CERN

Applying scalable databases at CERN

8th XLDB Conference

May 20, Stanford, CA, USA

Kacper Surdy

Nature of large databases at CERN

Typically store time series data, examples:

- Logging systems
  - LHC log data: 50kHz archiving, 200 TB + 90 TB/year
- Control and data acquisition systems (SCADA)
  - LHC detector controls
  - Quench Protection System: 150kHz archiving, 2TB/day
- Grid monitoring and dashboards

Example of LHC log numerical data

```
 VARIABLE_ID UTC_STAMP                                   VALUE
------------ --------------------------------- --------------
      906198 05-AUG-12 12.00.00.000000000 AM           -.0077
      907888 05-AUG-12 12.00.00.000000000 AM     -.0000140249
      907936 05-AUG-12 12.00.00.000000000 AM     -.0000087134
      907962 05-AUG-12 12.00.00.000000000 AM     -.0000042486
      340739 05-AUG-12 12.00.00.000000000 AM              2.2
      340737 05-AUG-12 12.00.00.000000000 AM             93.8
      895035 05-AUG-12 12.00.00.000000000 AM      -.000008338
     1193076 05-AUG-12 12.00.00.000000000 AM             15.7
     1214800 05-AUG-12 12.00.00.000000000 AM                1
     1214801 05-AUG-12 12.00.00.000000000 AM               34
     1214802 05-AUG-12 12.00.00.000000000 AM                1
```

The problem we want to tackle

RDBMS provides us

- fast pinpoint data extraction with indexes
- good writing frequencies (up to 1MHz)

What is missing?

- ad-hoc analytics, e.g. signal correlations and aggregations
  - requires sequential scanning of the data sets
- throughput limited to 1 GB/s
  - on currently deployed shared storage RDBMS clusters

Technologies chosen for evaluation

Hadoop

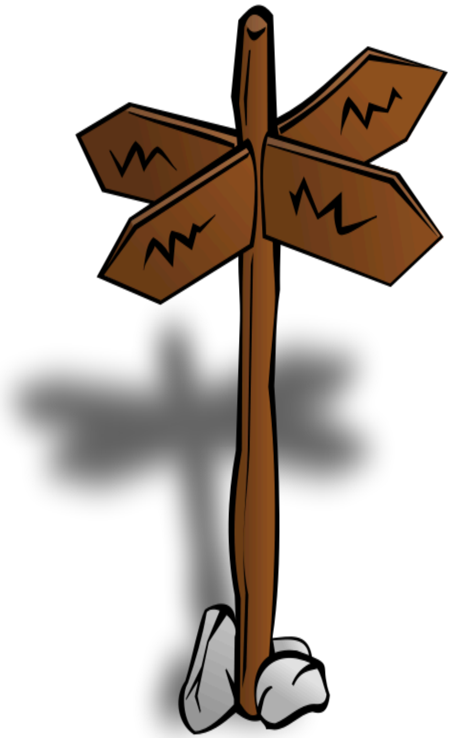
- open scalable architecture
- multiple analytic tools available
- widely used

Impala

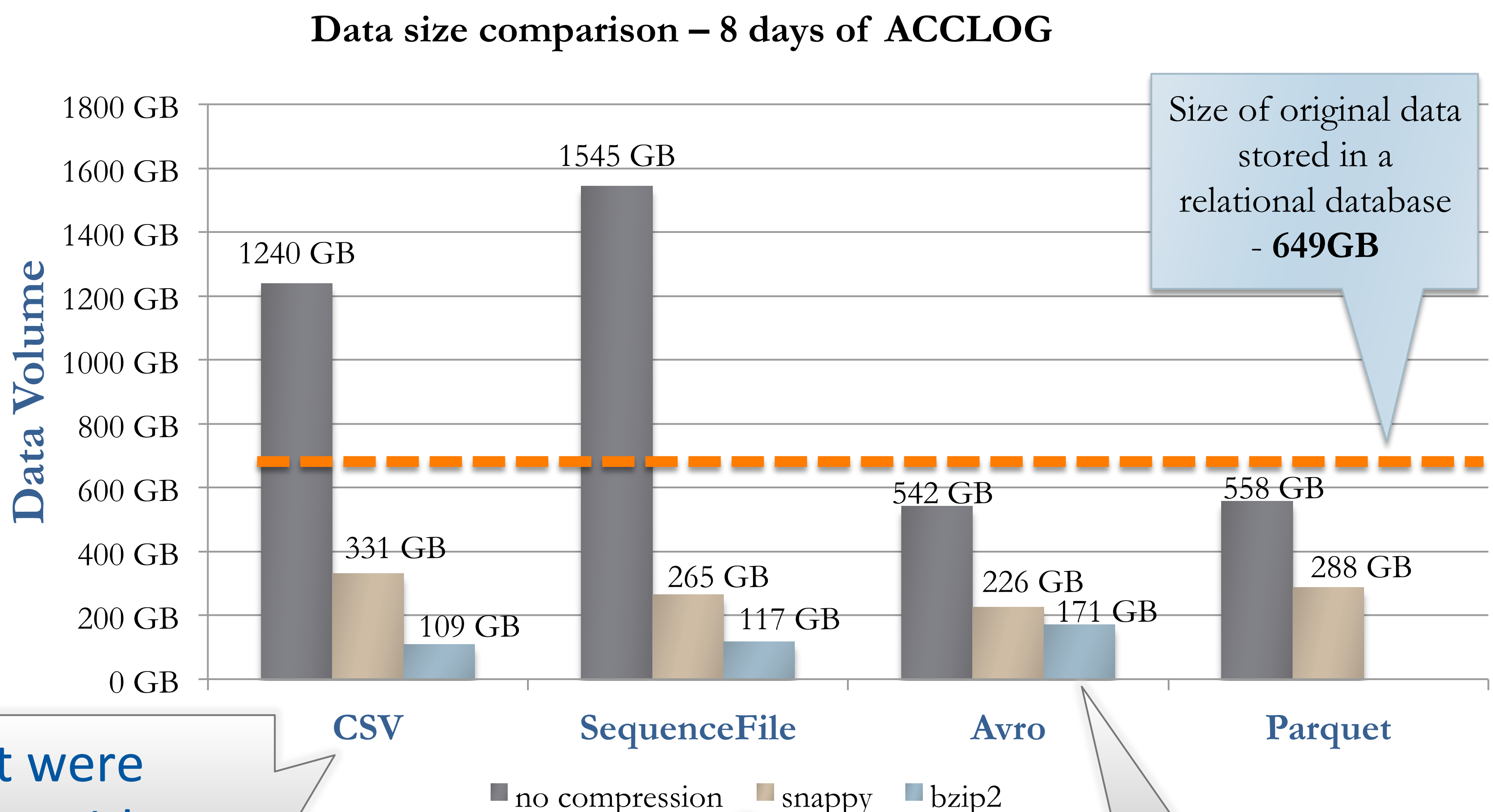
- SQL interface, ODBC/JDBC API available
- variety of supported file formats
- non MapReduce based

Different aspects of storing data

- Encoding: text vs binary
- Compression
- Partitioning
  - Vertical
  - Horizontal



Data format and compression is the key

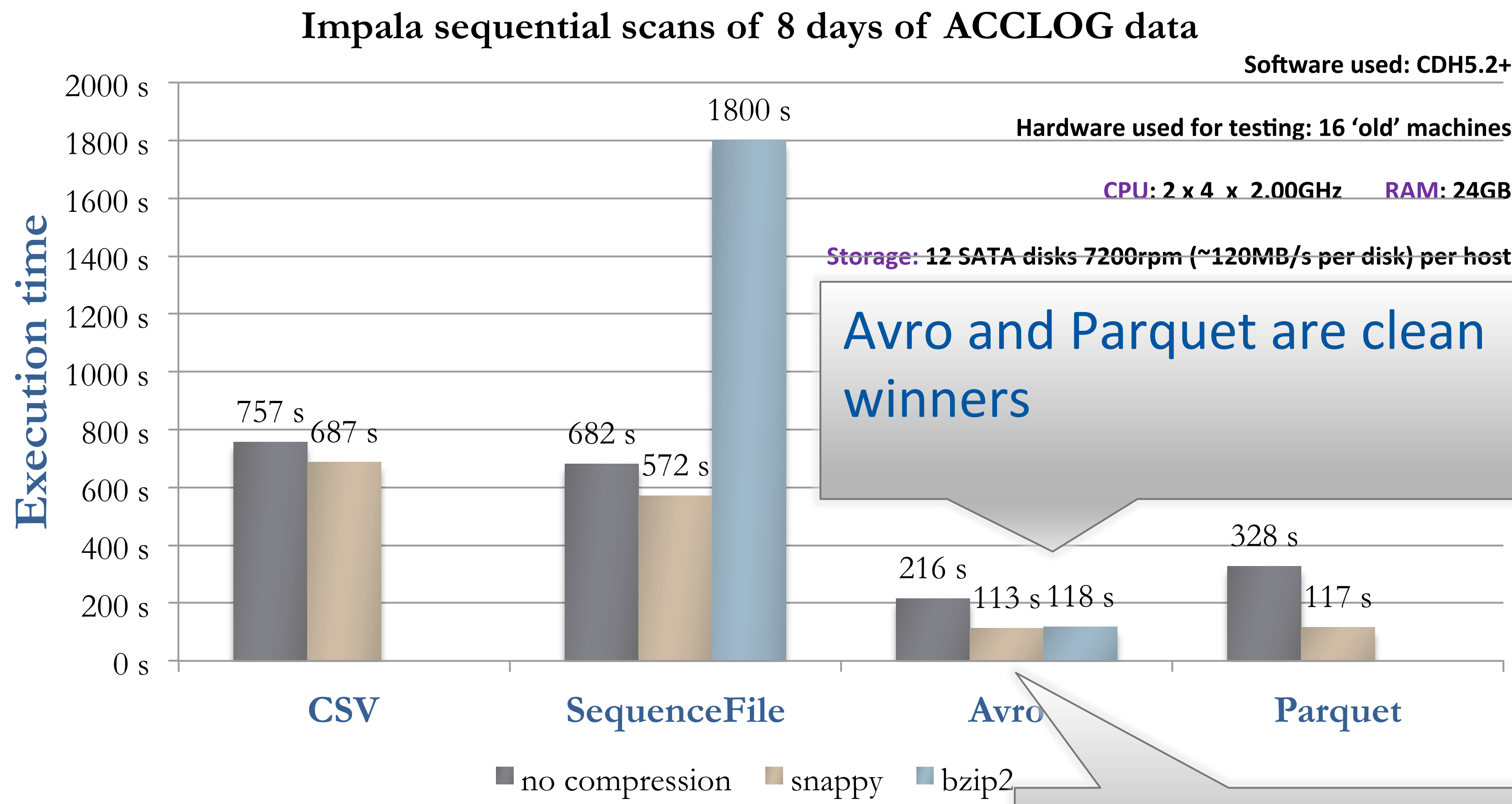


Test were done with several data file formats

2 compression types have been tested

bzip2 -> smallest files

Data format and compression is the key



Avro and Parquet are clean winners

snappy -> shortest execution times

Improving signal retrieval time

Problem: no indexes in Impala – full partition scan needed

- With daily partitioning we have 40 GB to read

Basic solution: fine-grain partitioning

- (year, month, day, **signal id**)

Concern: number of HDFS objects

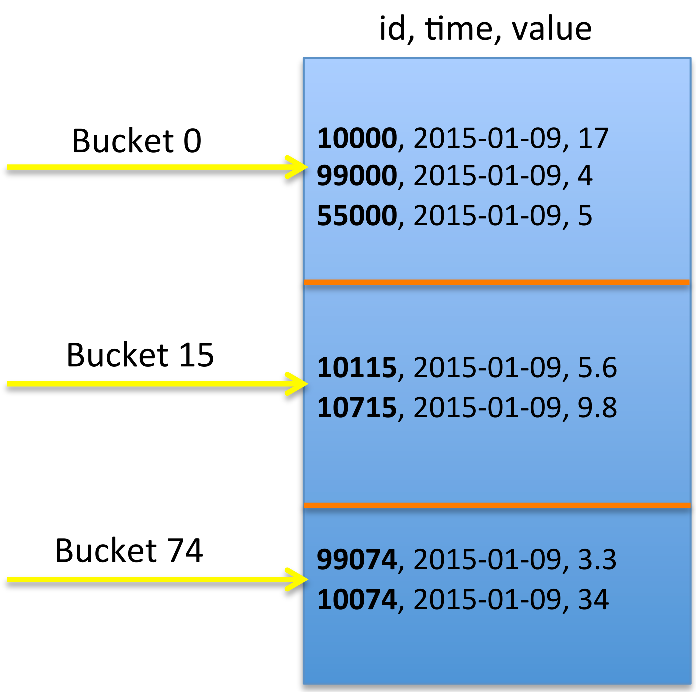
- 365 days * 1M signals = 365M of files per year
- file size: 41KB only!

Solution: multiple signals data grouped in a single partition

Bucketing: proof of concept

Based on mod(signal_id, x) function

- where x is tunable number of partitions created per day
- (year, month, day, **mod(signal id, x)**)



Bucketing: proof of concept

Based on mod(signal_id, x) function

- where x is tunable number of partitions created per day
- (year, month, day, **mod(signal id, x)**)

And it works!

- 10 partitions per day = 4GB to read
- Data retrieval time was reduced 10 times
  (from 15s to less than 2s)

We have modified the Impala planner code to make the function based partition pruning implicitly

- No need of explicit specification of a grouping function in 'where' clause

Conclusions

- Hadoop opens new use cases
  - many interfaces to the data
  - scalable
- Impala is good solution for our time series DBs
- choice of data format and compression type is critical
- customizations can be applied to improve performance

www.cern.ch