NEC'2019

EI3 – The ATLAS EventIndex for LHC Run 3

Evgeny Alexandrov [1], Igor Alexandrov [1], Zbigniew Baranowski [2], Dario Barberis [3], Gancho Dimitrov [2], Álvaro Fernández Casaní [4], Elizabeth Gallas [5], Carlos García Montoro [4], Santiago González de la Hoz [4], Julius Hrivnac [6], Andrei Kazymov [1], Mikhail Mineev [1], Fedor Prokoshin [1], Grigori Rybkin [6], Javier Sánchez [4], José Salt Cairols [4], Miguel Villaplana [7]

1: JINR Dubna, 2: CERN, 3: Univ./INFN Genova, 4: IFIC Valencia,
5: Univ. Oxford, 6: LAL Orsay, 7: University of Alberta

ATLAS Events

EventIndex is a system designed to be a complete catalog of ATLAS events, real and simulated data

- Event is the basic unit of ATLAS data
- Each event contains result of a single triggered interactions, plus eventually piled-up interaction
  - Signals from the detector
  - Reconstructed particles with their parameters
  - Trigger decisions
- Uniquely identified by the run number and the event number
- Event information is stored in many instances
- Have different formats to fit analyses needs
- Spread among the hundreds of GRID sites

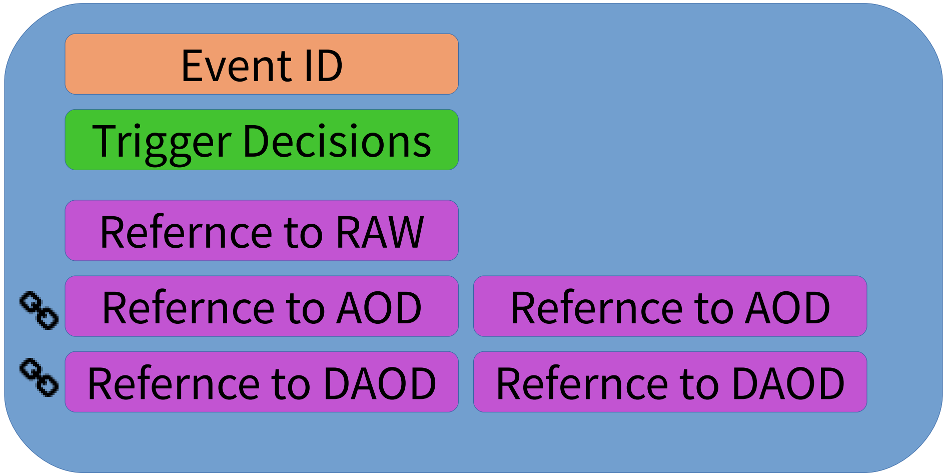
ATLAS EventIndex

- The ATLAS Experiment produces large amounts of data
  - several billion events per year
  - a database containing references to the events is necessary in order to efficiently access them from the distributed data storage
- The ATLAS EventIndex
  - provides a way to collect and store event information using modern technologies
  - provides various tools to access this information through command line, GUI and RESTful API interfaces
- An infrastructure was created that
  - allows fast and efficient selection of events of interest from the billions of events recorded, based on various criteria
  - provides an indexing system that points to these events in millions of files scattered through a worldwide distributed computing system
  - automatically detects and indexes new data that was produced in this system from events collected from the detector or simulated

EventIndex record

EventIndex records contain the following fields:

- **Event identifiers**
  - **Run and event number**
  - **Trigger stream**
  - **Luminosity block**
  - **Bunch Crossing ID (BCID)**



- **Trigger decisions**
  - **Trigger masks for each trigger level**
  - **Decoded trigger chains (trigger condition passed)**
- **References to the events at each processing stage in all permanent files generated by central productions** (for event picking)

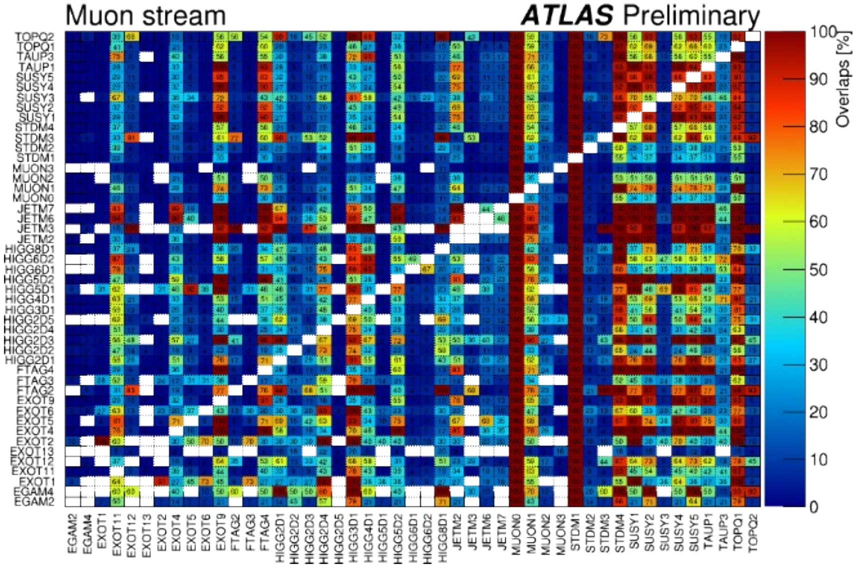
ATLAS Datasets

ATLAS event data is written in files that are organized in datasets

- Datasets can have different format depending of the processing stage
  - Detector data is first written in the RAW format
  - AOD datasets are produced after reconstruction.
  - Derived datasets (DAOD) for use in the specific analyses
- + Simulated (MC) datasets are produced on the GRID
  - EVNT datasets contain particles information
  - RDO contains detector signals
- There are various versions of the datasets originating from the same detector or simulated events
  - Different reconstruction settings and software version
- Reprocessing roughly yearly

Use cases

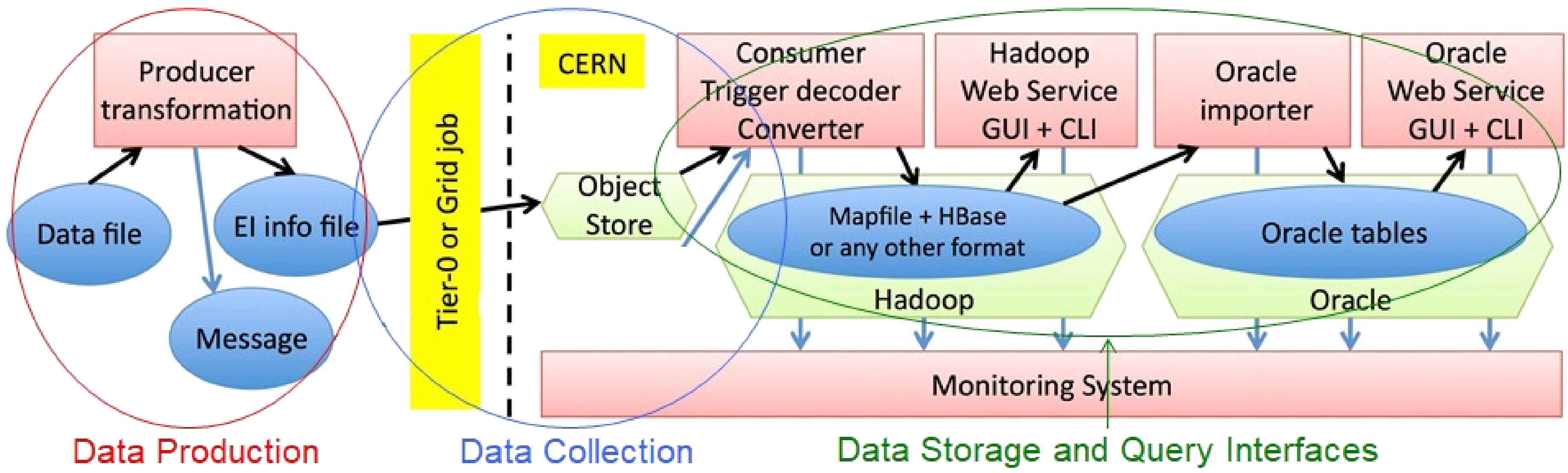
- Event picking
  - give me event in specific format and processing version
- Count and select events based on trigger decisions
- Production completeness and consistency checks
  - Data corruption, missing and/or duplicated events
- Trigger chain overlap counting
- Derivation overlap counting
- Dataset Browsing
  - Finding datasets of interest
  - Dataset report
  - Dataset Inspection



Summary of Use Cases here:

https://twiki.cern.ch/twiki/bin/view/AtlasComputing/EventIndexUseCases

EventIndex Architecture



partitioned architecture, following the data flow

Data production

- extract event metadata from files produced at Tier-0 or on the Grid

Data collection

- transfer EI information from jobs to the central servers at CERN

Data storage

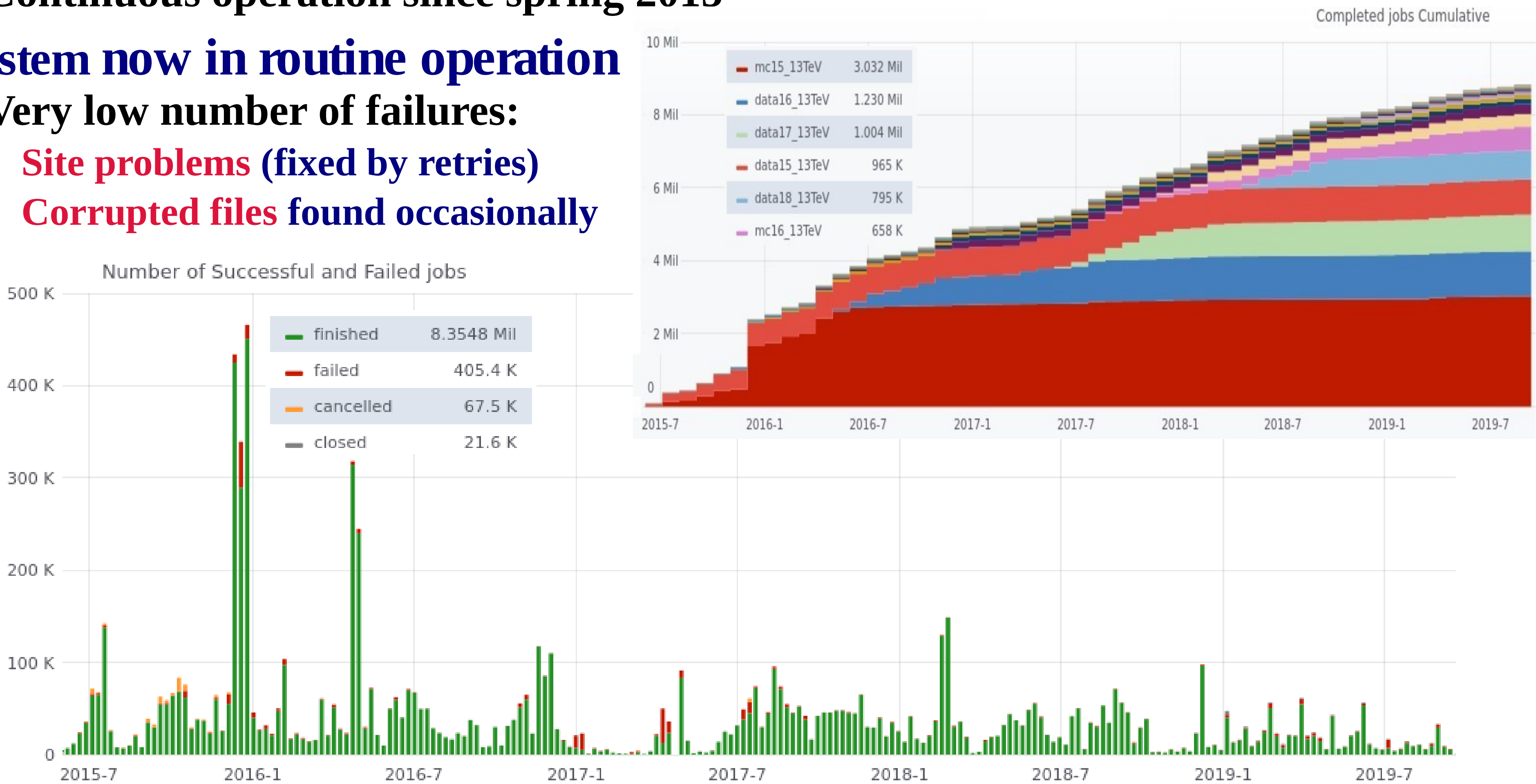
- Provide permanent storage for EventIndex data.
- full info in Hadoop; reduced info (only real data, no trigger) in Oracle
  - fast access for the most common queries, reasonable time response for complex queries

Monitoring

- keep track of the health of servers and the data flow

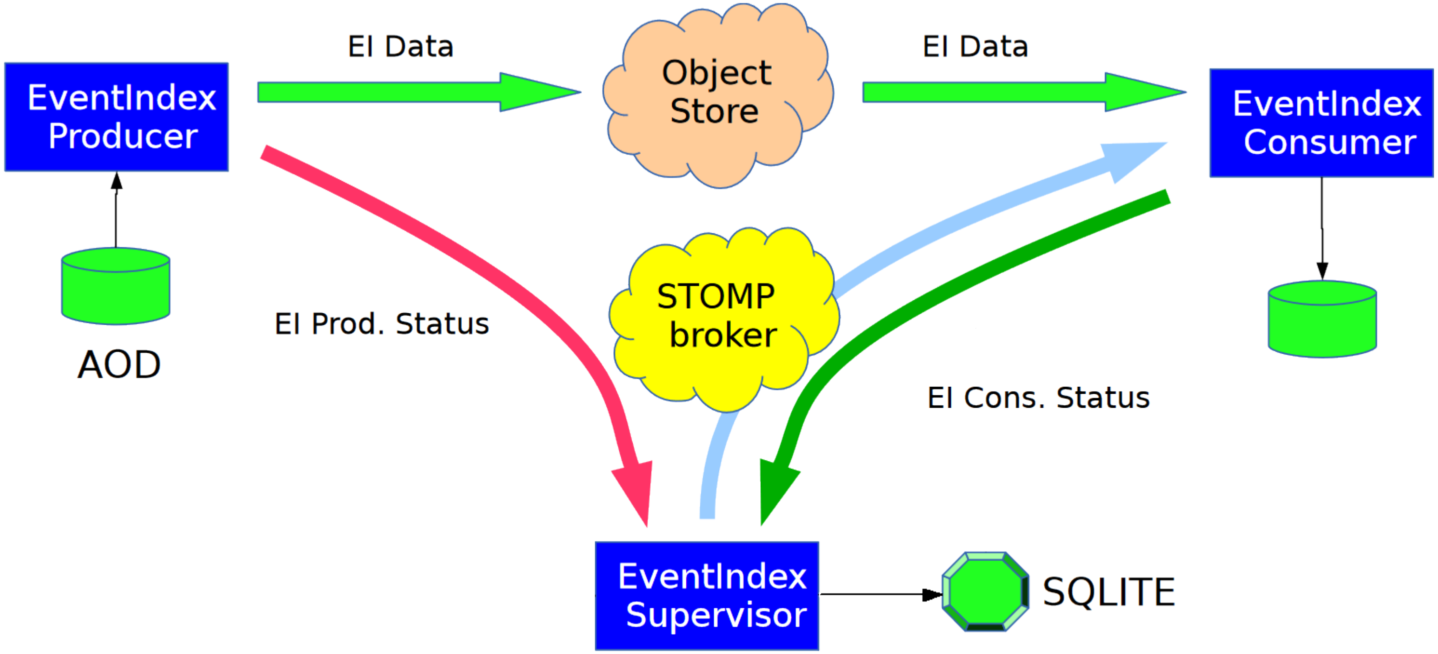
Data production

- **Tier-0 jobs index merged physics AODs, collecting also references to RAW data**
- **Similarly, Grid jobs collect info from datasets as soon as they are produced and marked "complete" in ATLAS Metadata Interface (AMI)**
  - **Other data formats (HITS, DAOD etc.) can be (and are) indexed on demand**
  - **Continuous operation since spring 2015**
- **System now in routine operation**
  - **Very low number of failures:**
    - **Site problems (fixed by retries)**
    - **Corrupted files found occasionally**



Data Collection

- **Producer: Athena Python transformation, running at Tier-0 and grid-sites. Indexes dataset data and produces an EventIndex file**
  - **EI Information is sent by each job as a file to the ObjectStore at CERN (CEPH/S3 interface) as intermediary storage.**
  - ***Google protobuf* data encoding (compressed)**
- **Supervisor: Controls all the process, receives processing information and validates data by dataset.**
  - **Signals valid unique data for ingestion to Consumers.**
  - **Operated with a web interface**
- **Consumers: Retrieve ObjectStore data, group by dataset and ingest it into Hadoop storage**

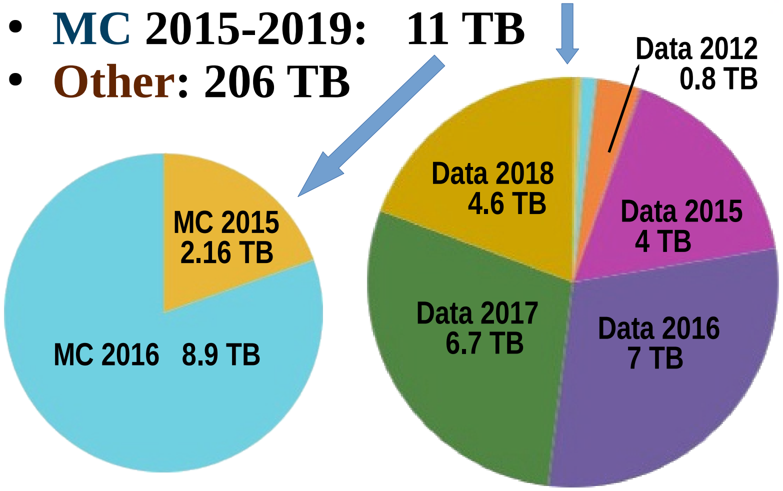


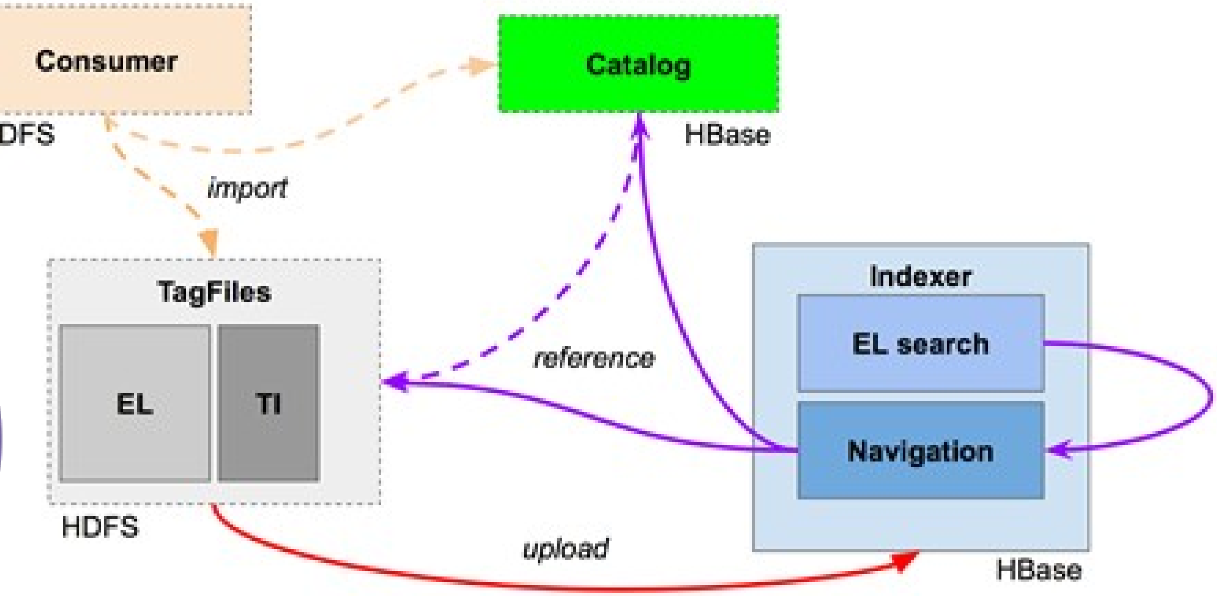
Hadoop Storage

- Hadoop is the baseline storage technology
  - It can store large numbers (10-s of billions) of simply-structured records
  - and search/retrieve them in reasonable times
- Hadoop "MapFiles" (indexed sequential files) are used as data format
  - One MapFile per dataset
  - Internal catalogue keeps track of what is where and dataset-level metadata (status flags)
  - Event Lookup index
- CLI, RESTful API and GUI interfaces available for data inspection, search and retrieval

Data volumes:

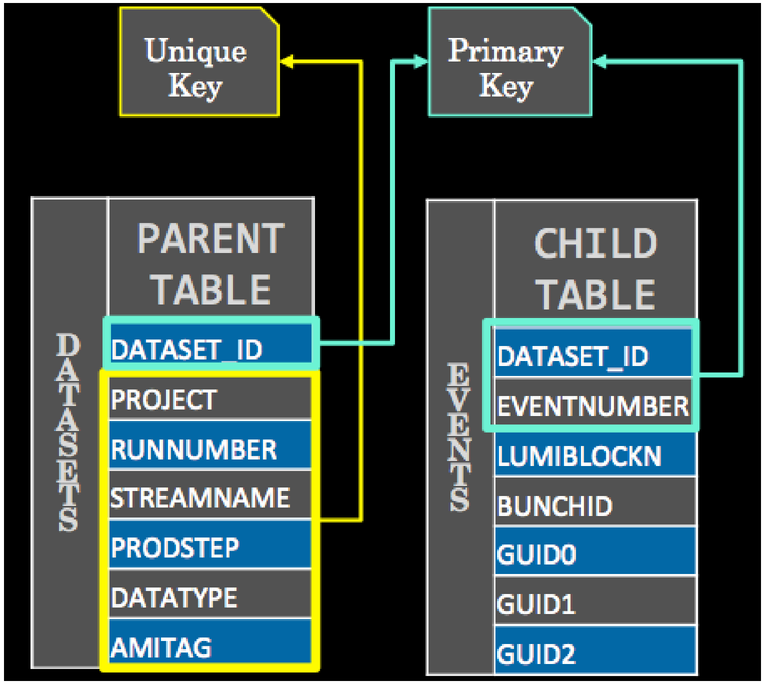
- Real data 2009-2019: 24 TB
- MC 2015-2019: 11 TB
- Other: 206 TB





Oracle storage

- Simple schema with dataset and event tables
  - Exploiting the relational features of Oracle
- Filled with all real data, only event identification and pointers to event locations
  - Optimized for event picking
  - Very good performance also for event counting by attributes (LumiBlock and bunchID)
- Connection to the RunQuery and AMI databases
  - to check dataset processing completeness
  - and detect duplicates
- Easy calculation of dataset overlaps
- GUI derived from COMA database browser to search and retrieve info



77 k Datasets (185 Billion event records)

stored within 3.4 TB table segments + 3.1 TB auxiliary index

Next generation EventIndex

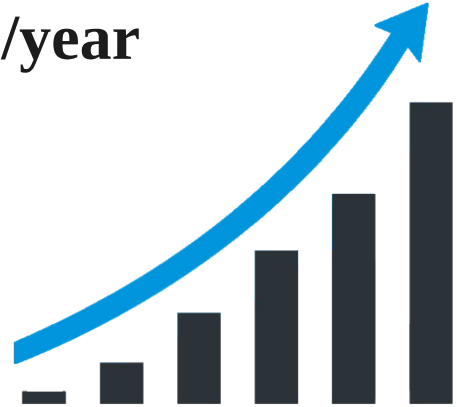
- An evolution of the EventIndex concepts
  - Currently: the same event across each processing step (RAW, ESD, AOD, DAOD, NTUP) is physically stored at different HADOOP HDFS files
  - Future: One and only one logical record per event
    - Event Identification, Inmutable information (trigger, lumiblock, ...)

    and for each processing step:
    - Link to algorithm (processing task configuration)
    - Pointer(s) to output(s)
    - Flags for offline selections (derivations)
- Support virtual dataset
  - A logical collection of events
    - Created either explicitly (giving a collection of Event Ids)
    - or implicitly (selection based on some other collection or event attributes)
  - Labeling individual events by a process or a use with attributes (key:value)

Next generation EventIndex

Evolve EventIndex technologies to future demanding rates

- **Currently:**
  - ALL ATLAS processes: ~30 billion events records/year
  - up to 350 Hz on average
  - This is update rate through the whole system
    - all years, real and simulated data
  - Read 8M files/day and produce 3M files



- **Future:**
  - Due to expected trigger rates, need to scale for next ATLAS runs:
  - At least half an order of magnitude for Run3 (2021-2023):
    - 35 B new real events/year and 100 B new MC event/year
  - An order of magnitude for Run4 (2026-2029):
    - 100 B new real events and 300 B new MC events per year
  - Then sum up replicas and reprocessing

New Use Cases

- **Adding “offline trigger” information:**
  - Store the results of selections that can be used to form derived datasets
    - Needs the ability to add info to part of event record
- **Using offline trigger information:**
  - Select events using online and offline trigger information to build a “Virtual dataset”
  - Use massive event picking to physically retrieve events belonging to a virtual dataset (probably in AOD format, but also RAW if very few) and continue the analysis with more info on reduced size datasets
    - Useful if selecting <1% of the events
- **Massive event picking:**
  - Selection of many events, touching a large fraction (or all) of the files in a dataset
  - May need a dedicated service, especially if input on tape (RAW)
- **Partial processing for production tests:**
  - May skip some input checks and then assign a finite lifetime to the information (delete once the test is done)

WORK IN PROGRESS

EI Evolution. HBase

- Apache HBase is the Hadoop database, a distributed, scalable, big data store.
  - Open-Source, distributed, versioned, non-relational database modeled after the Google BigTable paper
  - Built on top of HDFS and provides fast record lookups (and updates) for large tables
- HBase organizes data into tables.
  - Tables have rows and columns, which store values (like a spreadsheet)
    - ✓ Rows are identified uniquely by their row key
    - ✓ Each row can have a different schema
    - ✓ Data within a row is grouped by column family. Must be defined up front and not easily modified
- RowKey design desirable properties:
  - Have a small size
  - Should identify an event uniquely inside a ‘container’ (event picking)
  - Allow searches reading the smallest quantity of consecutive data using ‘range searches’
  - Avoid sparse insertions into existing RowKey space
  - Try to use all regions evenly both, reading and writing
  - When growing, use the RowKey space homogeneously

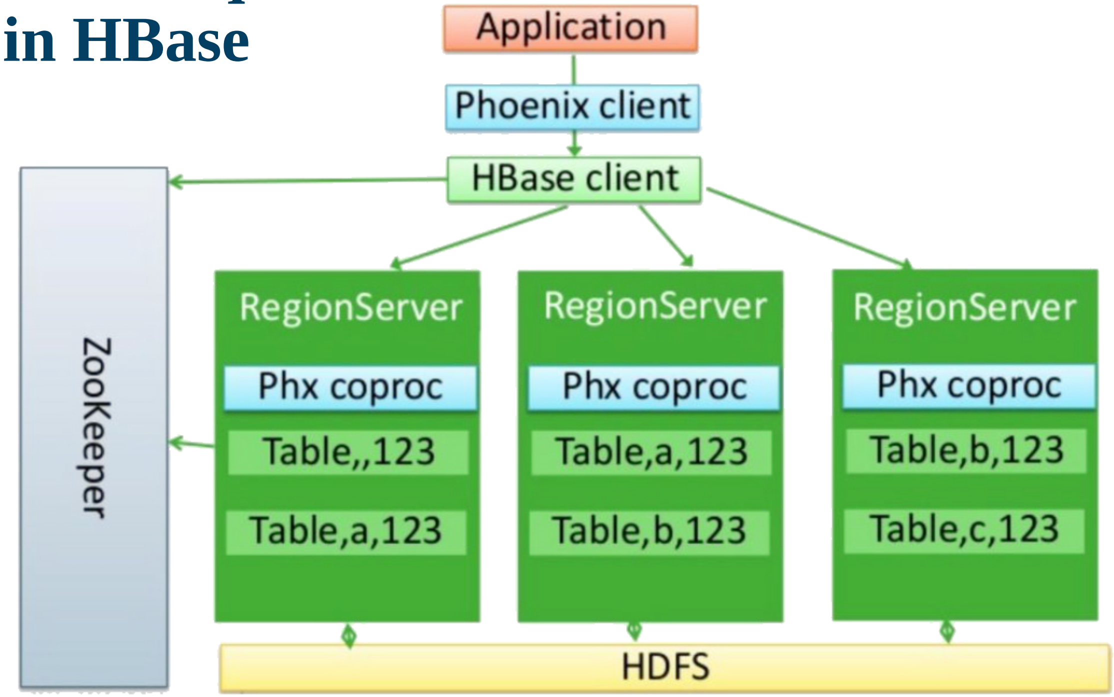
SQL on HBase

- HBase belongs to noSQL databse family
  - was created because of scalability problems
  - when data model is trivial - simple key-value store could satisfy it
- on the other hand SQL/structured schemas provides
  - structured data are easy to understand and maintain
  - standard declarative query logic, ‘optimized’ for complex queries
- Various possibilities for SQL on HBase
  - Apache Impala
  - Apache Hive

    } handling of a row key mapping has to be on the application side
  - Apache Spark - mainly for batch jobs
- Apache Phoenix
  - SQL layer on top of HBase. It provides:
    - structured schema of the tables instead of schemaless freeride
      - mapping of columns to HBase cells
      - serialization of data types to bytes
    - SQL planner and optimizer
    - built-in HBase related optimizations
    - server-side (optimized) executions
    - access via JDBC

Apache Phoenix

- OLTP and operational analytics for HBase through SQL
  - Takes SQL query
  - Compiles it into a series of HBase scans Direct use of the HBase API, along with coprocessors and custom filters
  - Produces regular JDBC result sets
- RowKey design can be adapted to phoenix's types and sizes
  - losing some performance
- Phoenix allows use of RowKey fields in queries but they are stored as one entity in HBase
- A number of test has been performed:
  - Loading Atlas EventIndex data to HBase via Phoenix
  - Phoenix queries on loaded data
  - These results are encouraging
  - Some basic functions are ready
  - Further work on performance and user interfaces is ongoing



EI3 – The ATLAS EventIndex for LHC Run 3

- The EventIndex project started in 2012 at the end of LHC Run 1 driven by the need of having a functional event picking system for ATLAS data
  - The data storage and search technology selected in the first phase of the project (Hadoop MapFiles and HBase, in 2013-2014) was the most advanced available at that time in the fast-growing field of BigData and indeed after a couple of initial hiccups it proved reliable and performed satisfactorily
    - Part of the data are replicated also to Oracle for faster access but mainly to have a uniform environment between event and dataset metadata
- Nevertheless the current implementation of the EventIndex started showing scalability issues as the amount of stored data increases
  - Slower queries, lots of storage (now eased by compression)
- Significant increase in the data rates expected in future LHC runs demands transition to a new technology
- Phoenix queries and HBase new event table prototypes have been tested, and show encouraging results
  - There is table schema candidate
  - Basic functionality is ready. Working towards improved performance and better interfaces
  - Need to keep testing with more data and get performance metrics
- The plan is to have the new system operational by the middle of 2020 in parallel with the old one, and phase out the old system at the end of 2020 (well in advance of the start of LHC Run 3)