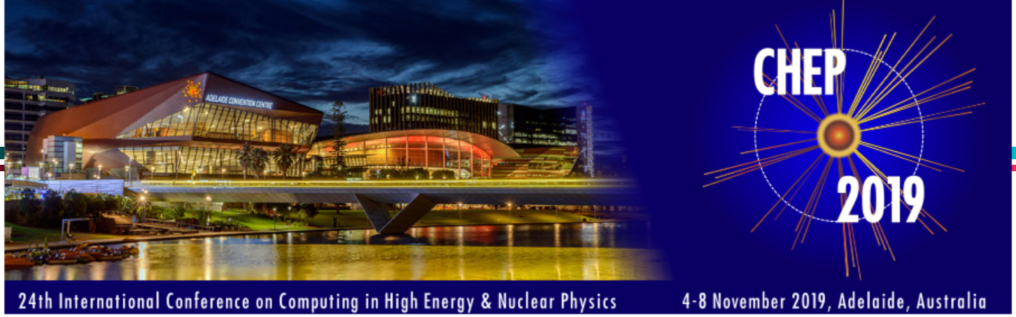

CHEP
2019
24th International Conference on Computing in High Energy & Nuclear Physics
4-8 November 2019, Adelaide, Australia

The ATLAS EventIndex for LHC Run 3

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

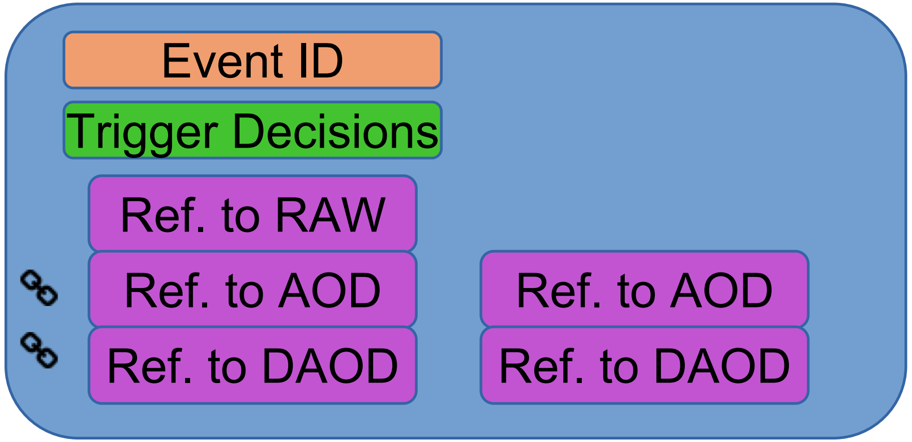
1: Univ./INFN Genova, 2: JINR Dubna, 3: CERN, 4: IFIC Valencia, 5: Univ. Oxford, 6: LAL Orsay, 7: Univ. of Alberta

The ATLAS EventIndex

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

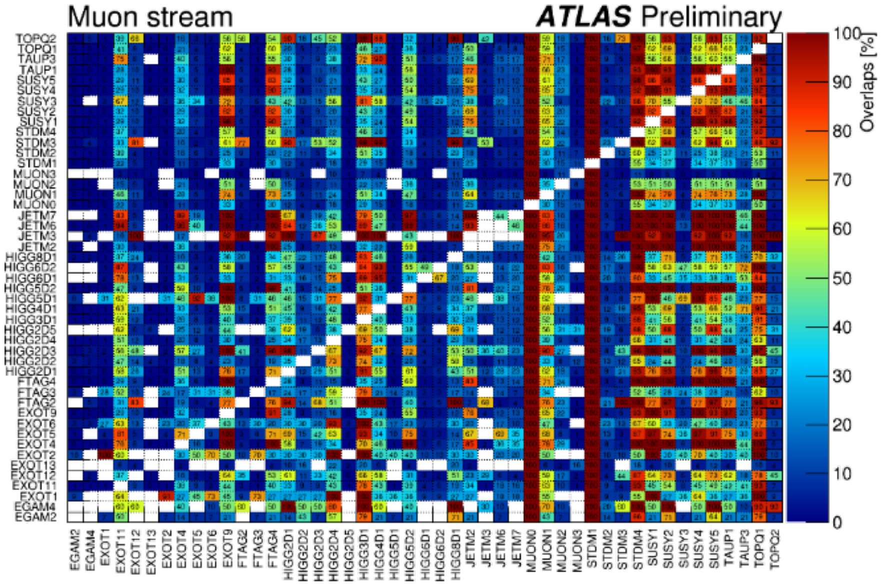
EventIndex records

- 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)

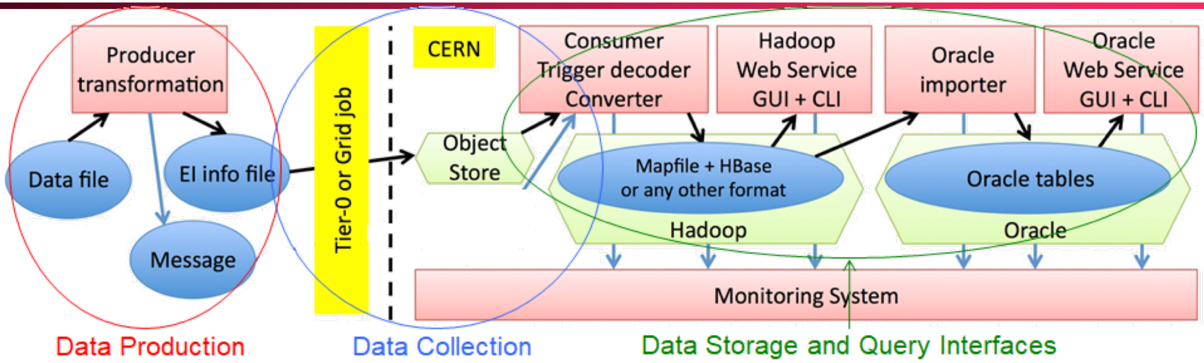


Use cases

- Event picking
  - Give me this event in a specific format and processing version
- Count or 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 $\longrightarrow$
- Dataset Browsing
  - Finding datasets of interest
  - Dataset report
  - Dataset Inspection



EventIndex Architecture



Partitioned architecture, following the data flow:

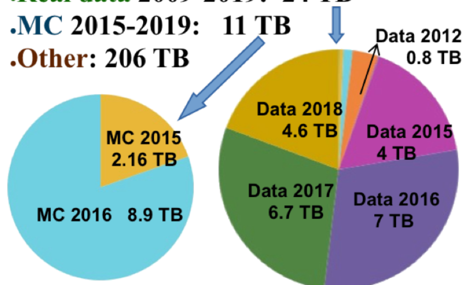
- Data production
  - Extract event metadata from files produced at Tier-0 or on the Grid
- Data collection
  - Transfer EI information from jobs to the central servers at CERN
- Monitoring
  - Keep track of the health of servers and the data flow

- 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

Current storage implementations

- Hadoop is the baseline storage technology
  - It can store large numbers (10 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 in HBase keeps track of what is where and dataset-level metadata (status flags)
  - Event Lookup index (also in HBase)

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



- CLI, RESTful API and GUI interfaces available for data inspection, search and retrieval

- 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 ATLAS 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

77k Datasets (185 Billion event records)
stored within 3.4 TB of table segments
plus 3.1 TB of auxiliary index

Next generation EventIndex (1)

- The current EventIndex storage implementation reflects the state of the art for BigData storage in 2012-2013 when the project started
  - Many different options appeared since then!
- Currently: the same event is physically completely stored in different Hadoop MapFiles for each data format and processing version
- But data-taking rates will increase x10 for HL-LHC
  - Target 100B new real and 300B new simulated events/year
- Future: one and only one logical record per event
  - Event ID, immutable information (trigger, lumiblock, ...)
  - and then for each processing step:
    - Link to algorithm (processing task configuration)
    - Pointer(s) to output(s)
    - Flags for offline selections (derivations)

Next generation EventIndex (2)

New use cases became important in the last few years:

- Massive event picking:
  - Selection of many events, touching a large fraction (or all) of the files in a dataset
  - Will need a dedicated service, especially if input on tape (RAW)
- 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
  - Select events using online and offline trigger information to build a "virtual dataset"
- Support for virtual datasets
  - 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)
- Labelling individual events with attributes (key:value)

EventIndex evolution: Hbase (1)

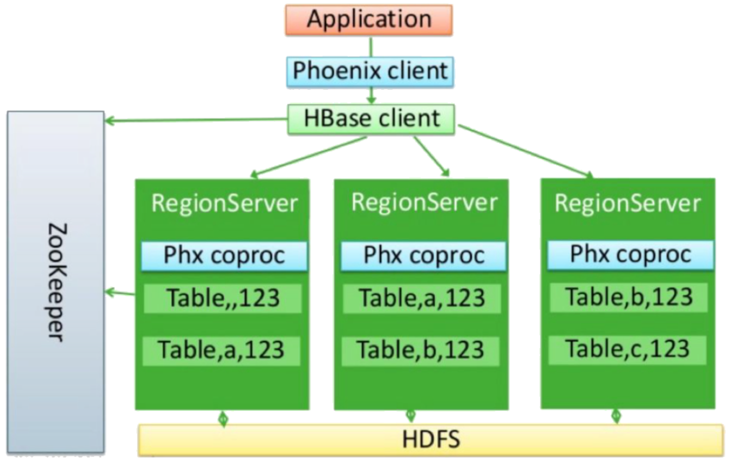
- Apache HBase is the Hadoop database, a distributed, scalable, big data store.
  - Open-source, distributed, versioned, non-relational database modelled after the Google BigTable paper
  - Built on top of HDFS, provides fast record lookups (and updates) for large tables
- HBase organizes data into tables
  - 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 upfront and not easily modified

EventIndex evolution: Hbase (2)

- HBase belongs to NoSQL database family
  - When data model is trivial, simple key-value store could satisfy it
- But SQL-structured schemas provide added values
  - structured data are easy to understand and maintain
  - standard declarative query logic, 'optimized' for complex queries
- Various possibilities for SQL on HBase
  - Apache Impala, Hive, Spark
- Apache Phoenix
  - SQL layer on top of HBase. It provides:
    - structured schema of the tables instead of schemaless free ride
    - 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 (Java DB Connectivity)

Apache Phoenix for the EventIndex

- 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
- HBase RowKey design must be adapted to Phoenix's types and sizes
  - losing "some" performance
- A number of test have been performed:
  - Loading ATLAS EventIndex data to HBase via Phoenix
  - Phoenix queries on loaded data



- Results are encouraging:
- Single event picking in 30 ms
- Full dataset queries in 6-10 seconds

Some basic functions are ready

Further work on performance and user interfaces is ongoing

	ROW KEY					Event Location Family		Provenance	L1 Trigger								High Level Trigger					
column	ds pid	ds type id	ds subt ypei d	eve nt no	seq	tid	sr	pv	lb	bcid	lpsk	eti me	id	tbp	tap	tav	lb	bcid	hps k	ph	pt	rs
bytes	4	1	1	8	2	4	24	26	4	4	4	12	8	[ ]	[ ]	[ ]	4	4	4	[ ]	[ ]	[ ]

Outlook

- Since the end of LHC Run 2 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)
- The 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 a good 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)