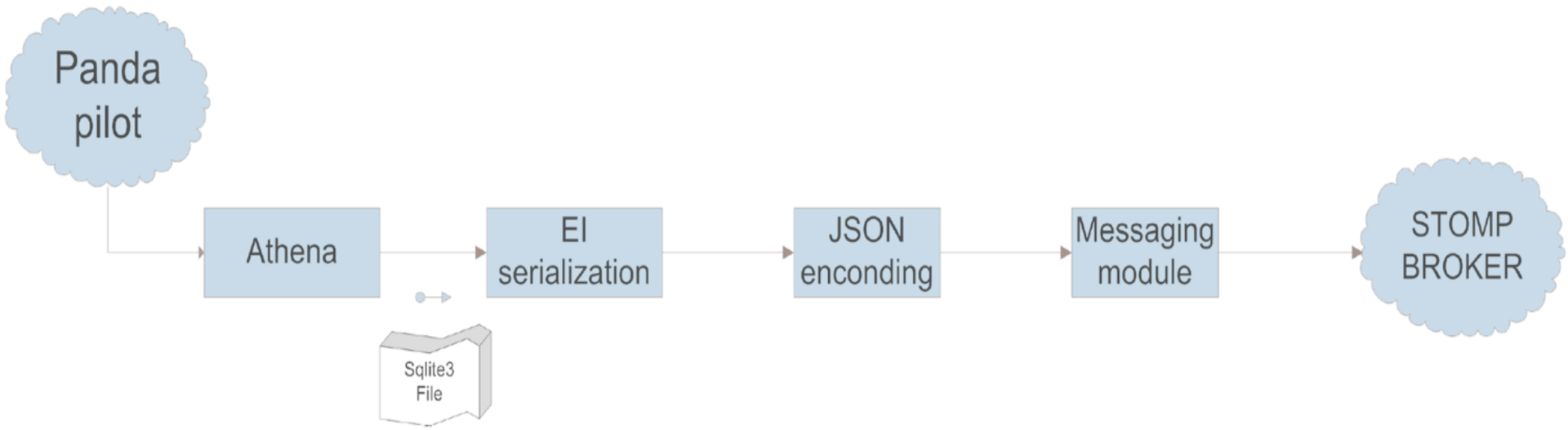
Data collection and storage (1)

- Data to be stored in the EventIndex are produced by all production jobs that run at CERN or on the Grid.
  - snippet of information of permanent output files (file unique identifier, event relevant attributes) has to be sent to central server @CERN
- accept over **80 Hz** of file records and insert into the back-end over **30 kHz** of event records (plus contingency).
- **producers** run within the "pilot" process on each worker node and transmit the data using a messaging system to a queue in one of the endpoints
- asynchronous **consumers** continuously get new data, check their validity with the production system and pass them into Hadoop
- **final step** is to inform the production system of the successful insertion, and remove the data from the queue



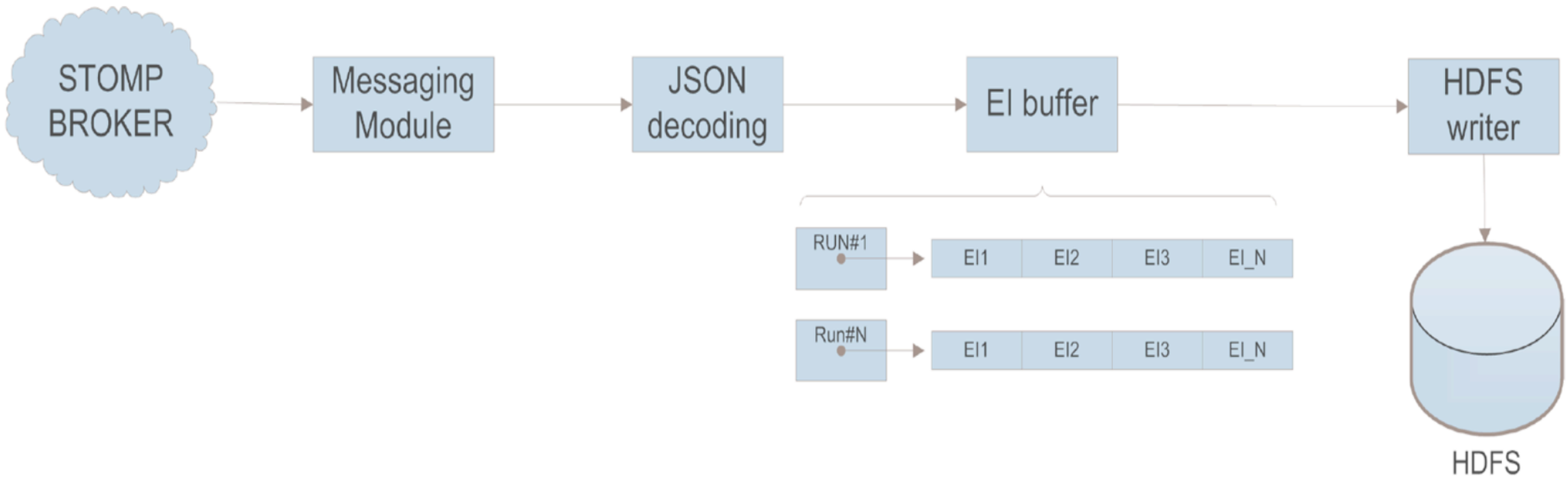
Data collection and storage (2)

- The data **producer** process is split in two parts:
  1. read the data file to create an intermediate EI file - used also to decouple the processing of the input file from sending the messages - extracting the relevant infos to be transmitted (event identification, trigger masks, references to the files containing the event). Currently a Python script that create SQLite3 EI file of Python serialized objects, containing key-value pairs in one table “shelf”
  2. Read information from EI file, build the messages (about 10kB each) and send them to the broker. Currently a Phyton script



Data collection and storage (3)

- The **consumers**:
  - receive the messages,
  - decode the information in JSON format,
  - order the events by run number in memory queues,
  - build CSV records and append them to files in Hadoop
  - the CSV file is then transformed into map file and included in the system Catalog to become searchable



Data collection and storage (4)

- **Messages** are sent in a compact format (JSON standard encoding) ActiveMQ chosen as the messaging service, and STOMP as the message protocol, as they are supported at CERN
- SSL endpoint at the broker can be used
- Replication of the message queues in other sites (duplication and high availability)
- Tests show that it is possible with a single broker to achieve the required data transfer rates, with peaks in the load of **200 kHz** of event records, and about **60 kB/s**
- If an adequate number of consumer processes is active, the payload is retrieved immediately and no backlog is created

Query services (1)

- adaptations and/or evolutions of the web services and APIs used to query the Tag DB to the EventIndex using Hadoop technology
- Simple data scan operations on the test dataset (1 TB of 2011 data) using the available Hadoop cluster of 20 nodes take about **15 minutes**;
  - We consider this as the worst case, as normally no query would need to read in all data from disk for a full year of data taking
  - There are many possibilities of optimisation, still to be studied and exploited
- Queries that give the run number and event number (event picking use case) and use the so-far unoptimised “accessor” method return their result in **3.7 seconds**
  - for comparison, the same query using a full Map-Reduce job returns in **90 seconds** (again, to be considered as worst cases - first prototype implementation)

Query services (2)

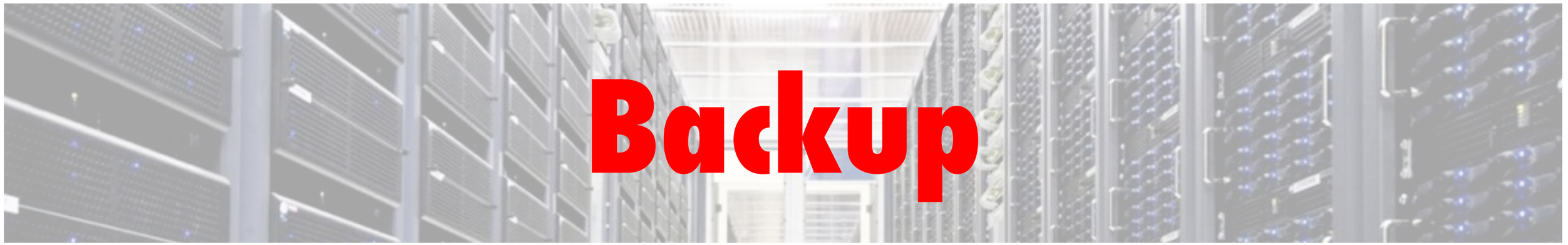
- **Important use case**: query/count the events that satisfied a given trigger condition, or a combination
  - Trigger chains need to be first decoded by knowing the trigger menu via the run number and finding in the COMA database the corresponding trigger info for each bit in the trigger words
  - The COMA database is stored in Oracle and accessed using ODBC drivers; the retrieved information is then used by MapReduce jobs in the Hadoop cluster
  - We also consider replication and caching of the relevant data to Hadoop in order to improve performance by using “internal” Hadoop calls instead of polling an external database.

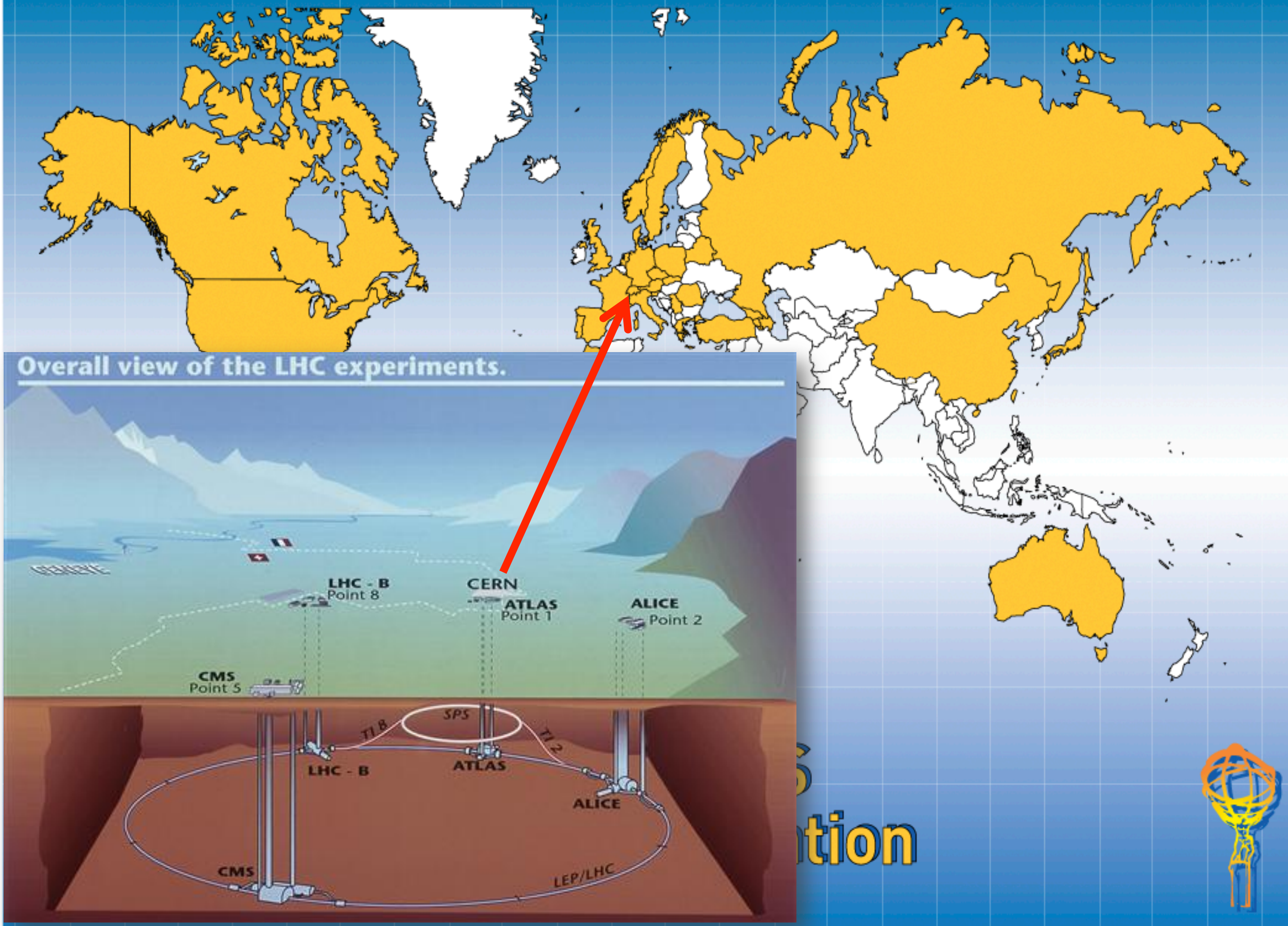
Conclusions and outlook

The EventIndex project is on track to provide ATLAS with a global event catalogue in advance of the resumption of LHC operations in 2015.

The global architecture is defined and prototyping work is in progress; preliminary results are encouraging.

Current work: full implementation of the final prototype, loaded with Run1 data, and the completion of deployment and operation infrastructure.

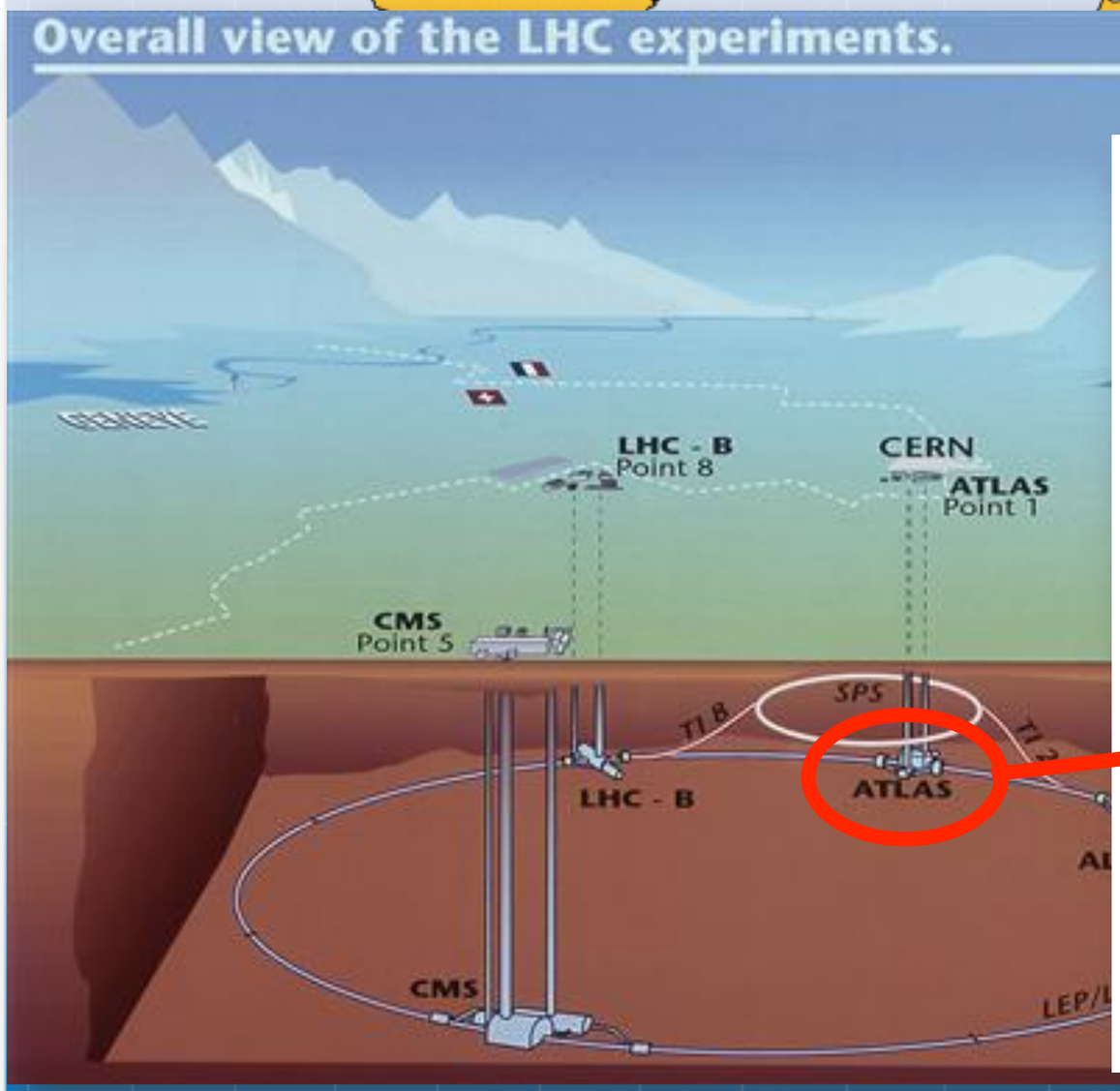

Backup



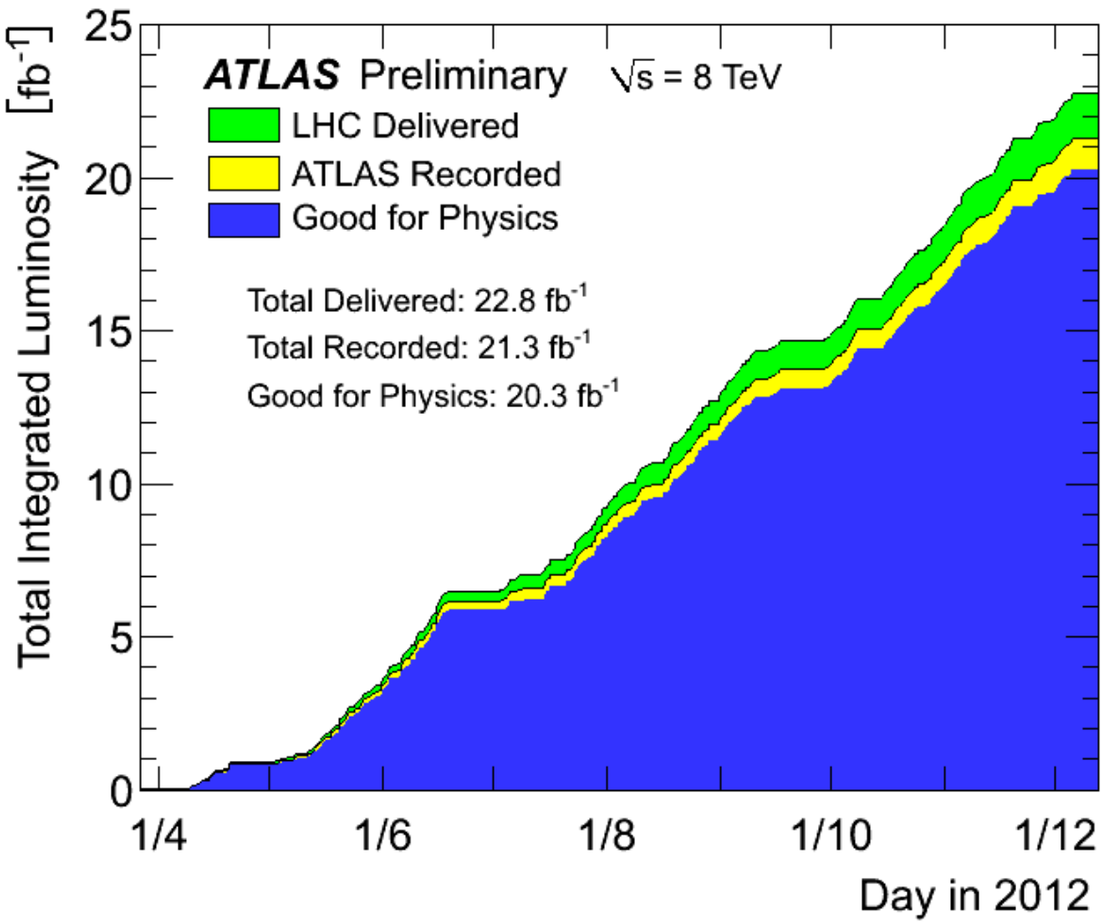
tion

July 2010

Overall view of the LHC experiments.



End of "Run-1" pp data



ATLAS p-p run: April-December 2012										
Inner Tracker			Calorimeters		Muon Spectrometer				Magnets	
Pixel	SCT	TRT	LAr	Tile	MDT	RPC	CSC	TGC	Solenoid	Toroid
99.9	99.1	99.8	99.1	99.6	99.6	99.8	100.	99.6	99.8	99.5
All good for physics: 95.5%										

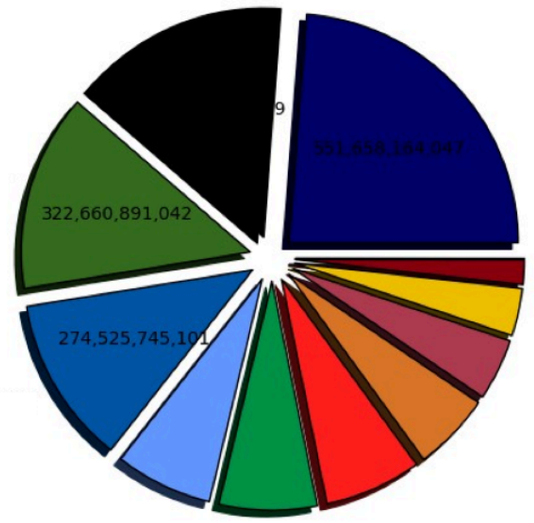
Luminosity weighted relative detector uptime and good quality data delivery during 2012 stable beams in pp collisions at √s=8 TeV between April 4th and December 6th (in %) – corresponding to 21.3 fb^{-1} of recorded data.

2012 8 TeV pp data: around 90% of data delivered is used for analysis (all analyses use same status cuts)

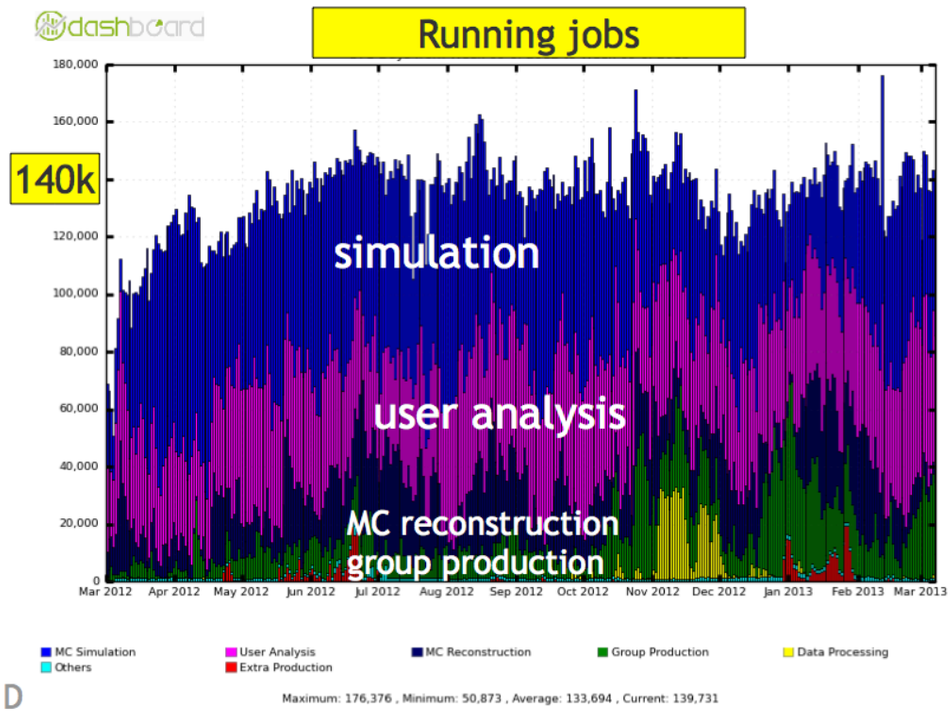
Fantastic LHC performance over the last 3 years

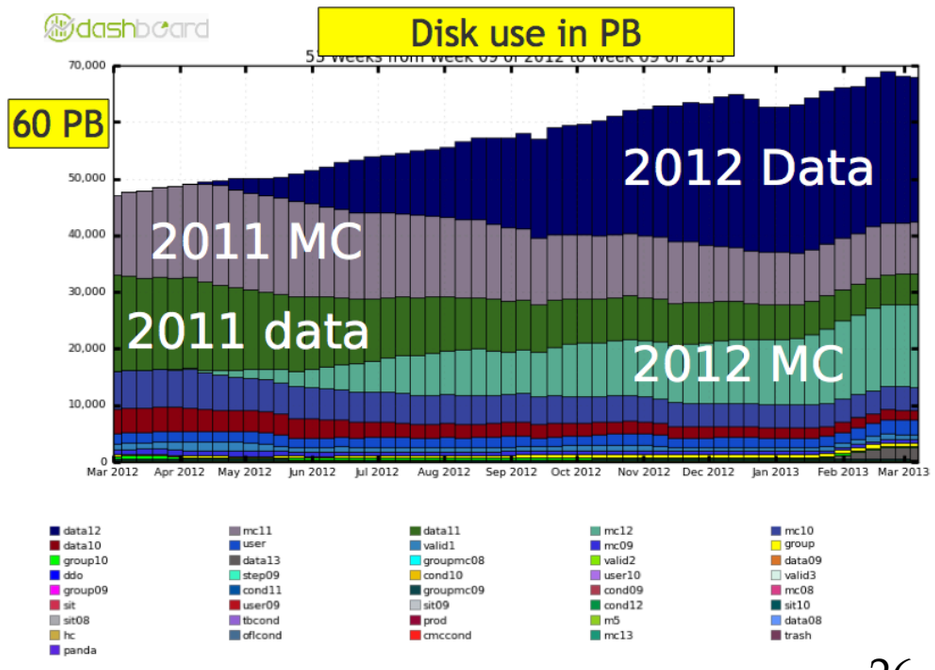
The importance of WLCG (Worldwide LHC Computing Grid)

- The performance of the WLCG Grid sites has been outstanding, and is fundamental to ATLAS physics analysis
- We have benefited from CPU resources beyond pledges which have enhanced the speed and breadth of our physics program
- We have heavily and continuously optimised our use of the resources during the last year - “everything is full”



CPU consumption by cloud
Jan 2012-present
All clouds contribute substantially





H/W of Hadoop cluster @CERN

	IT-DSS	Recommended Data node	Recommended Master node
Total number of nodes	15 (+1 powerful master node)	10	1
Disc storage	Attached 20x2TB total	Attached 80x1TB total 8x1TB per node	RAID 1.0 2x1TB
Total max. throughput	37'500 MB/s =15*20 *125MB/s	4'000 MB/s = 40x125 MB/s	125MB/s
Network per node	1x1Gb (shared)	2x1Gb	2x1Gb
RAM per node	12GB	24GB	24GB (48GB)
CPU	4 cores	2x4 cores	2x4 cores