NXCALS @Hadoop User Forum #1

Jakub Wozniak, BE-CSS-CPA (on behalf of the Logging team)

04/12/2023

Agenda

- What is NXCALS?
- NXCALS Data in HBase & HDFS
- Ideas / Conclusions

What is NXCALS?

NXCALS

- **NeXt CALS** (**No** e**X**cel **CALS…** 😉)
  - **Successor of CALS (**Oracle DB based Logging System from 2001**)**
- **Stores data (readings/settings)** from **accelerator complex devices**
- Data used for **online monitoring & offline analysis**
  - by variety of users from machine operators to beam physicists
- Helps improving accelerator **performance**
- **Decision support** system for management
- Avoids **duplicate logging** efforts

BE-CSS Logging Team



- **Key** -> **Value** store (with **Timeseries**)
- **Apache Spark** as **Extraction API in Python & Java**
- Needs **PB-size storage** → **relying on IT offerings**

NXCALS of Today

Data Sources

In production since 2021

TIM & REMUS Datasources

In production since April 2020

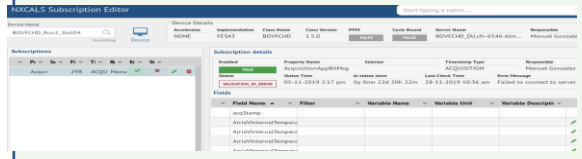
WinCCOA Datasources

In production since 2018

CMW Datasources

Settings, PM, etc

Configuration via CCDE



Heterogenous Inputs

Data Storage

Extensive User Documentation
http://nxcals-docs.web.cern.ch

New data

Core / Storage

Old data

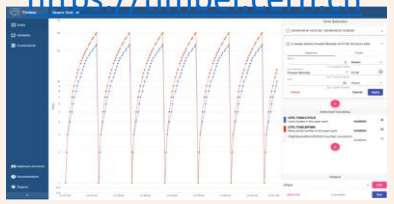
CALS Data Migration

Done (details in CCDE)

Spark API

NXCALS Data Access API

https://timber.cern.ch



X major releases

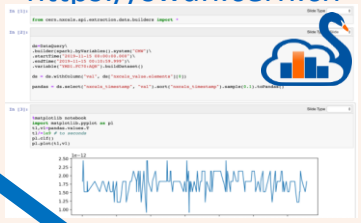
Spark Bundle "pip install nxcals"

"Thin" API

MetaData API

Data Extraction

https://swan.cern.ch



CERN Specific API

CALS "Backport" API

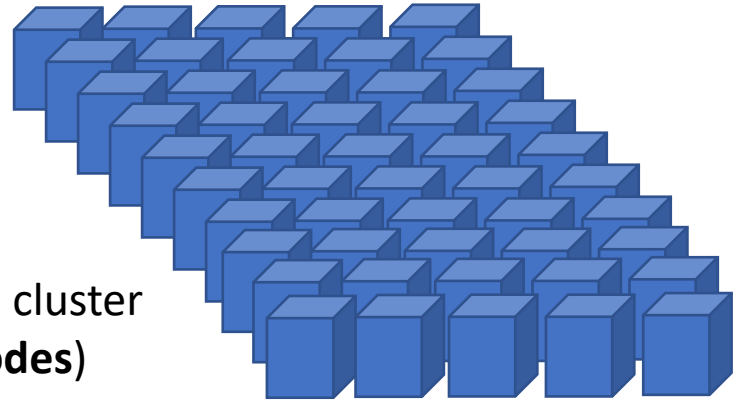
PyTimber (new)

Extensive Set of APIs & Tools

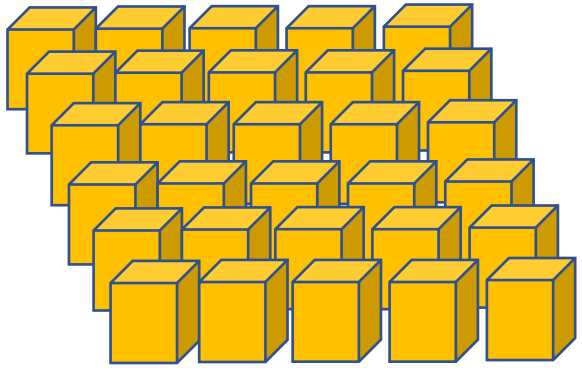
Infrastructure (PRO only)

IT BigData backend
(**22TB RAM, 10k VCores, 19PB HDD**)

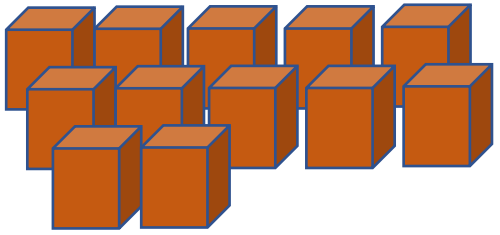
HDFS Cluster (**50 nodes**)



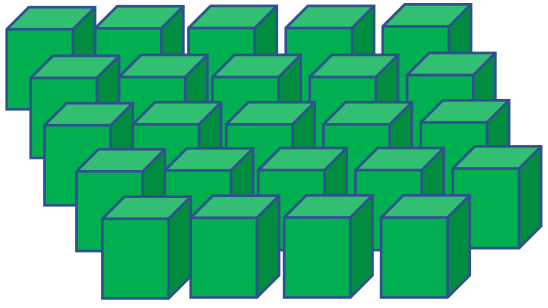
HBase cluster
(**30 nodes**)



NXCALS (**12 nodes**)

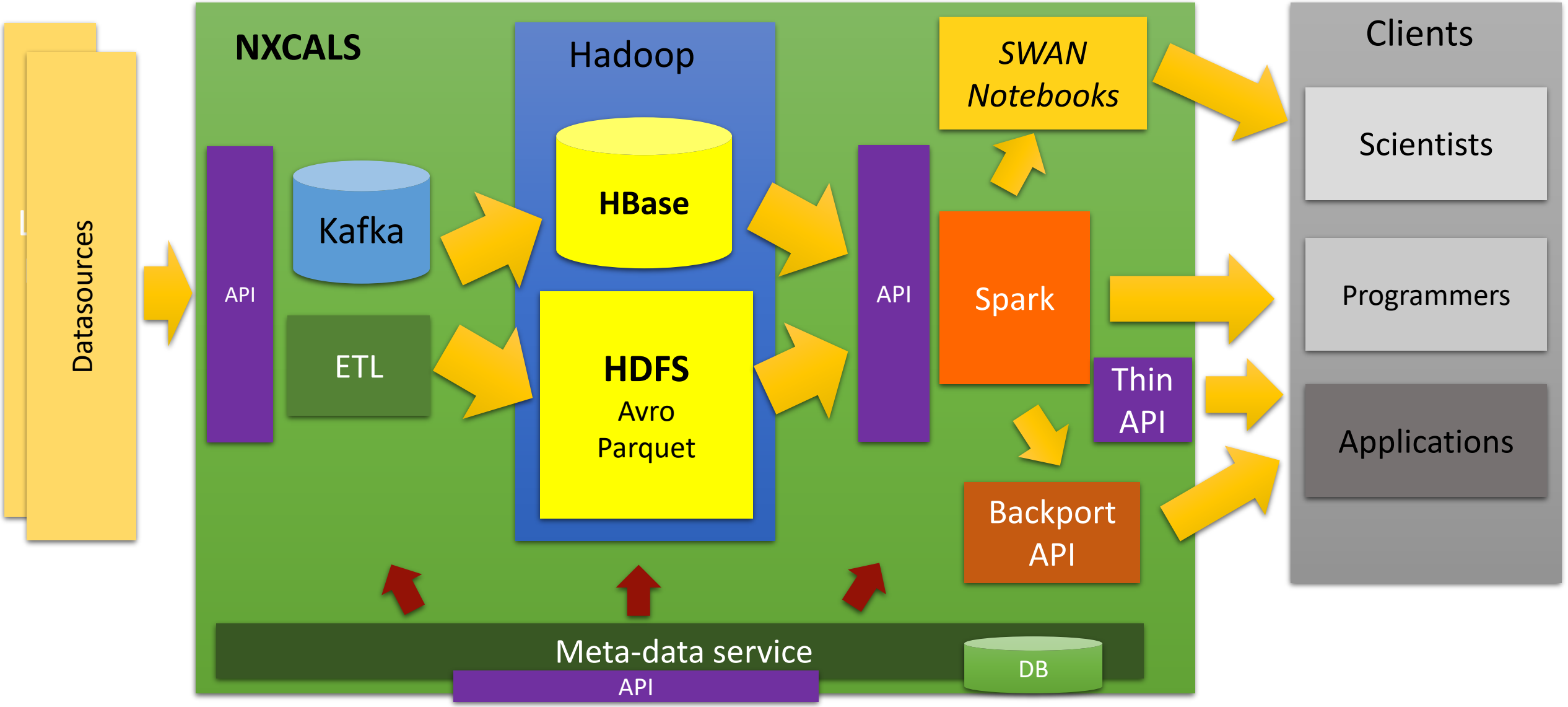


CMW+WINCCOA DSes
(**24 nodes**)



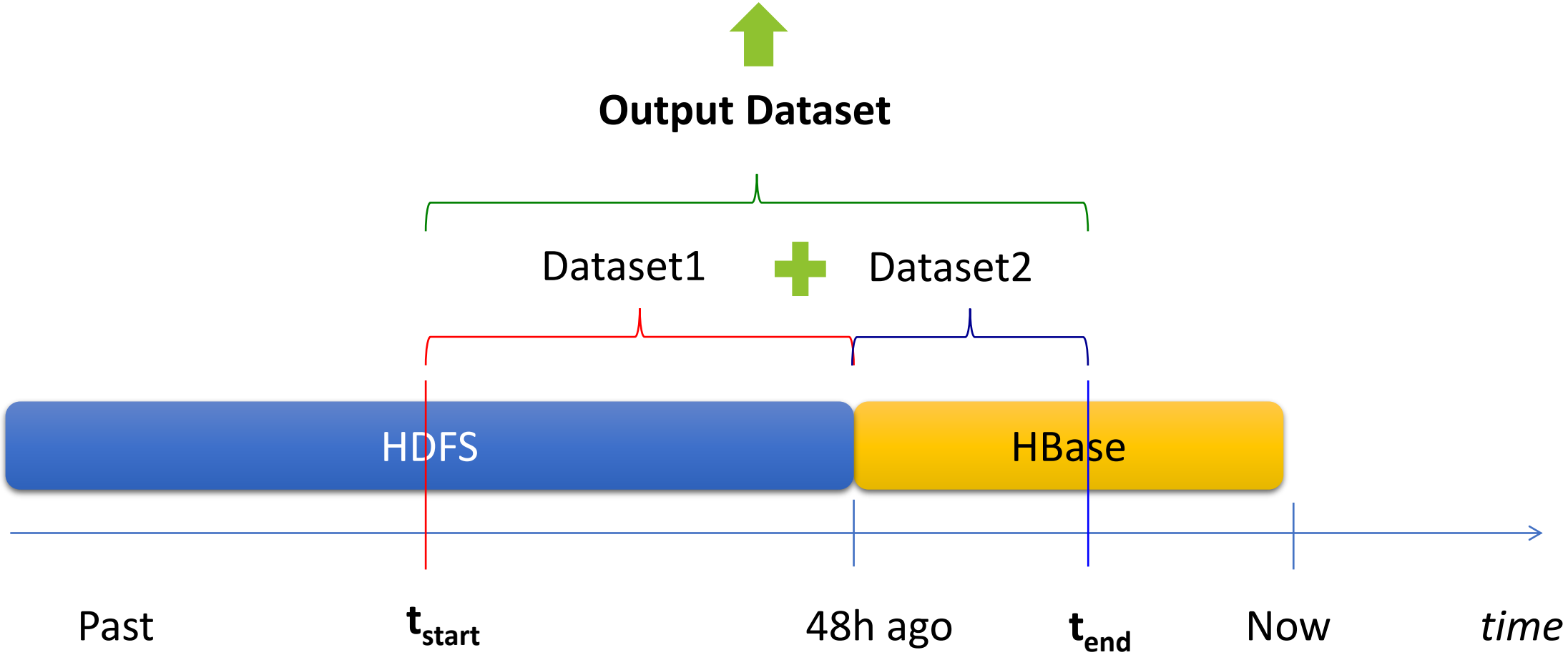
~1900
Users

NXCALS Architecture

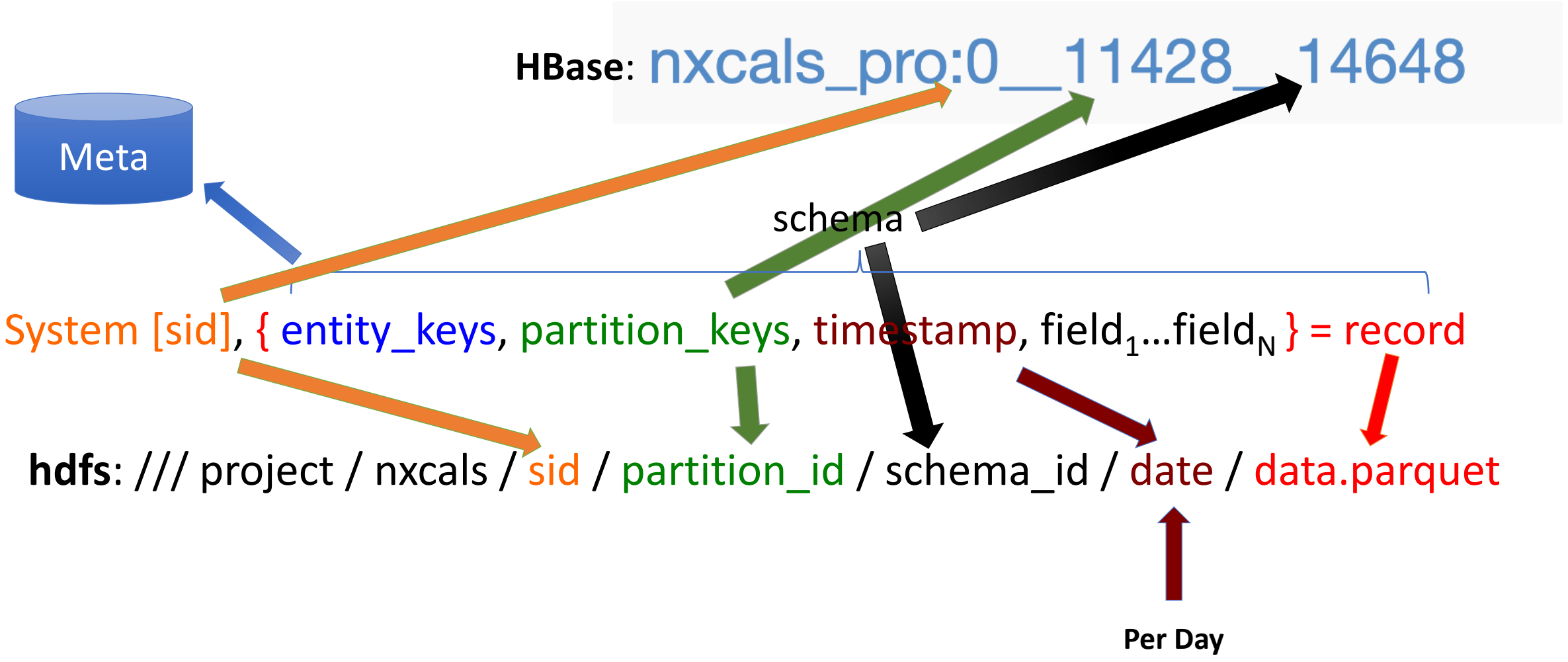


What about NXCALS Data?

Data Storage

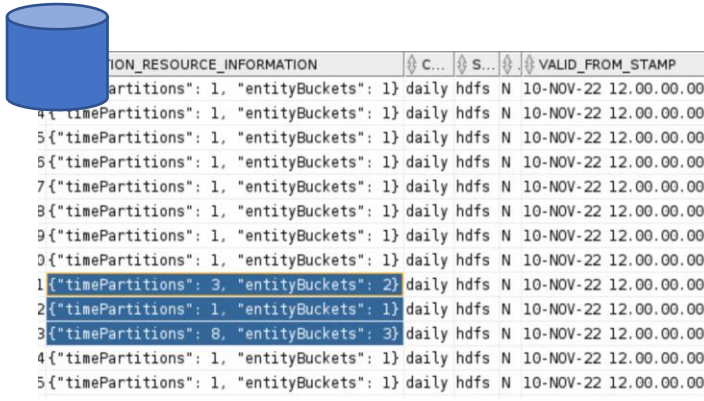


Data Partitioning



Extraction Performance Improvements

- More **fine-grained** physical **partitioning** of data **within one day (HDFS)**
- **Grouping data by:**
  - **time** (up to 96 partitions by day, depends on data size)
  - **entity id** (hashed bucketing)



	ION_RESOURCE_INFORMATION	C...	S...		VALID_FROM_STAMP
	artitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
4	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
5	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
6	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
7	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
8	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
9	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
0	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
1	{"timePartitions": 3, "entityBuckets": 2}	daily	hdfs	N	10-NOV-22 12.00.00.00
2	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
3	{"timePartitions": 8, "entityBuckets": 3}	daily	hdfs	N	10-NOV-22 12.00.00.00
4	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00
5	{"timePartitions": 1, "entityBuckets": 1}	daily	hdfs	N	10-NOV-22 12.00.00.00

Old partitions:

(...)/2022/9/10/00_03_H-part-00000-9018b8a9-5f6f-4462-93e6-8532ddc90fbb-c000.snappy.parquet
(...)/2022/9/10/00_03_H-part-00001-9018b8a9-5f6f-4462-93e6-8532ddc90fbb-c000.snappy.parquet
(...)/2022/9/10/03_06_H-part-00000-99bfa55a-9059-4dc8-afc5-a72327670ff3-c000.snappy.parquet
(...)/2022/9/10/03_06_H-part-00001-99bfa55a-9059-4dc8-afc5-a72327670ff3-c000.snappy.parquet
(...)/2022/9/10/06_08_H-part-00000-de9dc0ba-4a7f-4483-b1ba-cbb4062dab54-c000.snappy.parquet

New partitions:

(...)/2022/11/10/__sys_nxcals_**time_partition__=0**-__sys_nxcals_**entity_bucket__=0**-part-00000-3308343a-081a-4f7b-8dfb-56c1222391ea.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_time_partition__=0-__sys_nxcals_entity_bucket__=1-part-00000-4ae00dea-9c3e-4409-91d8-4476f61bf727.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_time_partition__=0-__sys_nxcals_entity_bucket__=2-part-00001-4ae00dea-9c3e-4409-91d8-4476f61bf727.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_time_partition__=1-__sys_nxcals_entity_bucket__=0-part-00000-bf7c511e-2009-4523-a041-6ce1fcb39f03.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_**time_partition__=1**-__sys_nxcals_**entity_bucket__=1**-part-00001-bf7c511e-2009-4523-a041-6ce1fcb39f03.c000.snappy.parquet

Extraction Performance Improvements

- **Extraction** done by **time window** & **entity id**
  - ***Time window* -> *Time partition number***
  - ***Entity id* -> *Entity bucket number***

Old-style query (**all files scanned**):

(...)/2022/9/10/00_03_H-part-00000-9018b8a9-5f6f-4462-93e6-8532ddc90fbb-c000.snappy.parquet
(...)/2022/9/10/00_03_H-part-00001-9018b8a9-5f6f-4462-93e6-8532ddc90fbb-c000.snappy.parquet
(...)/2022/9/10/03_06_H-part-00000-99bfa55a-9059-4dc8-afc5-a72327670ff3-c000.snappy.parquet
(...)/2022/9/10/03_06_H-part-00001-99bfa55a-9059-4dc8-afc5-a72327670ff3-c000.snappy.parquet
(...)/2022/9/10/06_08_H-part-00000-de9dc0ba-4a7f-4483-b1ba-cbb4062dab54-c000.snappy.parquet

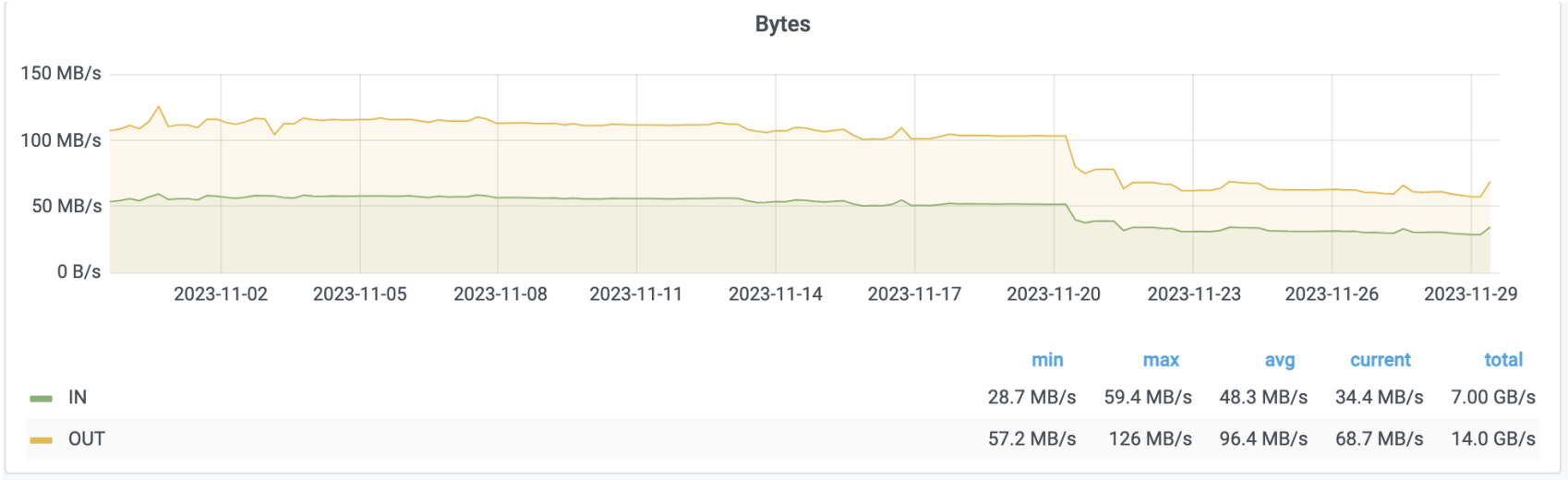
New-style query using **time partition** & **entity bucket** (**selected files scanned**):

(...)/2022/11/10/__sys_nxcals_time_partition__=0-__sys_nxcals_entity_bucket__=0-part-00000-3308343a-081a-4f7b-8dfb-56c1222391ea.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_time_partition__=0-__sys_nxcals_entity_bucket__=1-part-00000-4ae00dea-9c3e-4409-91d8-4476f61bf727.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_time_partition__=0-__sys_nxcals_entity_bucket__=2-part-00001-4ae00dea-9c3e-4409-91d8-4476f61bf727.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_time_partition__=1-__sys_nxcals_entity_bucket__=0-part-00000-bf7c511e-2009-4523-a041-6ce1fcb39f03.c000.snappy.parquet
(...)/2022/11/10/__sys_nxcals_time_partition__=1-__sys_nxcals_entity_bucket__=1-part-00001-bf7c511e-2009-4523-a041-6ce1fcb39f03.c000.snappy.parquet

Results in much less data scans -> **faster extraction**

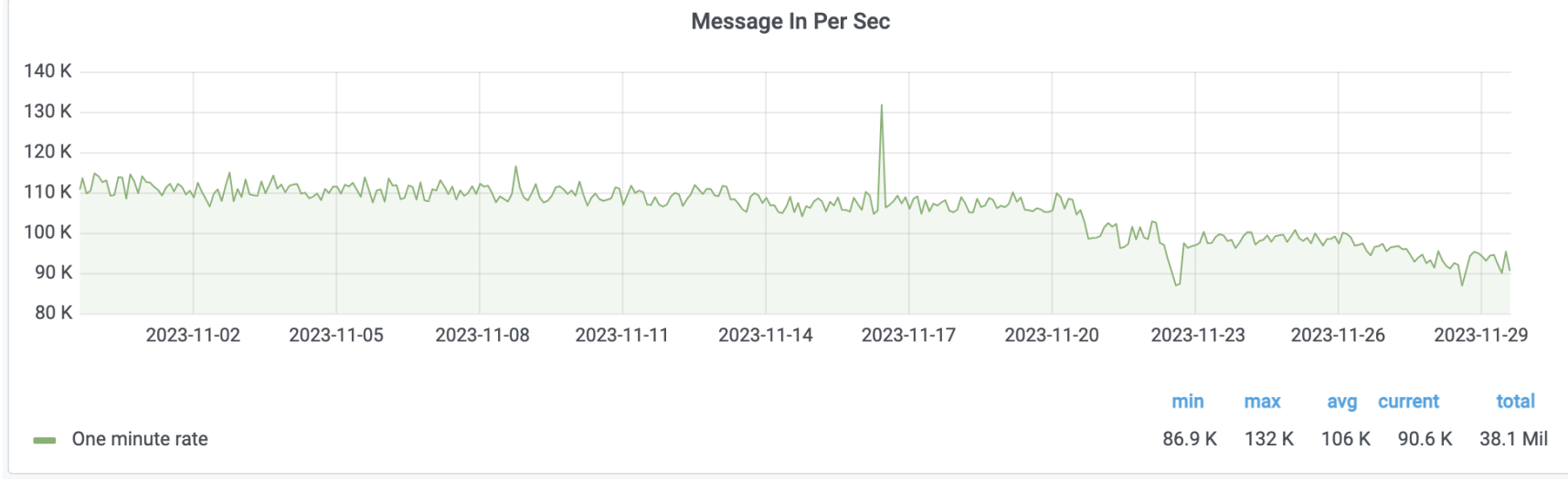
Data Volumes

Current throughput:
~50MB/sec (as seen in Kafka)



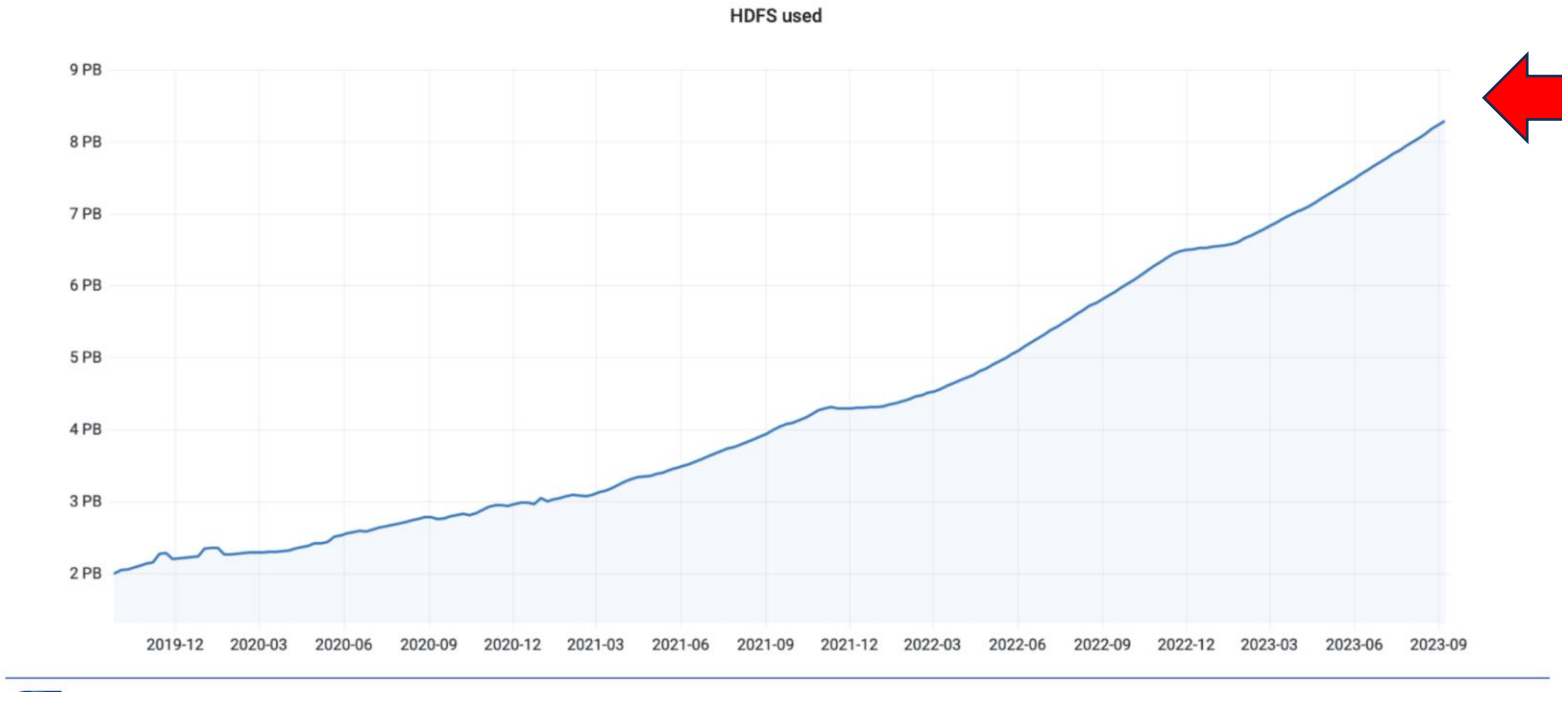
~110k rec/sec

While migrating old data:
~1.5-2.0M rec/sec



Data Volumes

Projected Data Growth



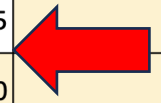
8.5 PB
(with 3x replicas)

Courtesy: Pedro Andrade (IT)

Data Volumes

Projected Data Growth

LHC	Year	PB in January	PB increase
LS2	2020	2.34	0.70
	2021	3.04	1.27
RUN3	2022	4.31	2.24
	2023	6.55	2.25
	2024	8.80	2.25
	2025	11.05	2.25
LS3	2026	13.30	1.50
	2027	14.80	1.50
	2028	16.30	1.50



Courtesy: Pedro Andrade (IT)

SWAN Data Extraction Example

In [9]:
```
from nxcals.api.extraction.data.builders import DataQuery
ds = DataQuery.builder(spark).entities().system('CMW') \
    .keyValuesEq({'device': 'LHC.LUMISERVER', 'property': 'CrossingAngleIP1'}) \
    .timeWindow('2022-04-29 00:00:00.000', '2023-04-30 00:00:00.000').build()
ds.toPandas()
```

Out[9]:

	DeltaCrossingAngle	Moving	__record_timestamp__	__record_version__	acqStamp	class	cyclestamp	device	property	selector	nxcals_entity_id
0	-10.0	False	1668002376170000000	0	1668002376170000000	LhcLumiscan	0	LHC.LUMISERVER	CrossingAngleIP1	None	57336
1	-10.0	True	1668002376171000000	0	1668002376171000000	LhcLumiscan	0	LHC.LUMISERVER	CrossingAngleIP1	None	57336
2	0.0	False	1668002422195000000	0	1668002422195000000	LhcLumiscan	0	LHC.LUMISERVER	CrossingAngleIP1	None	57336
3	-10.0	True	1668002468937000000	0	1668002468937000000	LhcLumiscan	0	LHC.LUMISERVER	CrossingAngleIP1	None	57336
4	0.0	False	1668002507214000000	0	1668002507214000000	LhcLumiscan	0	LHC.LUMISERVER	CrossingAngleIP1	None	57336
...	...	...	...	...	...	...	...	...	...	...	...
129	50.0	False	1663910909905000000	0	1663910909905000000	LhcLumiscan	0	LHC.LUMISERVER	CrossingAngleIP1	None	57336
130	0.0	False	1663911025431000000								
131	45.0	False	1663911068157000000								
132	45.0	True	1663911068158000000								
133	0.0	False	1663911173323000000								

134 rows × 11 columns

In [10]:
```
ds = DataQuery.getForVariables(spark,
                               system='CMW',
                               start_time='2018-04-29 00:00:00.000',
                               end_time='2018-04-30 00:00:00.000',
                               variables=['LTB.BCT60:INTENSITY','LTB.BCT50:INTENSITY'])

ds.toPandas()
```

Out[10]:

	nxcals_value	nxcals_entity_id	nxcals_timestamp	nxcals_variable_name
0	2384.68	52015	1524960300065000000	LTB.BCT50:INTENSITY
1	2392.26	52015	1524960399665000000	LTB.BCT50:INTENSITY
2	2397.61	52015	1524960756065000000	LTB.BCT50:INTENSITY
3	2341.85	52015	1524961093265000000	LTB.BCT50:INTENSITY
4	2414.15	52015	1524961191665000000	LTB.BCT50:INTENSITY
...	...	...	...	...
140614	1611.46	52034	1525045270865000000	LTB.BCT60:INTENSITY
140615	2381.04	52034	1525045444865000000	LTB.BCT60:INTENSITY
140616	1268.95	52034	1525045671665000000	LTB.BCT60:INTENSITY
140617	2370.73	52034	1525045966865000000	LTB.BCT60:INTENSITY
140618	7495.09	52034	1525046018465000000	LTB.BCT60:INTENSITY

140619 rows × 4 columns

Our Hadoop Technology Experience

- Not an easy **start & steep learning curve** (due to **multitude of technologies**)
- **HDFS** file system -> **rock-solid** & very performant
- **HBase**
  - Some problems in the past, instabilities, surprises, failures, etc
  - Currently (since ~2 years) **very stable**
- **Yarn**
  - **Very solid**
  - **Somewhat cryptic** when it comes to debugging – users cannot easily understand why their apps fail
- **Overall mature & stable technology**

Our Hadoop Service Experience

- **Very good contact** & always fruitful collaboration **with the IT teams**
- Vivid **Mattermost channel**
- Monthly status **meetings**
- **Exchange of information** about planned interventions / outages / etc
- Controllable (careful) **change management**
  - dev → test → stage → perftest → testbed → pro

Big Thanks to IT for all the efforts!

Future Ideas (short term)

- Contenerisaiton (**K8s**) focus in 2024 (on NXCALS side), CSS-IT collaboration for TN K8s
- **Size** will be **a problem** (eventually…)
  - Testing **Erasure Encoding** to avoid 3x replicas
  - **Federation** to overcome **namenode memory limitations** -> limited number of objects
  - **Data growth** in the view of **HL-LHC** (**is physical space a limitation**?)
  - **Cold storage** for “old” data?
  - Storage of **very big records >50MB** (sometimes ~GB)
    - Current solution has 50MB limitation due to Kafka, Hbase, etc.
- **Testing the limits** of the current instalation in terms of throughput
- Users want to **store data directly on HDFS** (files, results of analysis)
  - Currently this is not standardised, space is given case-by-case
  - How to share among users?

Future Ideas (long term)

- Is **HBase** a solution (or a future problem)?
- Testing other architectures/solutions
  - **Delta Lake** from Databricks ?
  - **Presto** for extraction ?
  - **NewSQL** Databases (Google Spanner, VoltDB, YugabyteDB, …) – can this be safely ignored for now?
- **Cloud-based** solutions (AWS, Azure, etc)
- Surveying **emerging trends**, new products, etc

- **ATS-IT NXCALS review in preparation**

Thanks for your attention!

Please contact us on:

acc-logging-support@cern.ch

https://mattermost.web.cern.ch/nxcals/channels/nxcals-community