

HBase / Phoenix-based Data Collection and Storage for the ATLAS EventIndex

Carlos García Montoro^{1*}, Javier Sanchez¹, Dario Barberis², Santiago González de la Hoz¹, José Salt¹

¹Instituto de Física Corpuscular (IFIC) CSIC – University of Valencia, Valencia, Spain ²University of Genoa; **INFN** Genoa, Italy *Corresponding Author

The ATLAS EventIndex

The Data Collection

Is the catalogue of all ATLAS real and simulated events. Is the process that: • Deployment and Operation of the ATLAS **EventIndex for LHC Run 3** describes its architecture. • This poster is about its **data collection**, its **data** storage, and their evolution for Run 3.

Evolution

- 1/ Original Implementation at the beginning of Run 2:
- Back-end based on HDFS files organized per dataset.
- Data Collection exclusively based on messaging from producers to consumers.
- Cumbersome.
- Production is not re-playable.
- 2/ **ObjectStore Implementation** for Run 2:
- The Data Collection **Supervisor** is introduced to orchestrate the whole data collection.
- Accountability of data collection and validation.
- Amazon-S3-like **ObjectStore** at CERN replaces pure messaging architecture.
- Simple and scalable.
- Production can be stored, backed up, and replayed. 3/ HBase/Phoenix Implementation for Run 3:

- **1) Extracts** the metadata to index.
- 2) Validates it, assuring correctness and completeness.
- 3) Loads the metadata into the EventIndex back-end.

Data Flow of Data Collection

EventIndex Records

- **Event Records:**
 - ~533 Billion
- Files indexed:
 - ~32 Million
- Datasets indexed: ~279,000



dstypeid : sm

a.sr : binary(32

b.pv : binary(34)

d.bcid1 : intea d.hpsk : intege

d.ph : smallint[

a.mcc : integ a.mcw : floa

- Back-end based on **HBase**, compatible with **Phoenix**.
- Improved data structures.
- Simplified management.
- Has a SQL interface.
- Spark/Scala Loaders replace consumers.
- Better performance.
- Improved scalability and adaptability.
- Open Source Industry standards.

Event Record Table



Very optimized primary key

Balanced use of all regions and region servers.

NoSQL:

Relations are inferred

but not enforced.

- Locality of events of each dataset:
 - dspid and dstypeid identify datasets.
- Locality of derivations for overlaps:
 - Same dspid, different dstypeid.
- Seq is a CRC16 to record duplicates, if any.

Compact

• Primary key is small, 128 bits, and identifies the

Event Record Primary Key

• Dataset names are too long.

trigmenu

- Use artificial dataset identifiers instead.
- **dspid** and **dstypeid** are generated by the supervisor by means of the autoincremental feature of its RDBMS.
- Monotonically increasing keys are undesirable in HBase.
- Reverse the bits of dspid to populate 010 011

001

100

101

110

111

Storage Logical Architecture

datasets

dstypes

Fallback

canonica

events

lint	 runno : integer project : varchar(200) datatype : varchar(200) streamname : varchar(200) prodstep : varchar(200) version : varchar(200) tid : integer dspid : integer dstypeid : smallint 	
RRAY[]	<pre>smk : integer events_rucio : bigint files : integer events : bigint events_uniq : bigint events_dup : bigint files_dup : integer</pre>	
np	rank : smallint status : varchar(200) rucio_at : timestamp updated_at : timestamp dups_at : timestamp trigger_at : timestamp is_open : boolean is_deleted : boolean	
	has_raw : boolean has_trigger : boolean prov_seen : smallint ARRAY[] sr_cnt : varchar(200) sr_clid : varchar(200) sr_tech : varchar(200) name : varchar(250)	

datasets

canonical	
n • 4 • 2 • • 8 • 4 • 7 • 7 •	
smi eve file: eve eve file: ran stat ruc upo dup trig is_o has pro is_r sr_sr_sr_sr_sr_sr_sr_sr_sr_sr_sr_sr_sr_s	





0

010

c.lb : integer c.bcid : integer c.lpsk : integer c.etime:timestamp c.id : integer c.tbp : smallint[] c.tap : smallint[] c.tav : smallint[]

d.lb1 : integer d.bcid1 : integer d.hpsk : integer d.lph : smallint[] d.ph : smallint[]

dataset with a pair of numbers, no names. • Trigger stored as smallint arrays, no names.

Families to read just what is needed on each use case:

- a) For event picking.
- b) For provenance.
- c) For L1 trigger operations (count, overlap,...). d) For L2 and HLT trigger operations.
- **Compatible with Phoenix but without using** its exclusive features.

• Depend only on HBase, but not on Phoenix.

all the key space uniformly.

• Reserve the first bit to distinguish between data and mc.

• Reserve bits 2 to 4 for versioning.



Bit reverse

Phoenix Loaders and Importers

Write the EventIndex data into HBase/Phoenix event table. • Scala with Spark using **Resilient Distributed Dataset** API.

•Lazy •In phases **SPBDataSource** High parallelism • Resilient





MICINN supported this work under grant PID2019-104301RB-C21