Distributed Data Collection for the ATLAS EventIndex

A. Fernández Casani, S. González de la Hoz, J. Sánchez

On behalf of the ATLAS Collaboration

Instituto de Física Corpuscular (IFIC)
CSIC- Universitat de València

CHEP 2015 Okinawa – 13-17 April 2015

Outline

- Event Index Data Collection
- Architecture
- Producer
- Messaging
- Consumer
- Validation
- Performance
- Conclusions

What is the ATLAS Event Index

- Purpose:
  - Build a complete catalog of all ATLAS physics events (real and montecarlo) for all processing stages.
- To allow:
  - Production consistency checks
  - Event picking
    - Give the reference to a specific event
  - Event service
    - Provide references to production processes
  - Trigger checks and event skimming
    - Count or give an event list based on trigger selection.

See detailed info in the talk by D. Barberis in this session (abstract #208)

You are invited also to look at posters by F. Prokoshin (#220) and J. Hrivnac (#221)

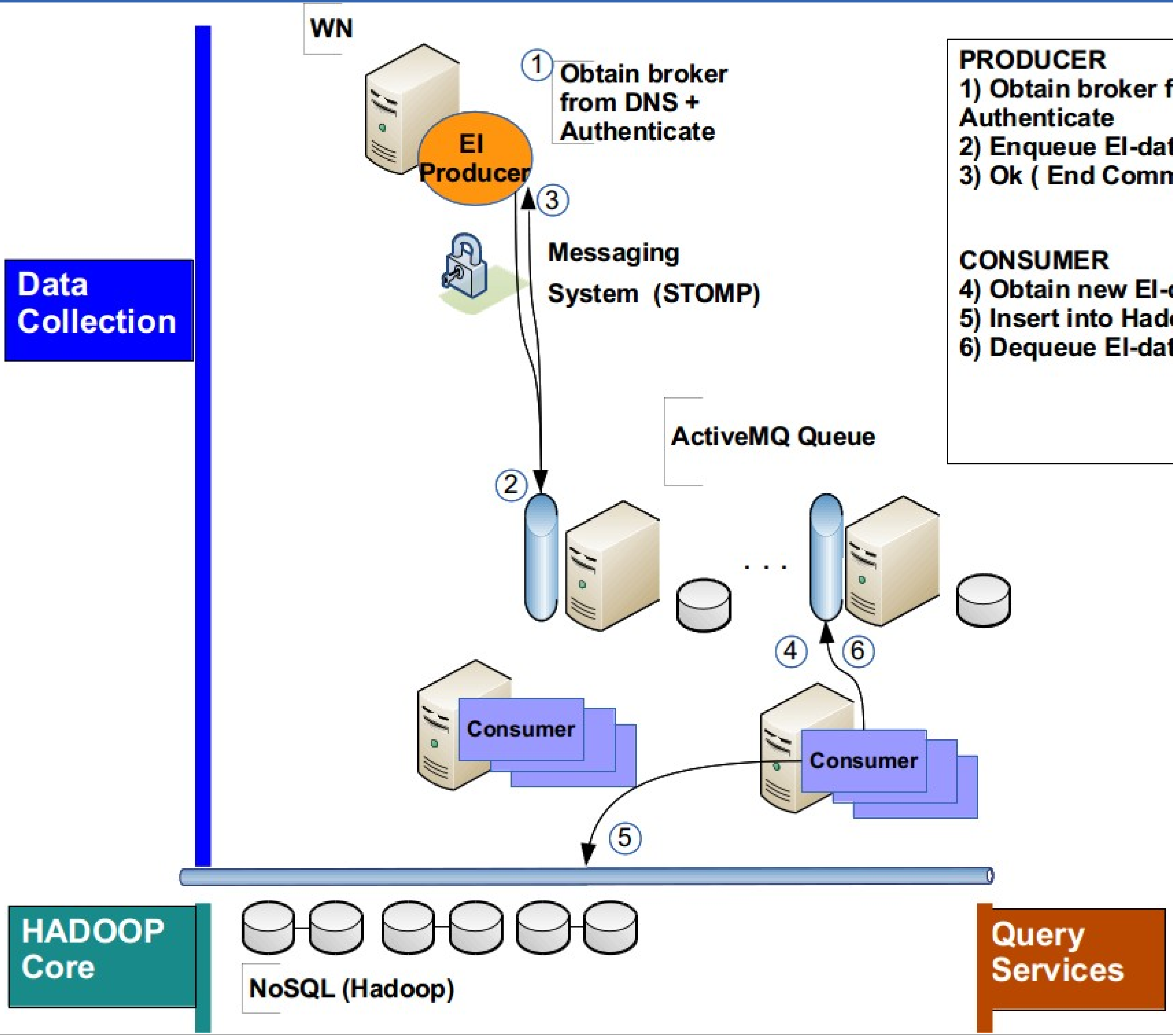
Event Index Data Collection

- Goal:
  - Get relevant information for each event, process, store into HADOOP at CERN and make it available to users and tools quickly.
- Requirements:
  - Process should be fast and quasi synchronous with data production that runs at Tier0 farm at CERN and around the world using the GRID. Data production is inherently distributed.
  - Index information per event has to be small.
  - Transport mechanism should be independent of common ATLAS data flow (DDM) to save steps and avoid polluting it with too many small and transient files.
  - Ensure availability and reliability so no information is lost
    - Although it can always be extracted again later if needed

Architecture

- Design adoptions:
  - Use a producer/consumer architecture in which producers run at distributed centers and consumers run centrally at CERN
  - Producers extract event data in the same job that produce them.
    - Run the Event Index producer as a substep in the same job
    - Send information to central servers at that moment
  - Use a messaging protocol to transport Event Index information between producers and consumers
    - Streaming Text Oriented Messaging Protocol (STOMP)
    - Apache ActiveMQ servers (Red Hat JBoss A-MQ) provided by CERN-IT
  - Encode Event Index data into JSON to build the text message
  - Consumers get Event Index data provided by producers, process, validate and store them into HADOOP

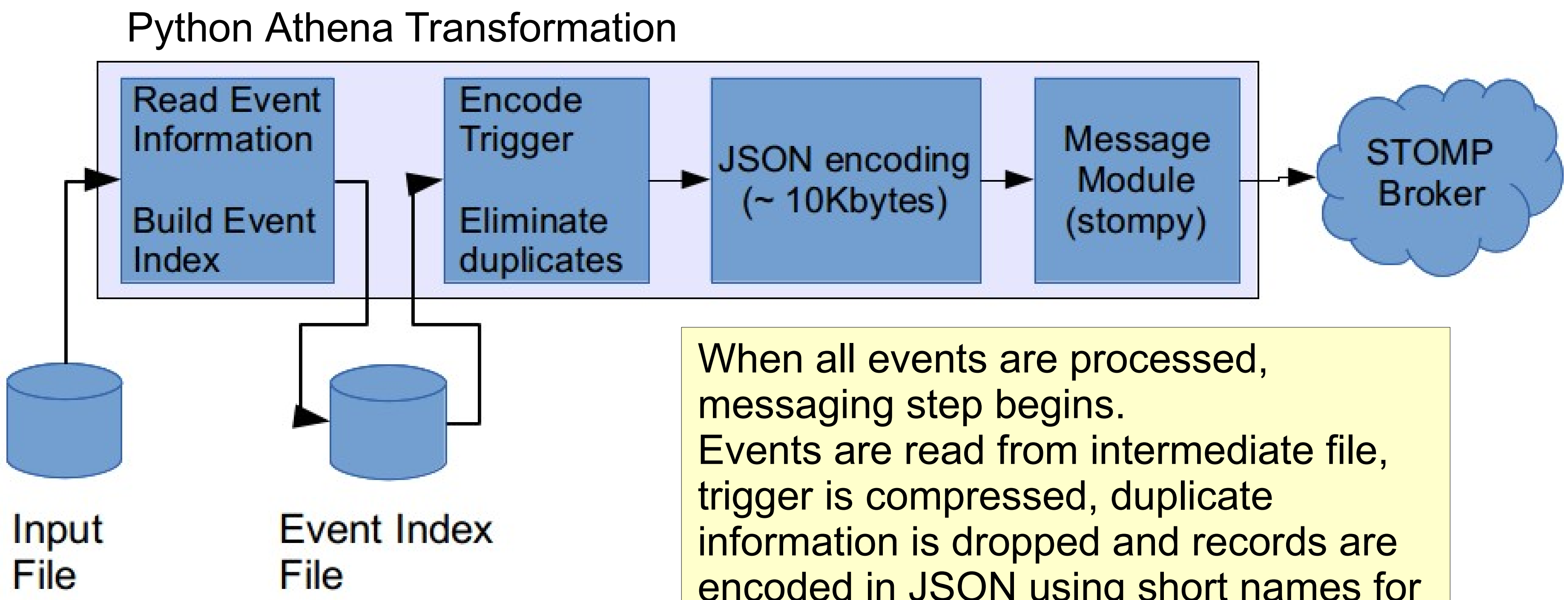
Data Flow Overview



PRODUCER
1) Obtain broker from DNS + Authenticate
2) Enqueue EI-data (Messaging)
3) Ok (End Communication)

CONSUMER
4) Obtain new EI-data
5) Insert into Hadoop
6) Dequeue EI-data

Producer Architecture



File GUID is sent across all stages

Information is serialized event by event (cpickle) and stored in sqlite format

When all events are processed, messaging step begins.
Events are read from intermediate file, trigger is compressed, duplicate information is dropped and records are encoded in JSON using short names for keys (eg, “r” for RunNumber, “e” for EventNumber”)

Data extracted:
- Event Identification (run, event ..)
- Navigational Information
- Trigger Information

can be expanded to include some physics quantities.

Intermediate EI file can be registered into ATLAS dataflow or deleted silently

Message Flow

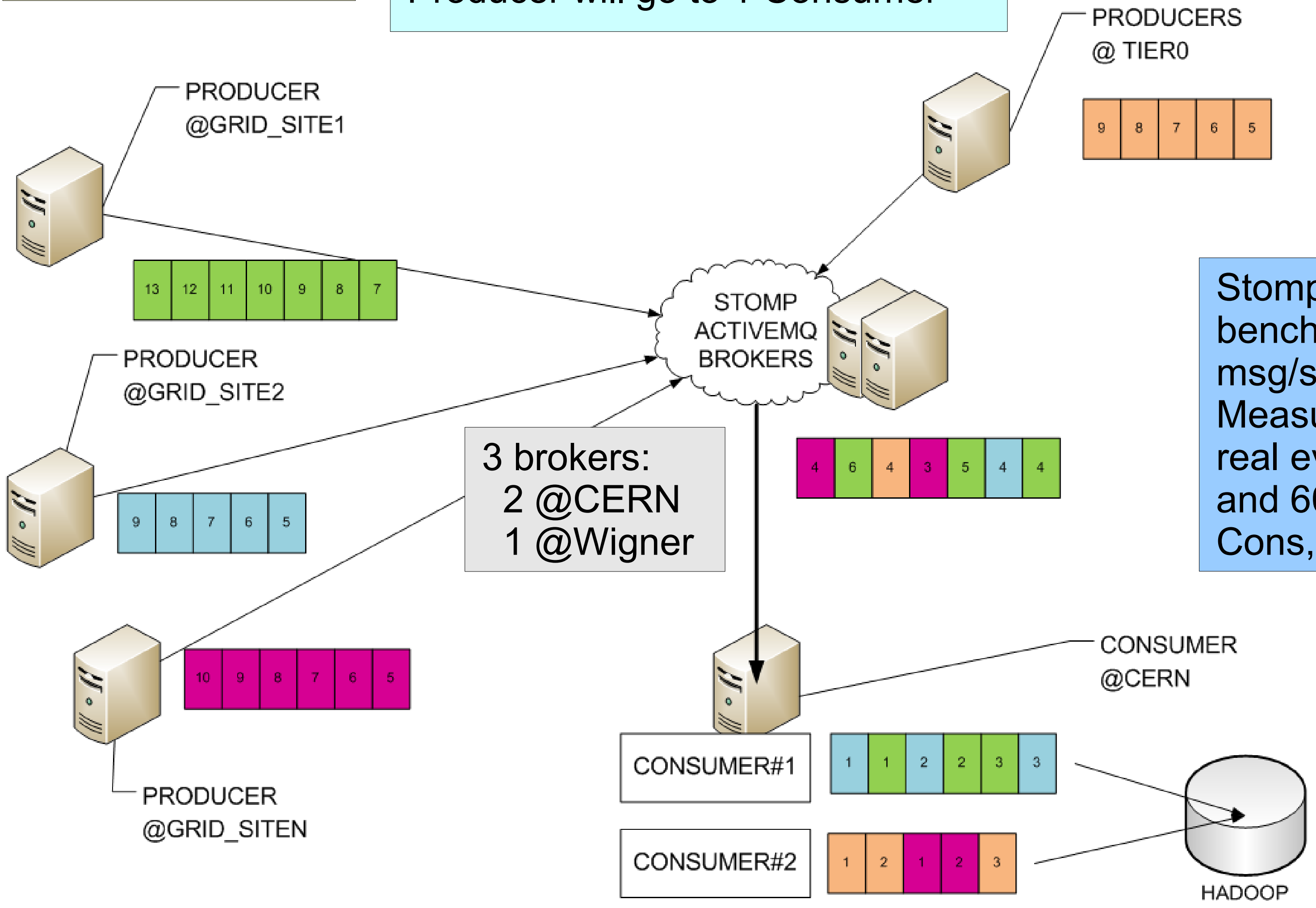
Message size is set small 1-10kB to keep broker queues agile.

Producers tag messages by group (JMSXGroupID)

Ensures that all messages from 1 Producer will go to 1 Consumer

Atomic transactions on Producers: if connections breaks no partial processing occurs.

Status messages are sent from producers and consumers to an alternate queue.

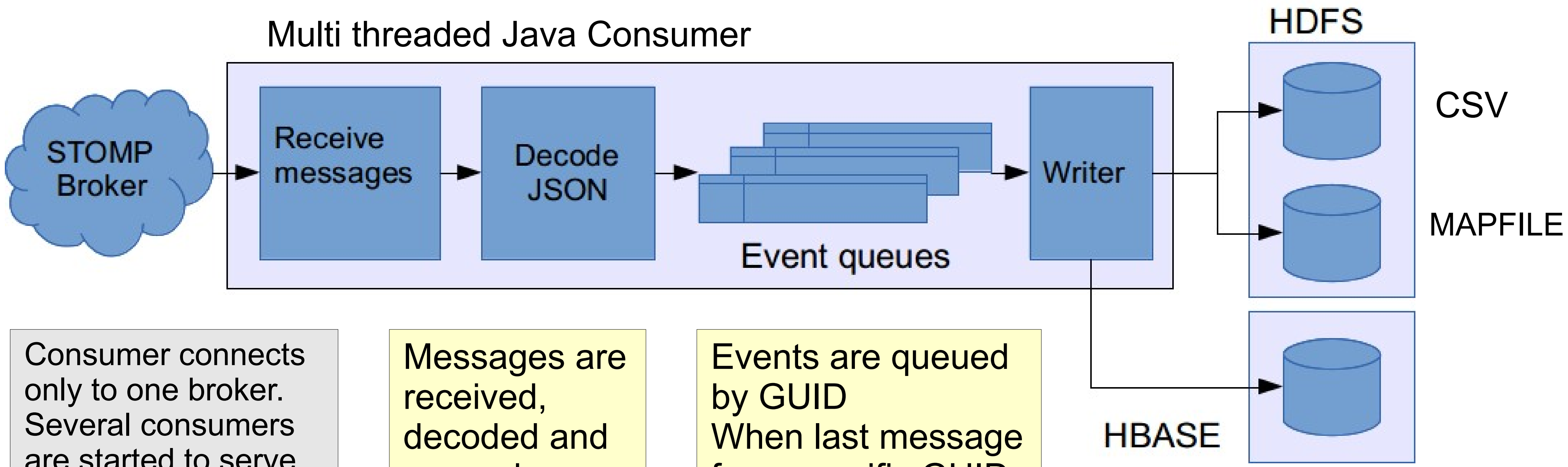


3 brokers:
2 @CERN
1 @Wigner

Stomp performance in our benchmarks reaches over 350 msg/s and 10Mb/s per Producer. Measured performance for sending real events reached 200K event/s and 60Mb/s (1Broker, 6 Prod/s, 4 Cons, 50K events/job)

Files stored on Mapfile format usable by Hadoop Core Services

Consumer Architecture



Consumer connects only to one broker. Several consumers are started to serve all brokers

Messages are received, decoded and queued. When a new GUID arrives, a new queue is created.

Events are queued by GUID
When last message for a specific GUID is received, queue is closed and released to writer

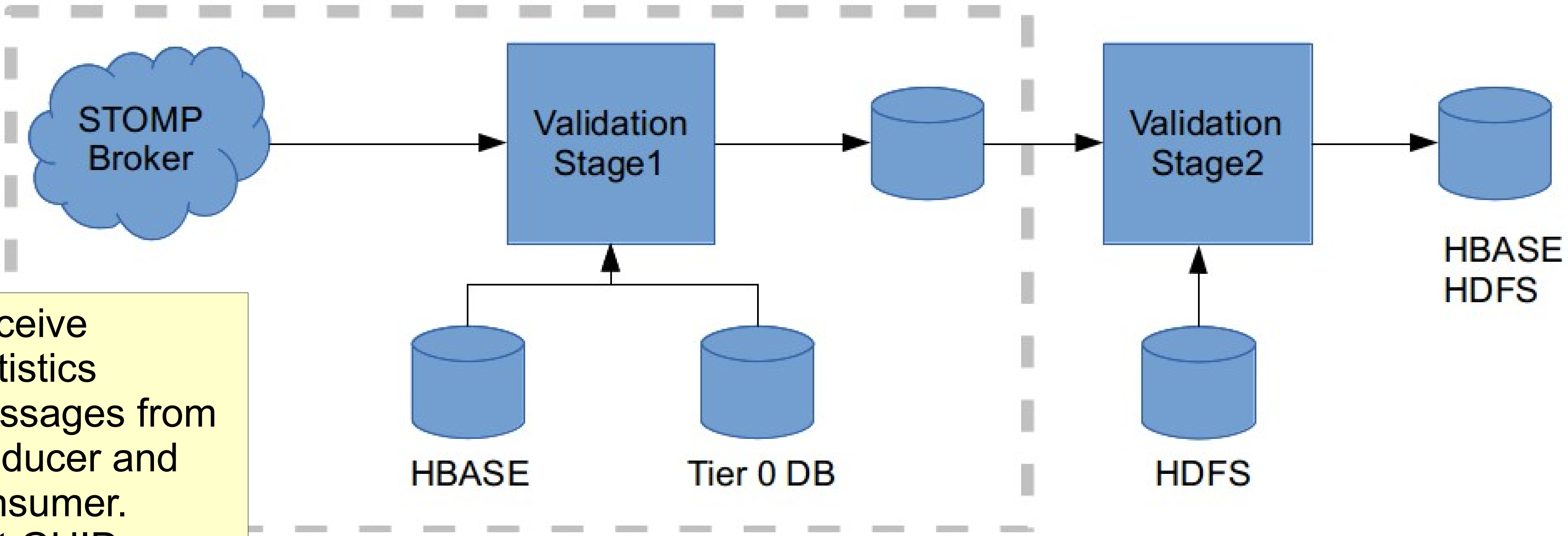
HDFS files are created by GUID in both, CSV and MAPFILE format

Event navigational info (POOL TOKEN) is stored in HBASE

Consumer sends back statistic messages (# messages, # files, # events) to broker to report status. Agents can subscribe to those to monitor progress and healthy working.

Validation

Validation verifies that all Event Index for a collection (dataset) contains all the files (GUIDs) and all the events. Detect multiple processing of the same file and job failures.



Receive statistics messages from producer and consumer. Get GUIDs produced and consumed and number of events.

Get number of events per GUID

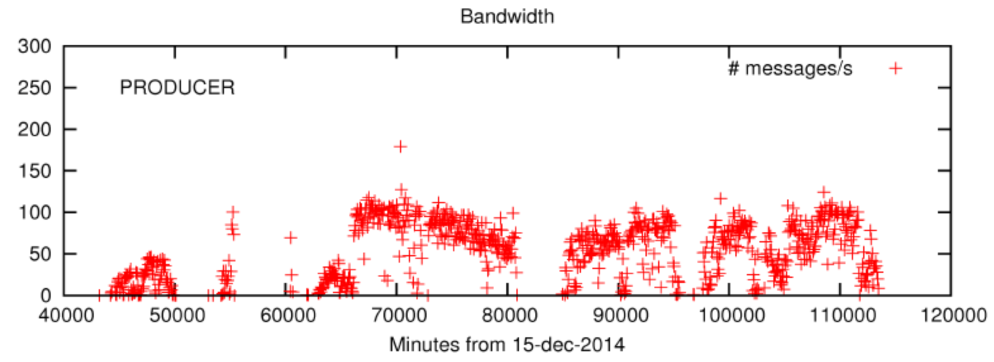
Get tasks (datasets) and number of jobs (files) from Tier 0 farm

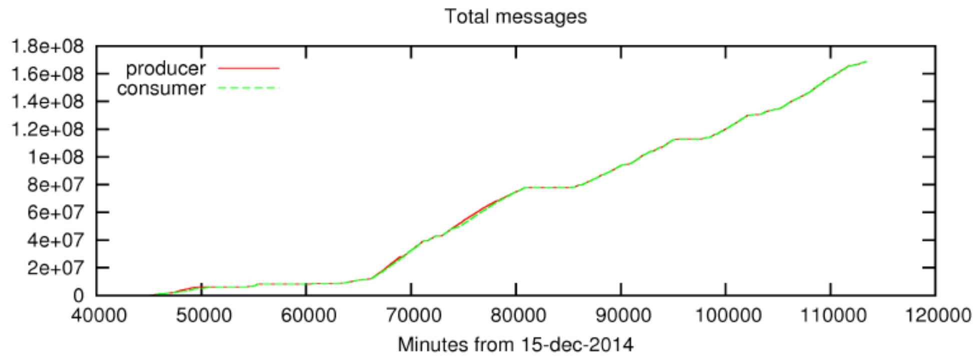
Check files present on HADOOP

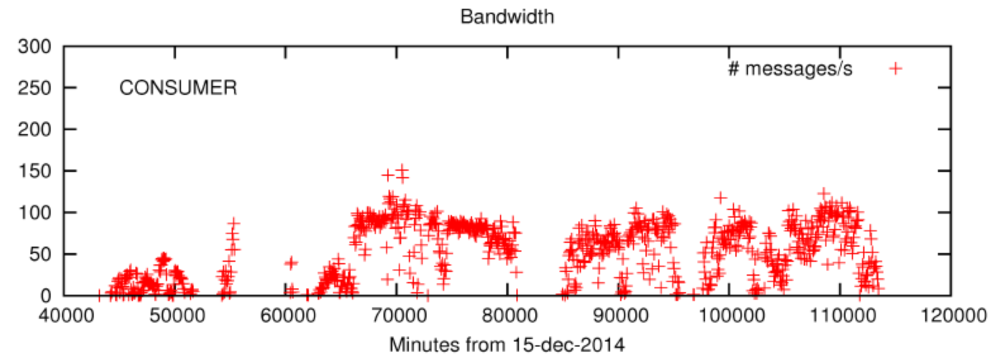
Signal Core to import data and make it available.

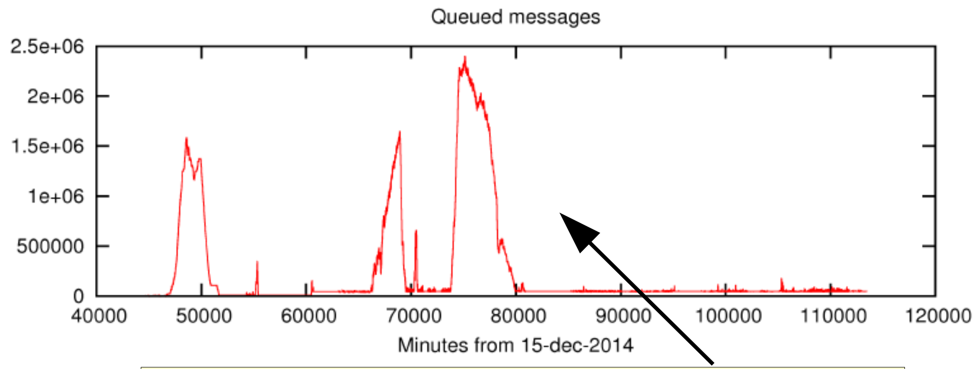
Future: Use AMI (ATLAS Metadata Interface) as a source of information in stage1 and make validation in a single step.

Performance









Tier 0 processing campaign from
11 January to 8 March
Sustained processing rate of 100
messages/second

Some backlog due to consumer
performance
Improved later with a refinement
of writing algorithm

Using
- 3 brokers @ CERN
- 3 consumer (1 machine) @ CERN
- 1 monitor @ IFIC

- 472 K files produced/consumed
- 170 M messages
- 5,2 G events
- 1,8 Tbytes

Conclusions

- The ATLAS EventIndex Distributed Data Collection presented in this talk:
  - Underwent final fine tuning in recent months
    - Using recent cosmic ray data taking and processing
  - Has now reached production-level operations
  - Is actively loading all the most recent data
  - Is able to keep up with production processing rates
  - Delivers all the data needed to satisfy the use cases for currently produced data formats

  → Ready for LHC collision operations event metadata collection !
- Next steps:
  - Include AMI database for validation
  - Integrate EI transform into ATLAS processing chain as a substep
  - Extend producer capability to extract event info for other ATLAS data formats not supported yet