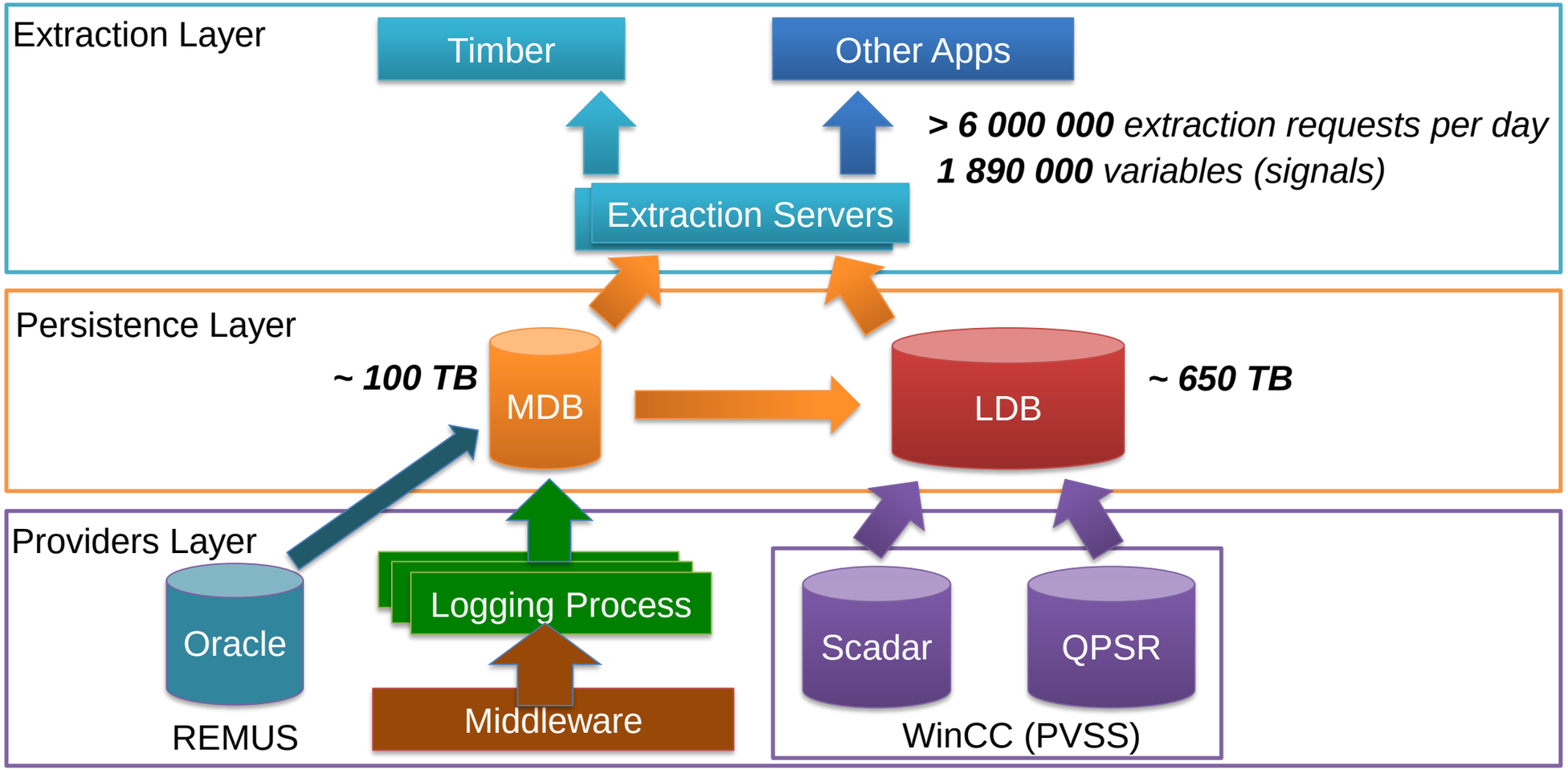
NXCALS Data Extraction and Analysis with Apache Spark

Nikolay Tsvetkov

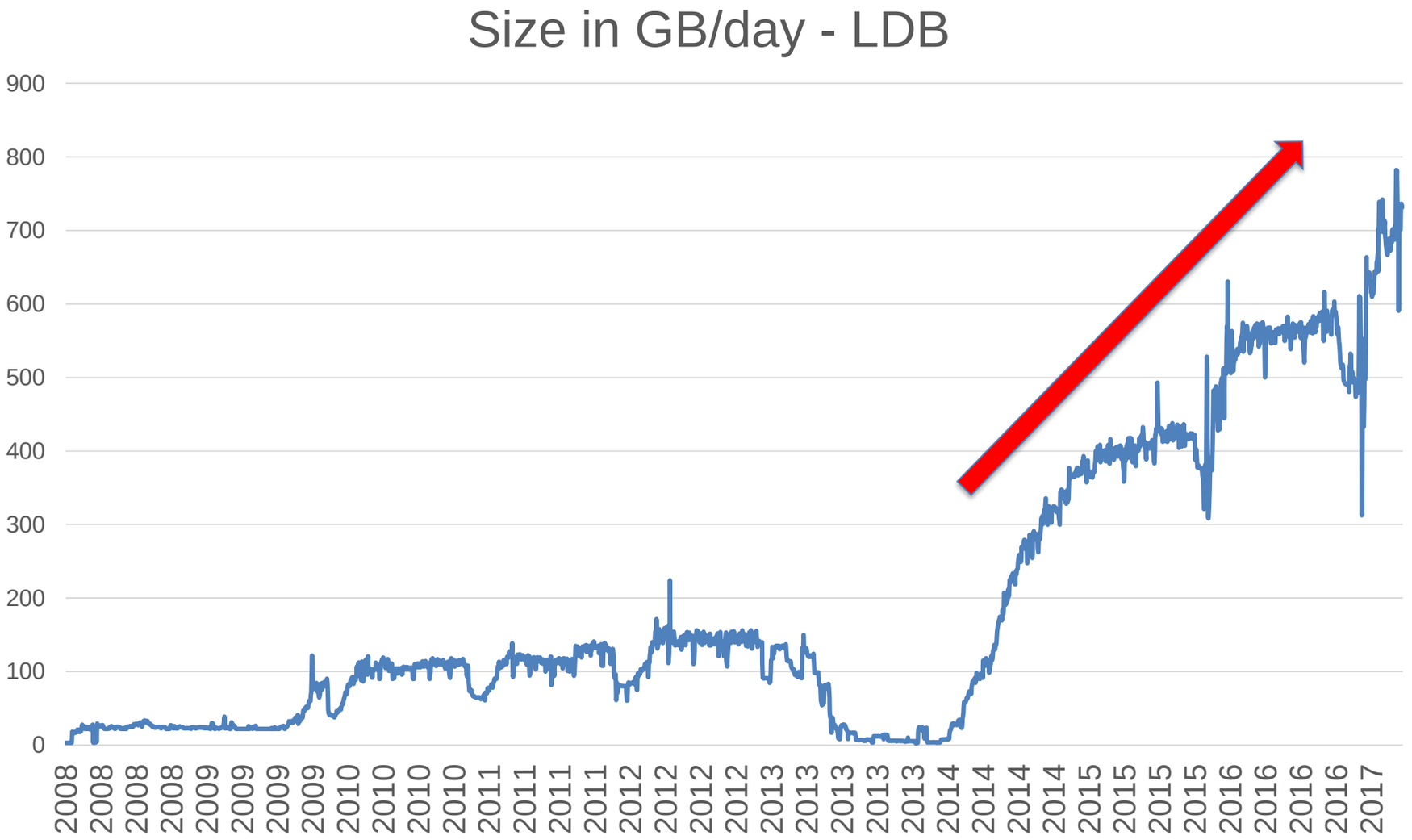
BE-CO-DS

Co-authors:
Jakub Wozniak
Marcin Sobieszek
Chris Roderick

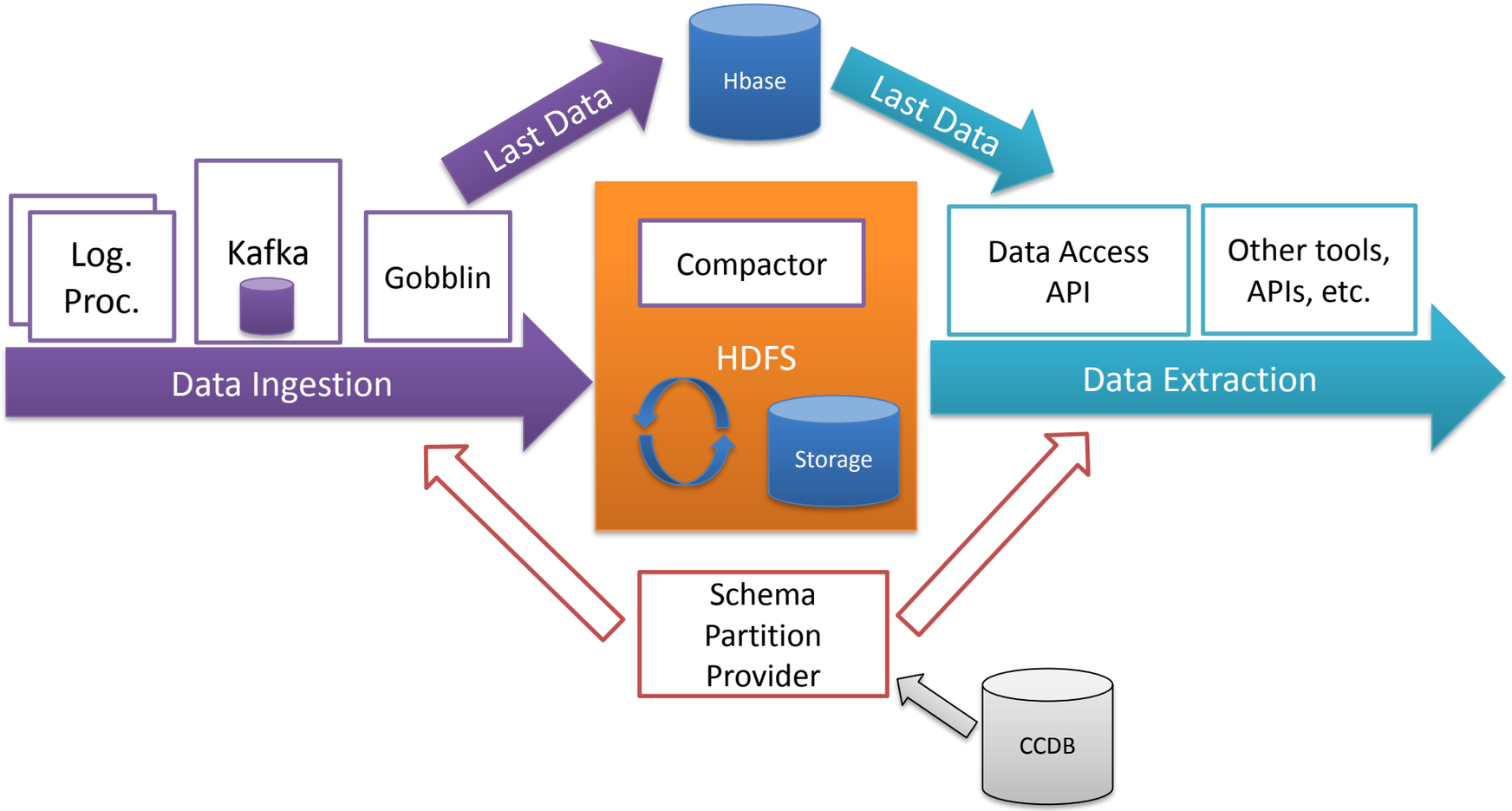
CALS Architecture



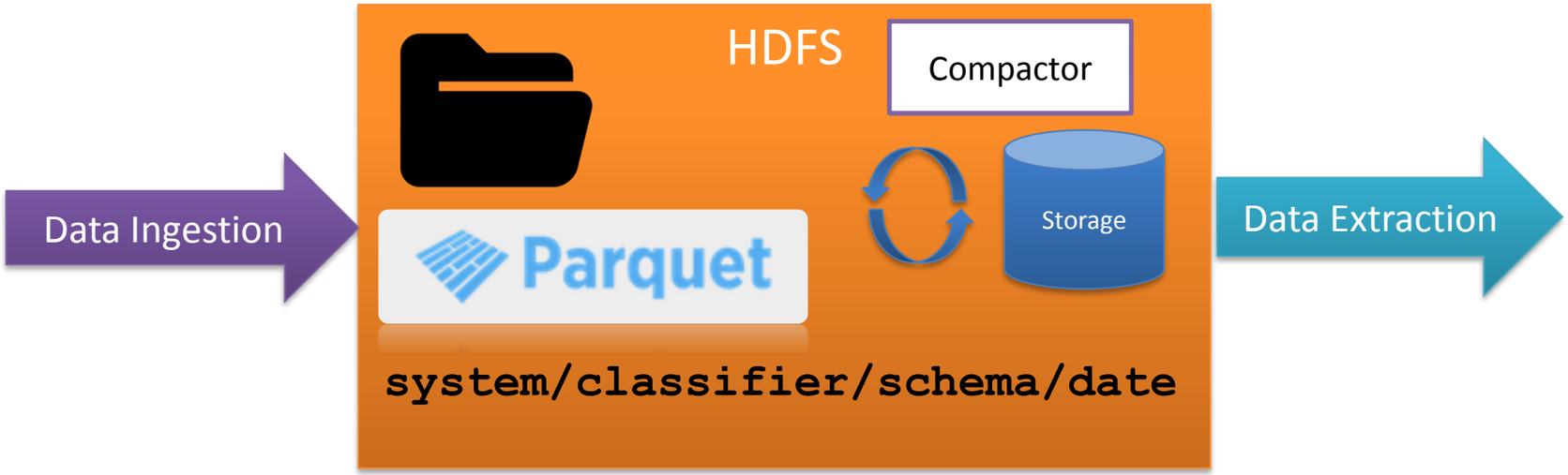
Storage Evolution



NXCALS Architecture

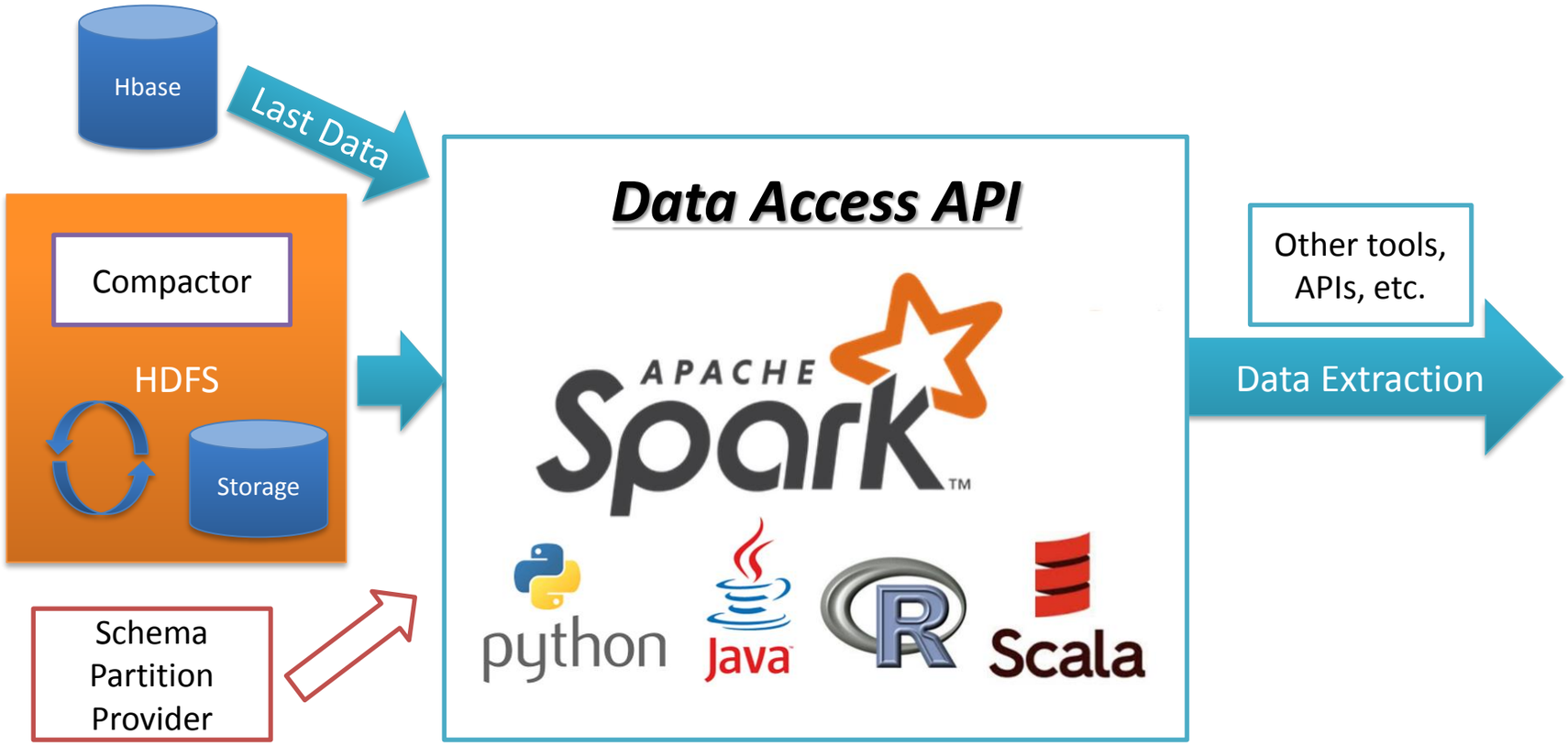


NXCALS Data Partitioning



- Data stored in **Parquet** file format of **records {f1,f2,…,fn}**
  - Accepts dynamic records of **ANY** content
  - Represents a change of "**state**" in **time** for some "**entities**"
  - Schema per entity **CAN** change over time

NXCALS Data Access API



DATA PROCESSING FRAMEWORK

GENERAL PURPOSE

Batch processing

Stream processing

Machine learning

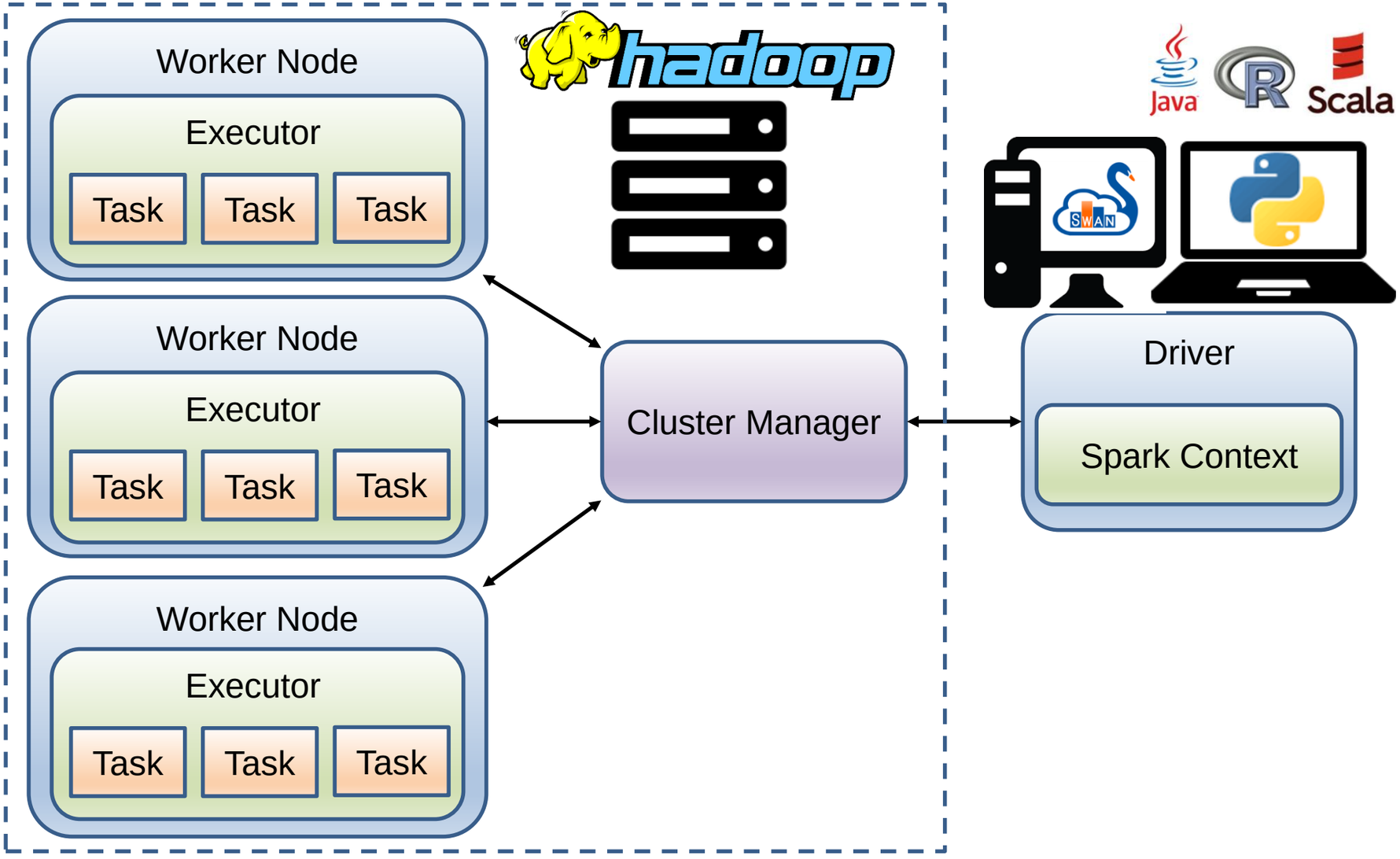
EASY TO USE */comparing to MapReduce/*

IN-MEMORY and fast */10-100x faster than MapReduce/*

SCALABLE *- increase the processing capacity by extending the cluster*

FAULT TOLERANT *– automatically handles failure of node in the cluster*

Spark High-level Architecture



NXCALS PySpark example

```
>> query = DevicePropertyQueryBuilder().system("CMW") \
        .device("BLM1").property("Acquisition") \
        .start_time("2017-05-22 00:00:00.000") \
        .end_time("2017-05-23 13:30:00.000").build_query()

>> df = spark.read.options(**query.get_map()) \
        .format("cern.accsoft.nxcals.data.access.api").load()

>> df2 = df.select("acqStamp", "field1").where("field2 = 123")
>> df2.show()
+-------------------+---------+
|     acqStamp      |  field1 |
+-------------------+---------+
|1495760359000000000|   101.10|
|1495760370000000000|     23.4|
+-------------------+---------+
```

Spark DataFrame & SQL APIs

```
# Spark SQL example
>> df.createOrReplaceTempView("temp_table")
>> df2 = spark.sql(
        "SELECT t.boolField, avg(t.doubleField) avg_double, max(t.intField) max_int " \
        "FROM temp_table t " \
        "WHERE t.acqStamp between 1495760359000000000 and 1495760370000000000“ \
        "GROUP BY t.boolField").collect()

# Spark DataFrame API
>> df2 = df.filter(df.acqStamp.between(1495760359000000000L,1495760370000000000L))\
       .groupBy("boolField").agg(func.avg("doubleField").alias('avg_double'),\
        func.max("intField").alias('max_int'))
>> df2.show()
```

```
+---------+------------------+----------+
|boolField|        avg_double|   max_int|
+---------+------------------+----------+
|     true|0.3678530290220542|2116819985|
|    false|0.6552150529696672|1608088701|
+---------+------------------+----------+
```

PySpark SQL JOIN example

```
>> query_bpm = DevicePropertyQueryBuilder().system("CMW") \
        .device("BPM1").property("Acquisition").start_time("2017-05-22 00:00:00.000") \
        .end_time("2017-05-23 13:30:00.000")
        .fields("cycleStamp", "floatField", "acqStamp", "doubleField").build_query()
>> df_bpm = spark.read.options(**query_bpm.get_map()) \
        .format("cern.accsoft.nxcals.data.access.api").load()

>> df_bpm.createOrReplaceTempView("bpm_temp_table")
>> df2 = spark.sql(
        "SELECT t1.boolField result, count(t1.boolField) cnt " \
        "FROM temp_table t1 JOIN bpm_temp_table t2 ON t2.cycleStamp = t1.cycleStamp " \
        "WHERE t2.floatField < 0.1 " \
        "GROUP BY t1.boolField")
>> df2.show()
+---------+--------------+
|  result |num_operations|
+---------+--------------+
|     true|         16379|
|    false|            22|
+---------+--------------+
```

From Spark to Pandas data analysis

```
>> df2 = spark.sql(
        "SELECT t.acqStamp, max(t.doubleField) max_value" \
        "FROM bpm_temp_table t " \
        "GROUP BY t.acqStamp")
>> df2.show()
```

```
+-------------------+---------+
|     acqStamp      |max_value|
+-------------------+---------+
|1495760359000000000|    0.093|
|1495760370000000000|    0.095|
|...                |      ...|
+-------------------+---------+
```

```
# Collect to Pandas:
>>> pandas = df2.toPandas()
>>> pandas["max_value"].plot()
```

NXCALS User Guide: https://wikis.cern.ch/display/CALS/NXCALS+-+Data+Access+User+Guide

SWAN-Spark_NXCALS_Example (autosaved)

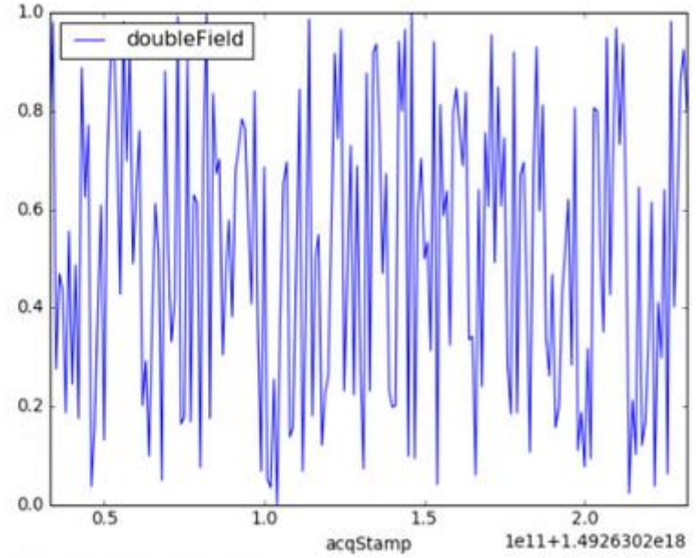
Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 2

Markdown | CellToolbar

Integration of SWAN with Spark clusters

In [4]:
```
df2.toPandas().plot(x="acqStamp")
```



NXCALS State?

Official dates : ***Ready by LS2 (Q1 2019)***

System development solid beta for **Q1 2018**

- ✓ Data Ingestion Java API ready
  - We store data from CMW devices
- ✓ Data Access API with Spark first version ready
  - **Any Spark analysis** on top of Spark **DataSets**
- ✓ **Python binding & SWAN** integration

- ❑ What comes next (for data extraction and analysis):
  - ❑ Clients integration (**business use-cases**)

Questions ?

E-mail: n.tsvetkov@cern.ch