Desarrollos y herramientas para el almacenamiento y procesado de grandes cantidades de datos aplicado en ATLAS Tier2 y EventIndex EI3

Álvaro Fernández Casaní

Servicios Informáticos

ATLAS Tier2

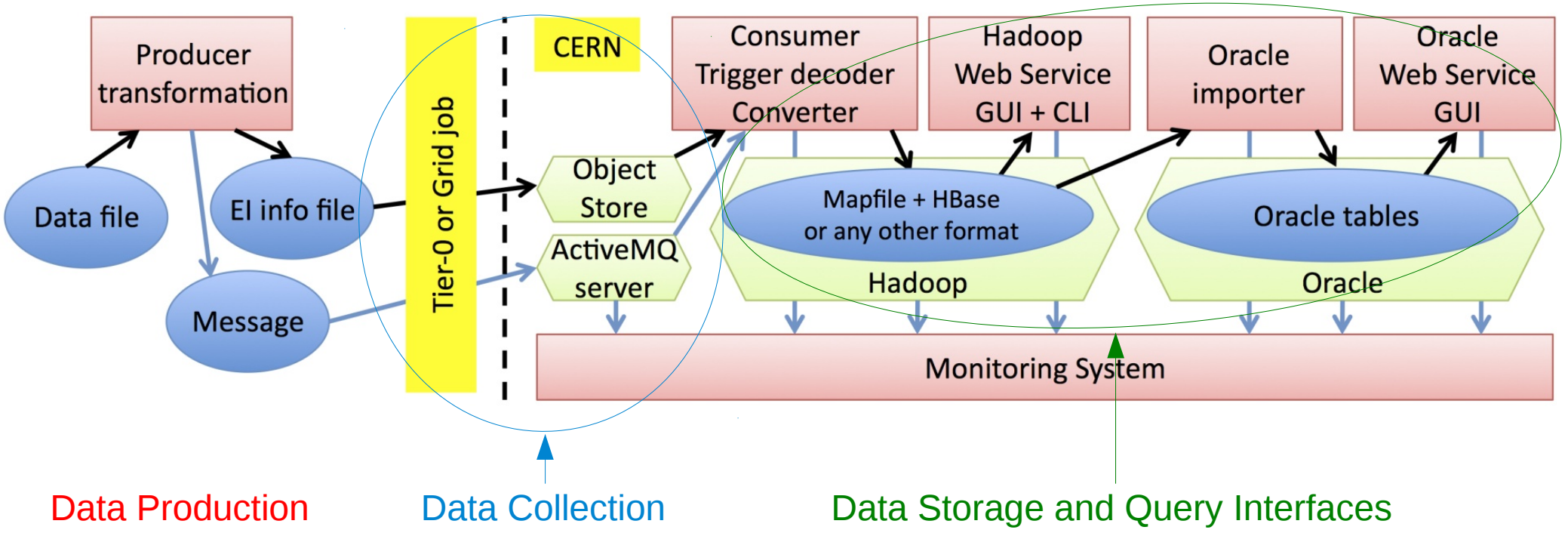
Jornadas Tecnológicas

Valencia, 11th March 2020

Contenido

- ATLAS EventIndex Architecture
- Object Store : CEPH
- Hadoop
- Hbase / Phoenix
- Entornos proceso de datos: MapReduce y Spark
- Monitorización: InfluxDB y Grafana
- Otros: GitLab

EventIndex Architecture

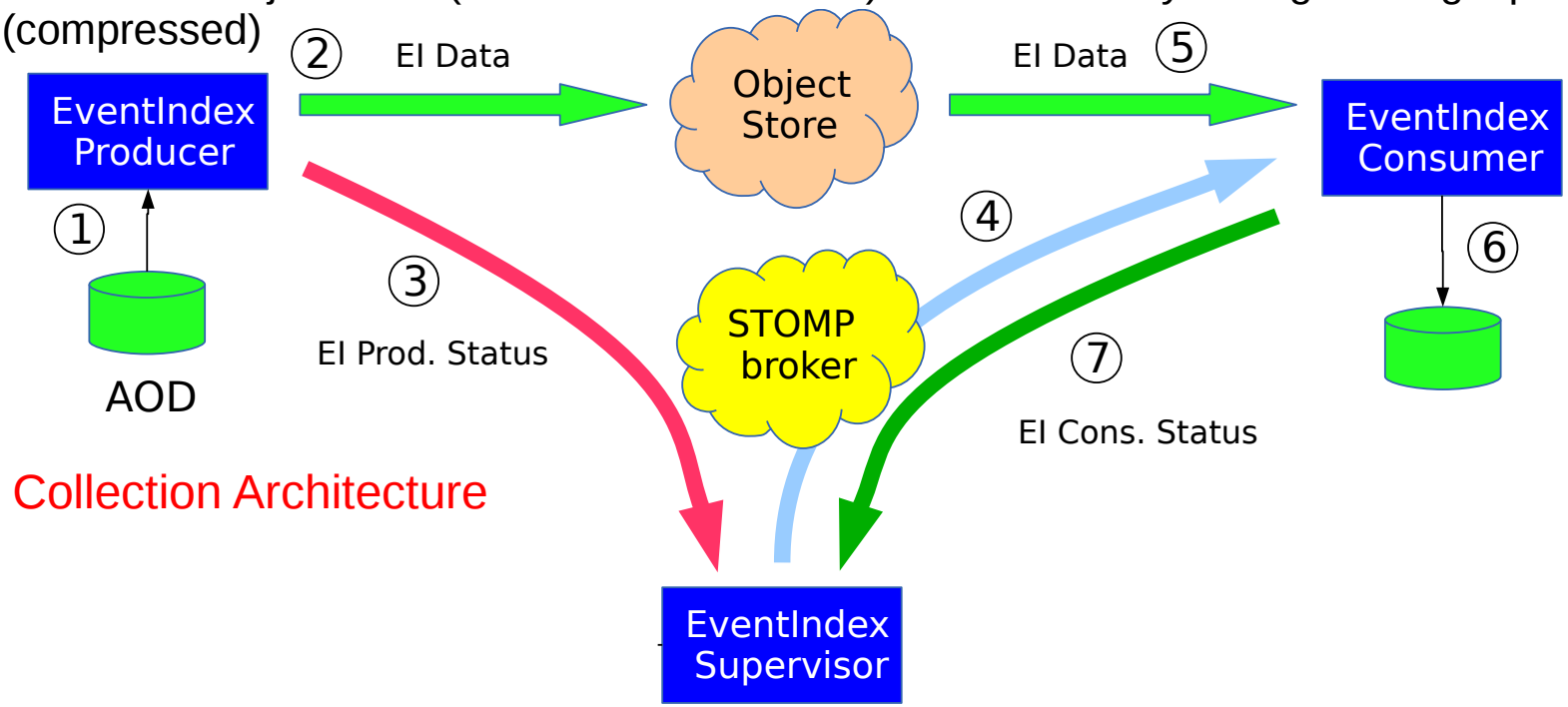


IFIC coordinates Data Production task to ensure all event index grid production performs correctly.

IFIC is responsible for all Data Collection task: a distributed producer/consumer architecture to collect all indexed data and ingest it to the Data Storage services.

ATLAS EventIndex Data collection

2015 - mid 2017: Pure Messaging Based architecture (ActiveMQ brokers / Stomp protocol). Json data encoding. (production peaks showed bottlenecks on messaging brokers)
mid 2017 onwards: ObjectStore (CEPH / S3 interface) as intermediary storage. Google protobuf data encoding (compressed)

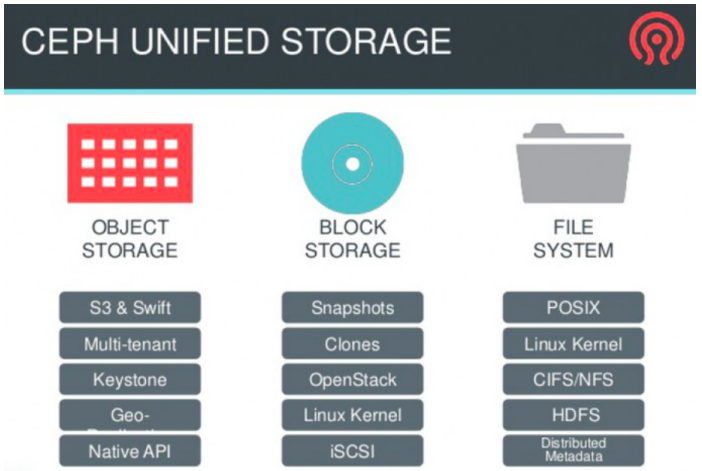


2020 Data Collection Architecture

Producer: Athena Python transformation, running at Tier-0 and grid-sites. Indexes AOD data and produces an **eventindex file**, stores in **ObjectStore**
Supervisor: Controls all the process, received processing information and validates data by dataset. Signals valid unique data for ingestion to Consumers. Operated with a web interface
Consumers: Retrieves ObjectStore data, groups by dataset and ingest it into **HDFS (Hadoop distributed Filesystem)**

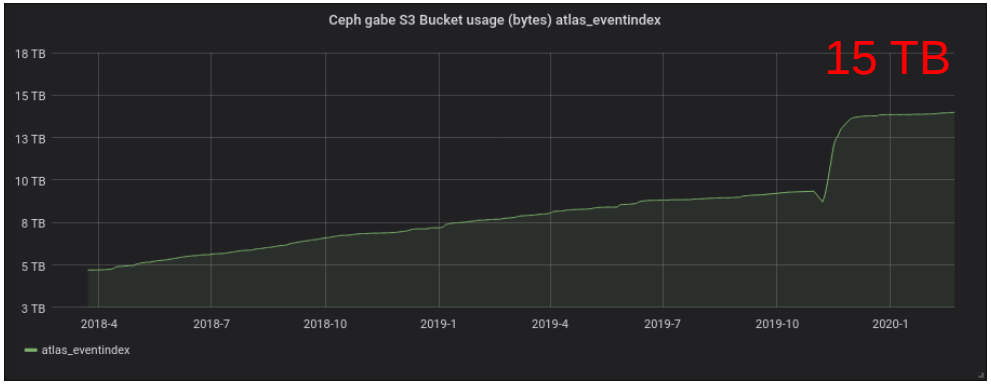
Object Store: CEPH

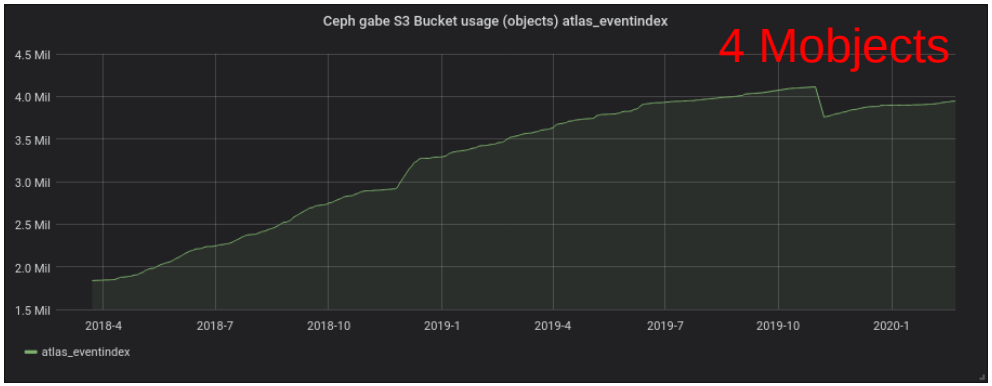
- Objects have a Name and are stored into **Buckets**. Flat structure (no hierarchy)
  - s3://**atlas_eventindex**/panda/20642610/20642610.G_4651828070.0_1791f9f3d73143aa959957eca97baa5d.ei.spb
- Scalability and Versatility
- Metadata (Key:Value) live with the object (ie: EI3 defined: #events, size, producer id, #files)
  - `x-amz-meta-uuid: 1791f9f3d73143aa959957eca97baa5d`
  - `x-amz-meta-startproctime: 1582301255663`
  - `x-amz-meta-endproctime: 1582301310107`
  - `x-amz-meta-nfiles: 29`
  - `x-amz-meta-size: 250604`
  - `x-amz-meta-usize: 12298971`
  - `x-amz-meta-inputdsname: mc16_13TeV:mc16_13TeV.361107.PowhegPythia8EvtGen_AZNLOCTEQ6L1_Zmumu.simul.HITS.e3601_e5984_s3126_tid18133419_00`
  - `x-amz-meta-nevents: 29000`
- Interfaces: Object (Amazon S3), Block Storage, File System.
- IFIC Services running on top: AFS, IBOX, Email Storage, Indico, ...



EventIndex Object Store usage

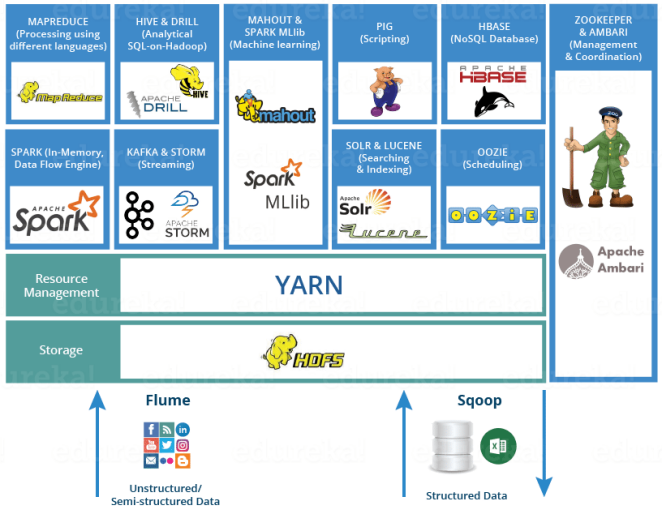
- 15 TB in 4 Million objects as of March 2020
- Pull-model approach for the distributed data collection of the ATLAS EventIndex project:
  - object store as a temporary storage
  - dynamic data selection = avoid consuming duplicated produced data
  - Avoid payload segmentation, store all info from producer in single object = Increased data compression ratios

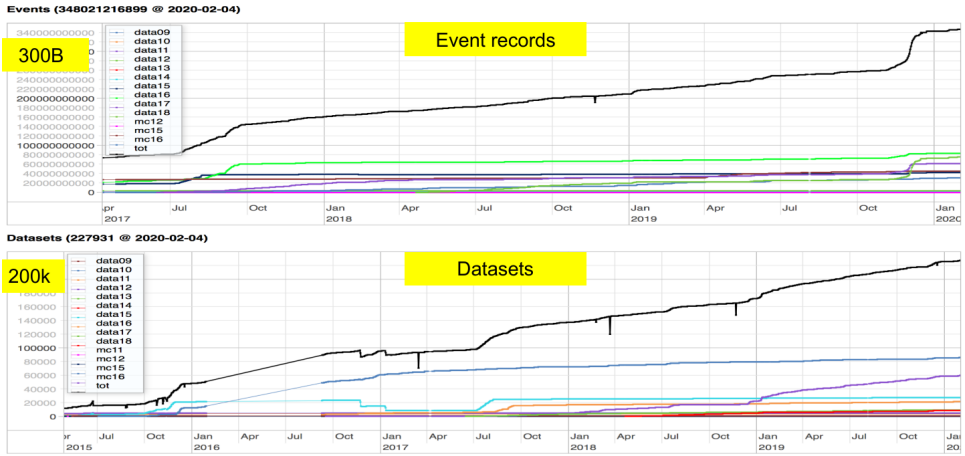




Hadoop

- Framework for big data storage and processing.
- Unstructured/semi-structured/structured data format (Data Lake)
- HDFS Filesystem: record-oriented big files. Distributed file/record processing with custom programs.
- EventIndex: dataset level Mapfiles (indexed sequential files) useful for analytics but no random-access.
  - Issues: some use cases require random access (event picking)
  - Fixed EI Event format can be exploited further. (see Hbase EI3 approach)
- Pull-model approach consuming from Object Store allows to consolidate directly in bigger files, without cleaning-up procedures.
- IFIC Testbed and CERN Production Cluster:
  - 18 Nodes/ 1.1 TB RAM
  - 1.3 PB capacity: 970 TB HDFS (3x replication)

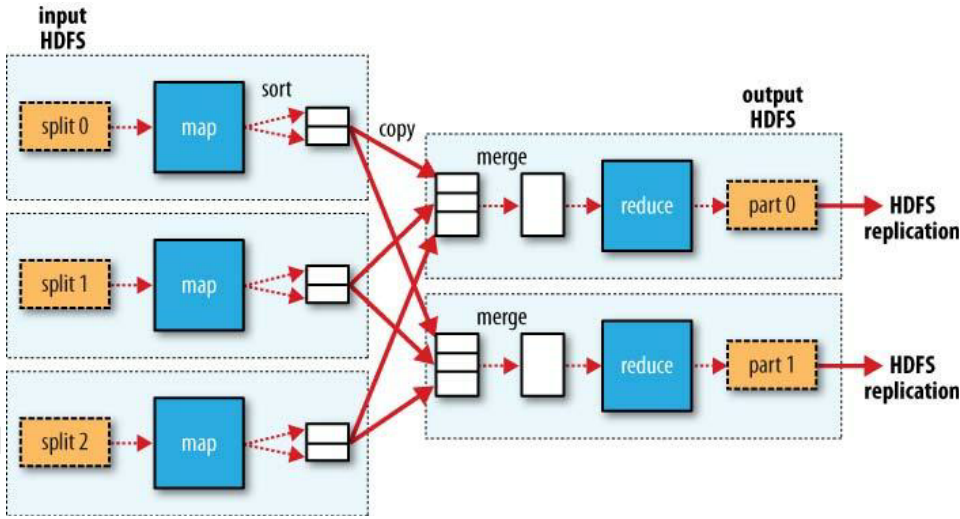


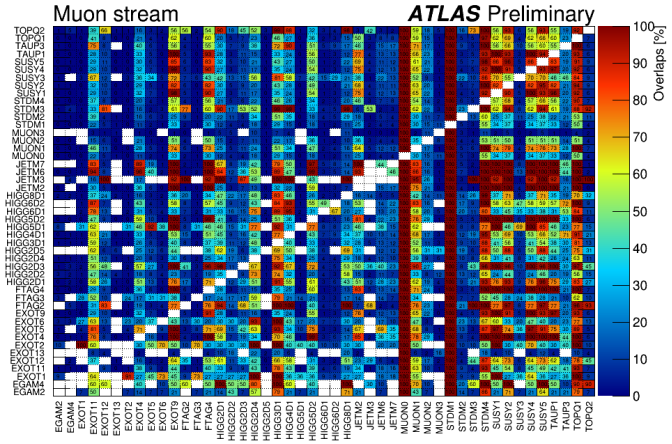


MapReduce Use Cases

Analytic workloads can be run with MapReduce over big amounts of data.

- **Overlap detection** in derivation framework: construct the overlap matrix identifying common events across the different files. Can be done with **2 MapReduce jobs.**
- **Job1: Obtain all derivations for particular events**
  - **Input data:** Mapfiles containing datasets and its derivations. 1 file can be divided in 1 or more splits, that will be the input to the Mapper.
  - **Map:** produces a duple for every input record with the info (event : derivation)
  - **Shuffle** (sort, copy, merge) from mappers
  - **Reduce:** with event ordered data, produce tuples (event : derivation#1, derivation#2,… derivation#n). This is input data to **Phase2**
- **Job2: count entries for every pair of derivation occurrences, and produce matrix**
  - **Map:**for every event produce n pairs (derivation_i,derivation_j : value) with a value identify is if a event is found in none, both, or with one derivation is into.
  - **Reduce**:Count overlaps for every (derivation_i,derivation_j) pair to produce matrix

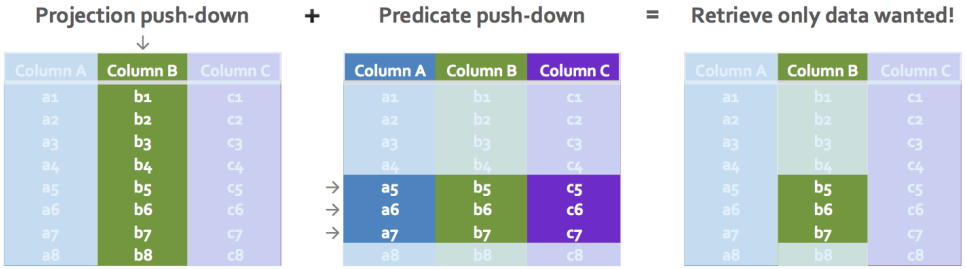




Storing and accessing thousands of files and millions of events in reasonable time is done with Hadoop Technologies.

HBase

- Some use cases require random access (Event picking). Previous approaches:
  - Oracle specific tables. Expensive,duplicate data (only real data, not MC) and complex procedures.
  - Hbase small table only with RunNumber/Event Location.
- In EI3 we generalize the approach storing all data in Hbase/Phoenix with a structured data schema.
  - Unify data for all use cases, reduce data duplicity, inconsistencies and complexity in procedures
- **Columnar storage**: improve compression, reduce I/0.
- Benefit from common access patterns:
  - Related data (reprocessing) sit close to each other on disc. Reduce redundancies and improve navigation.
  - Recent loaded data is the most accessed data (possibility to apply in-memory data caching)



HBASE – EI Event design

	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	[ ]	[ ]	[ ]

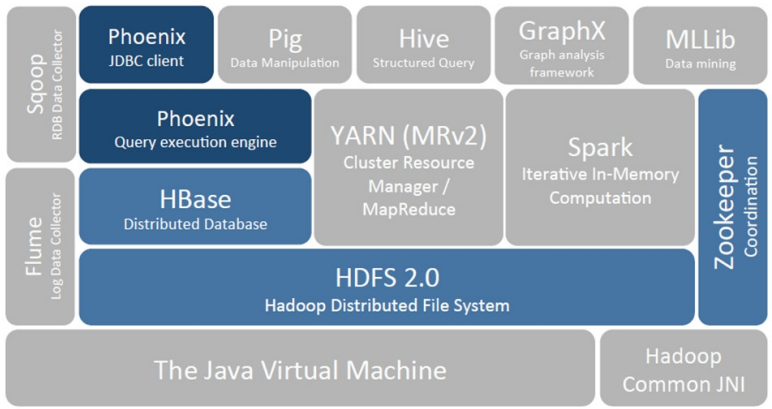
- Each event has a unique Row Key. Design is essential to for good sharding or distribution among servers and scale correctly. Types are adapted to Phoenix types and sizes:
  - **dspid (integer)** is assigned by the supervisor/consumer at loading time using: project, runNumber, streamName, prodStep, version
  - **dstypeid: (tinyint)** (RAW, ESD, DESD, AOD, DAOD, EVNT, EVGEN, HITS, RDO)
  - **dssubtypeid: (tinyint)** (NONE, BPHY1, BPHY4, …)
  - **eventno (bigint)**
  - **seq =** crc16 (guid:oid1-oid2) Allow event duplicates

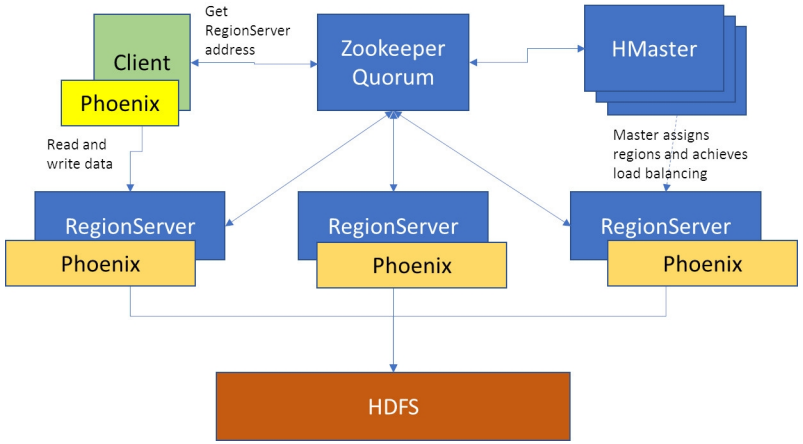
- Families represent *related* data that is stored together on the file system. Therefore, all column family members should have the same general access pattern
  - Event location.
  - Event provenance.
  - Level 1 trigger (L1).
  - High Level Trigger (HLT).

Phoenix

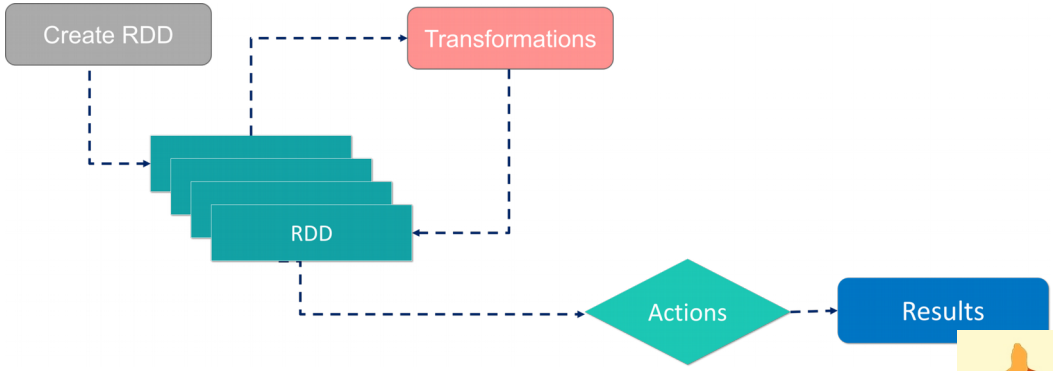
```
> select eventno,EI_REF0(sr,'full') from t0trig3.events where EI_TRIG0(d.11,'tav','5
and 45') limit 1;

| 1510912226 | 98336FF4-AAD0-7C42-9FD6-D01A98DD6655:00000261-000024C3 |
```

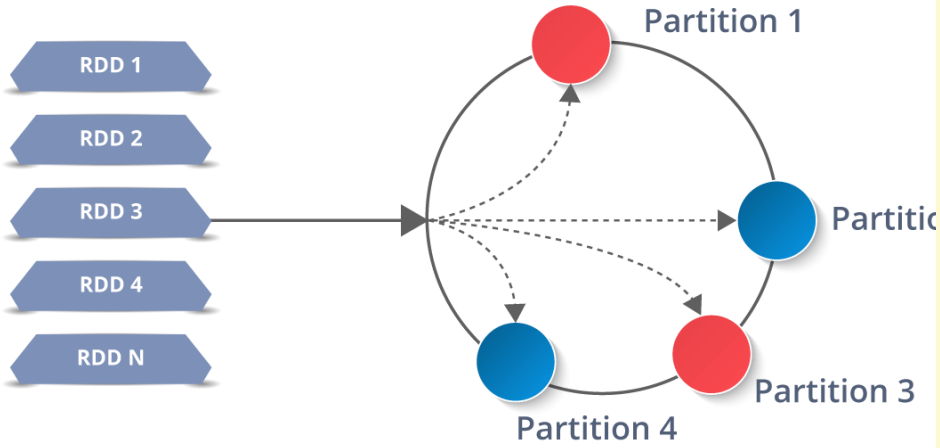
- Phoenix SQL layer on top of HBase
  - structured schema of the tables instead of schemaless freeride
  - mapping of columns to HBase cells
  - serialization of data types to bytes
  - SQL planner and optimizer:
    - Takes SQL query
    - Compiles it into a series of HBase scans
    - Direct use of the HBase API, along with coprocessors and custom filters
    - Produces regular JDBC result sets
  - built-in HBase related optimzations like salting, statistics etc
  - server-side (optimized) executions
  - access via JDBC

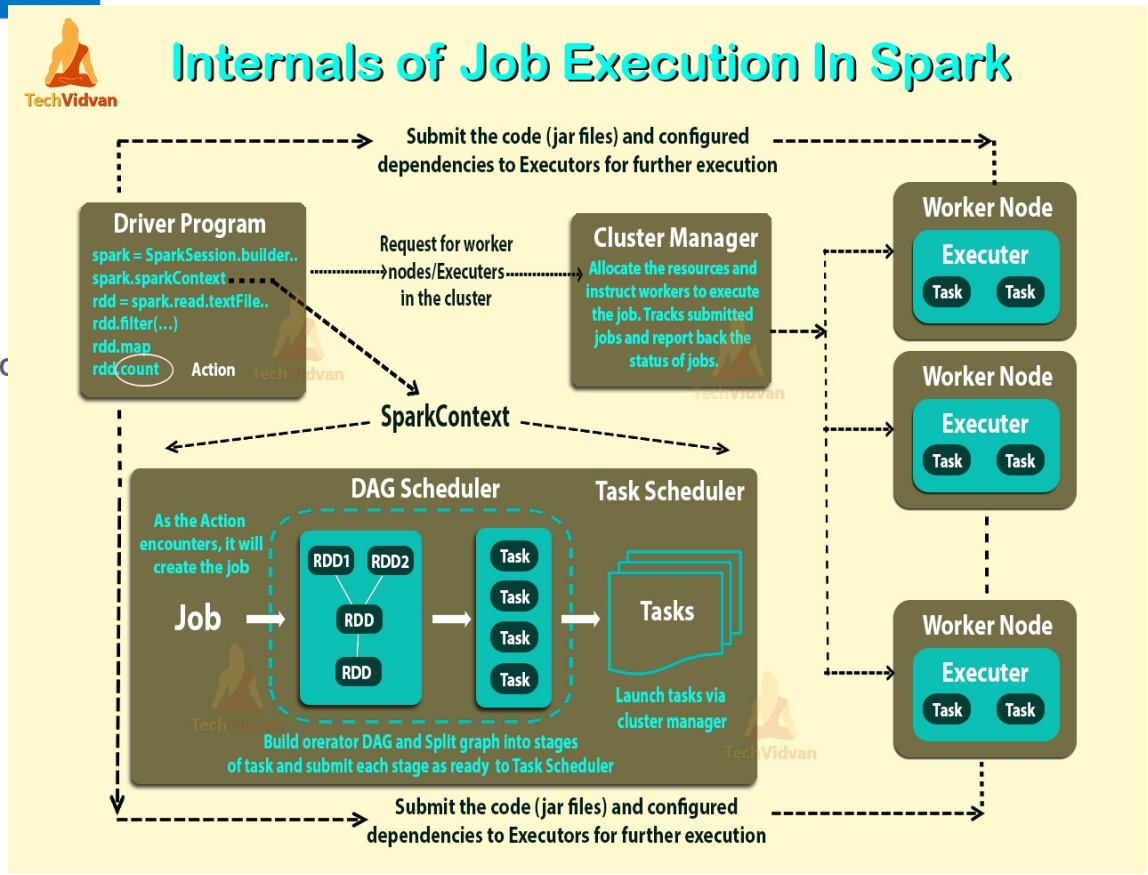




Processing framework: Spark

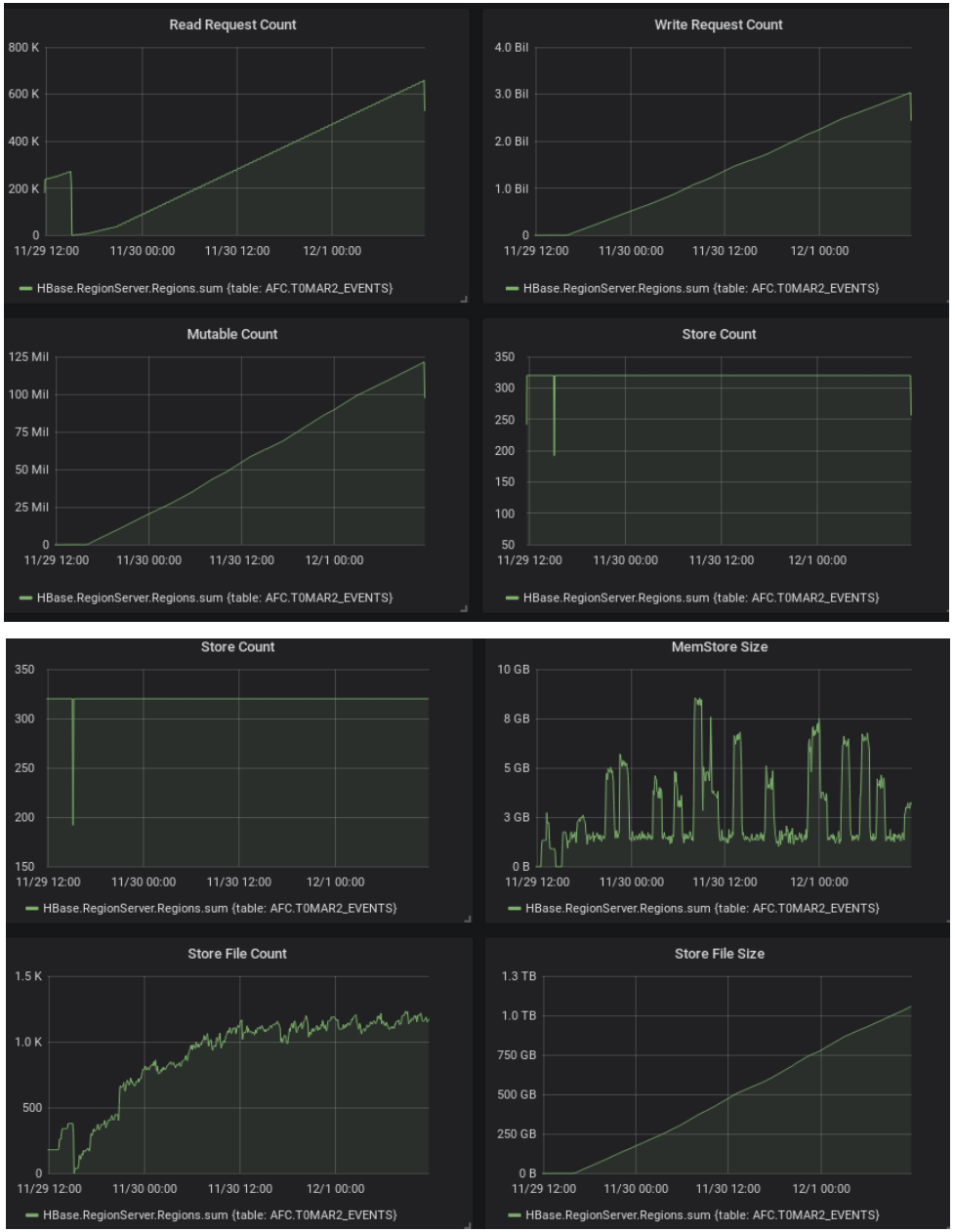






Devel and Monitoring: InfluxDB + Grafana

- **InfluxDB** useful to store metrics that change over time
  - Metrics contain a name, timestamp, tags and fields.
- **Grafana** is a web dashboard interface to plot and visualize data: several sources Influxdb, carbon/graphite, elastic, etc.
- **Applications:**
  - ATLAS Tier2 to monitor Lustre performance.
  - EI3 to monitor performance and identify possible problems:
    - Reads/Writes per client/Hbase region, Memory/File size, number of compactions
  - Other IFIC use cases
- **Tools for Softare development GitLab: igit.ific.uv.es**



Resumen

- IFIC colabora en el proyecto ATLAS Tier2 y en el EventIndex en varias areas que requieren el almacenamiento y tratamiento de grandes cantidades de datos.
- Se ha contribuido liderando la tarea de recolección de datos y más recientemente en el desarrollo del backend de almacenamiento (HBase/Phoenix) para el EI3.
- Se han presentado algunos desarrollos y herramientas con los que se ha ganado experiencia:
  - Infraestructura de almacenamiento: Object Store CEPH (e
  - Ecosistema BigData: Hadoop HDFS, Hbase, MapReduce framework, Apache Kudu, Apache Phoenix, Spark
  - Herramientas de monitorización: InfluxDB, Grafana
  - Desarrollo: GitLab
- Más información:
  - Álvaro Fernández, et al. "Distributed Data Collection for the Next Generation ATLAS E ventIndex Project." EPJ Web of Conferences. Vol. 214. EDP Sciences, 2019.
  - Dario Barberis et al, “The ATLAS EventIndex for LHC Run 3” ATL-COM-SOFT-2020-023 . To be published in CHEP2019 proceedings