Storage statistics through Hadoop ecosystem

HEPiX Autumn 2019 at Nikhef, Amsterdam,The Netherlands

Author: Antoine DUBOIS

Co-Author: Osman AIDEL

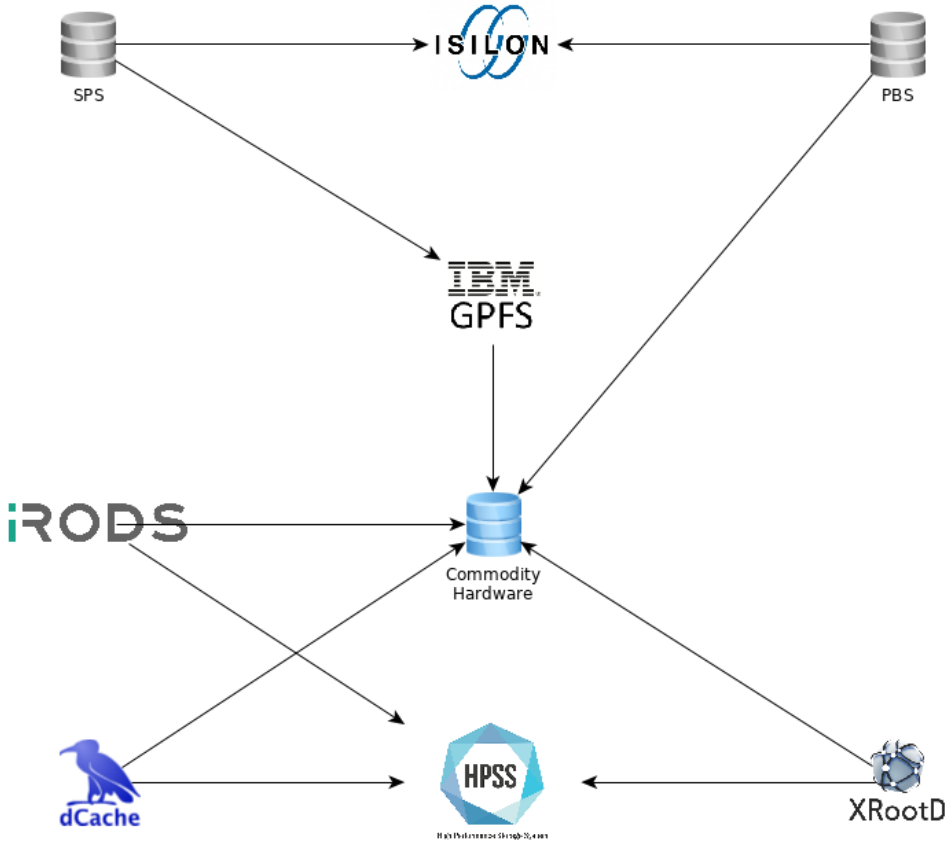
Summary

- Context
- Storage statistics project
- Conclusion

Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

Context

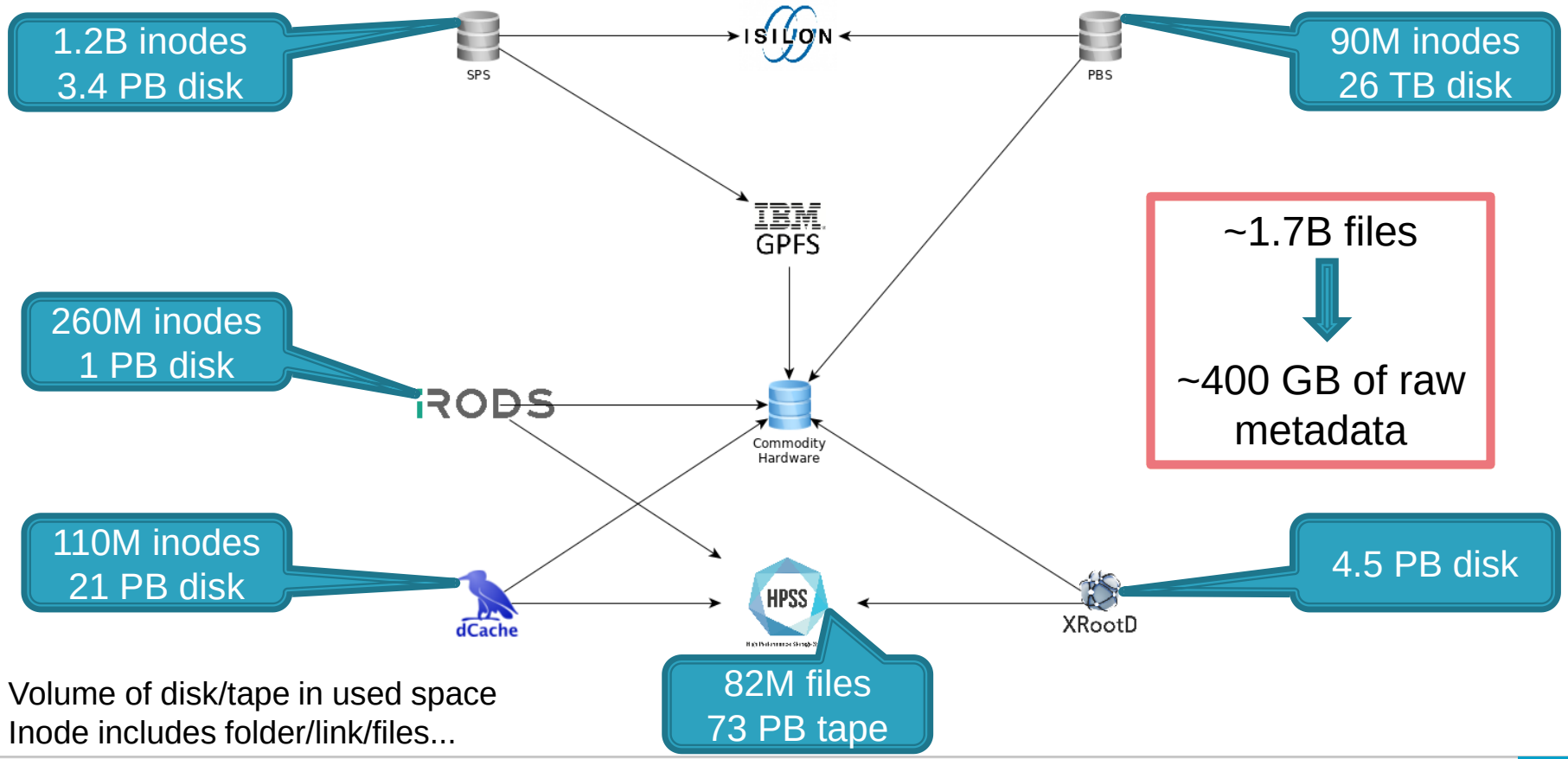
Storage elements at CC-IN2P3 1/2



SPS: is a Semi Permanent Storage with high performance used as large shared group space for data

PBS: is a Permanent Backed-up Storage used for home folders, web hosting, job applications…

Storage elements at CC-IN2P3 2/2



Volume of disk/tape in used space
Inode includes folder/link/files...

The future of data

- Exponential data growth is expected :
  - LHC Run 4: more data than ever before.
  - LSST: at least 15 PB only for the catalogues in the next 10 years.
  - Euclid: estimation of 10 PB in the next 5 years.

- In 2030, we expect up to 4 TB of metadata only.

Storage statistics project

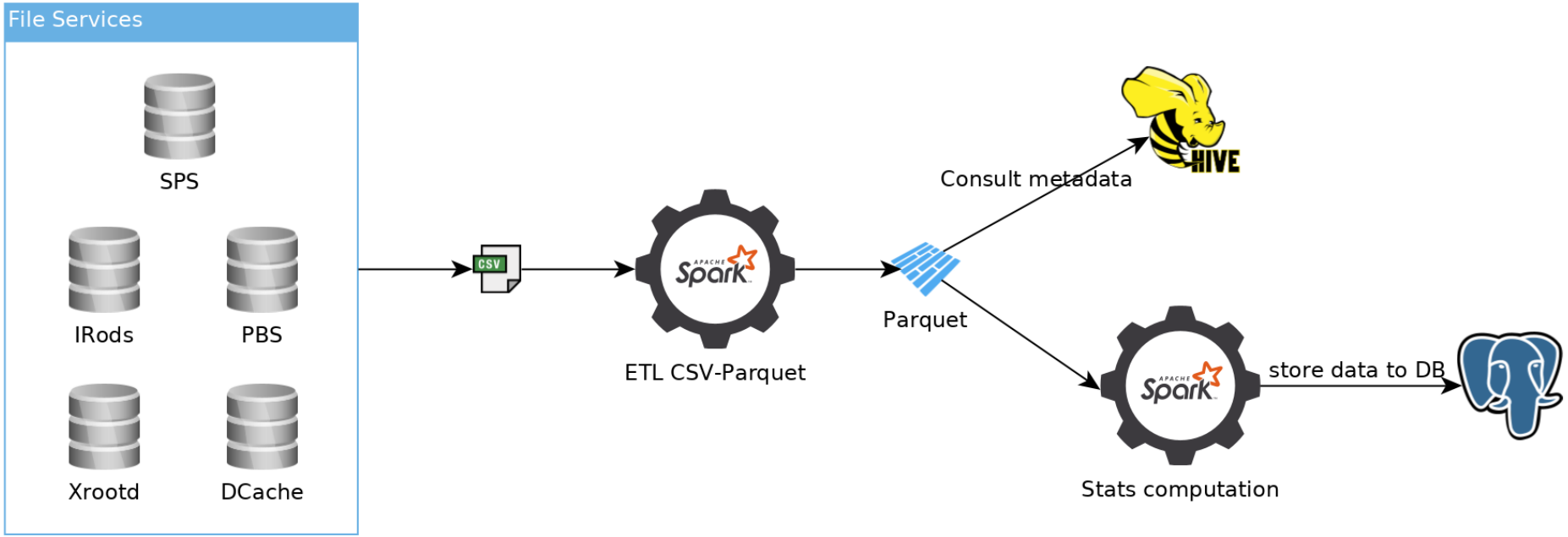
Why a new storage statistics project

- A centralized solution.
- A scalable solution.
- Compute:
  - Any simple stat (example: file size/user) on any storage element with the same code.
  - Storage element specific stat (example: file/server).
- Offer a simple interface:
  - To consult/develop statistics.
  - To consult storage element metadata.
- Regular stats

First objectives

- Reproduce existing statistics for each storage
- Provide access to consolidated metadata for customized requests.
- Integrate those data/stats into the CC-IN2P3 Management team tools.

Implementation 1/3



Implementation 2/3

Common columns

Storage Element	Path	Type	UID	GID	C-time	A-time	M-time	disk
IRods	✓	✓	?	?	✗	✗	✓	✓
DCache	✓	✓	✓	✓	✓	✓	✓	✓
HPSS	✓	?	✓	✓	✓	✓	✓	?
PBS/SPS	✓	?	✓	✓	✓	✓	✓	?
XRootD	✓	?	✓	✓	✓	✓	✓	?

Specific columns

Storage Element	ID	Server / pool	User name	Group name	Cr-Time	Read count	Write count	COS	Permission	Blocks	# links
IRods	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
DCache	✓	✓	?	?	✓	✗	✗	✗	✗	✗	✗
HPSS	✗	✗	?	?	✗	✓	✓	✓	✓	✗	✗
PBS/SPS	✗	✗	?	?	✗	✗	✗	✗	?	✓	✓
XRootD	✗	?	?	?	✗	✗	✗	✗	?	✓	✓

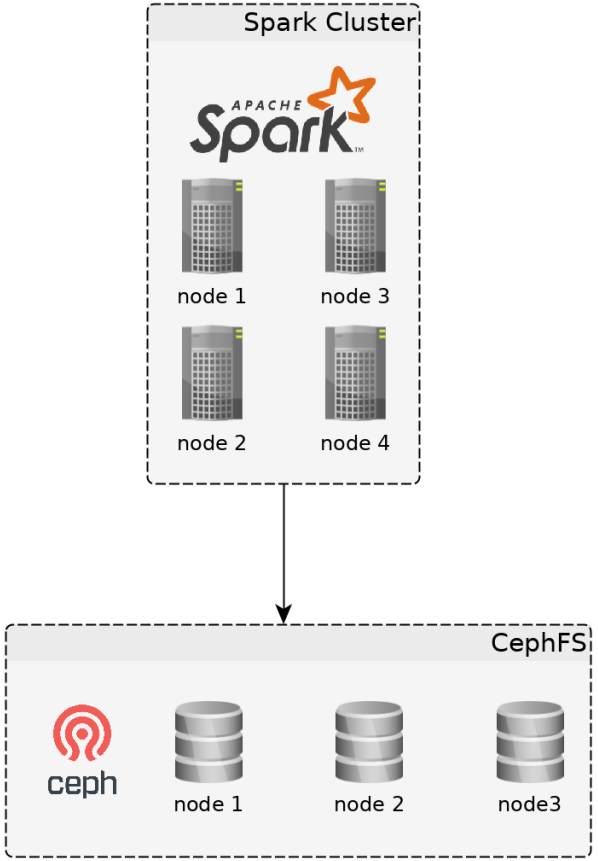
Unvailable	Guessable	Available
✗	?	✓

Implementation 3/3

Service	A name
Disk	True or False
C-time	Epoch timestamp
A-time	Epoch timestamp
M-time	Epoch timestamp
Type	File/link/folder/…
Directory	The directory containing the file
Filename	The basename of the file
extension	Trying to identify file type by reading extensions
Directory level 1 to 5	5 columns that contains the 5 first folder of the path
Option 1 to 5	5 columns that contains storage element specific values

Technical architecture

- Standalone Spark cluster
  - 4 virtual machines
  - 8 CPU per node
  - 32 GB Ram per node
- Storage
  - CephFS shared between all nodes
- Hive cluster
  - Ongoing tests on a separate platform

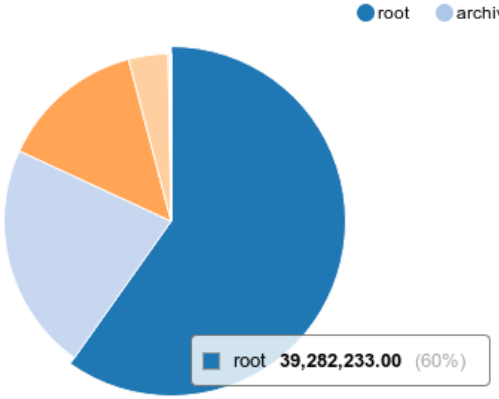


First results (1/4)

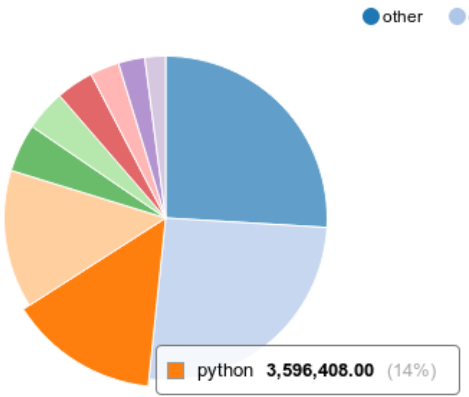
- Data already integrated :
  - PBS since 1st July 2019
  - DCache since mid September 2019

- Data to be integrated (in the coming weeks):
  - HPSS
  - IRods
  - XRootD
  - SPS

First results (2/4)



DCache

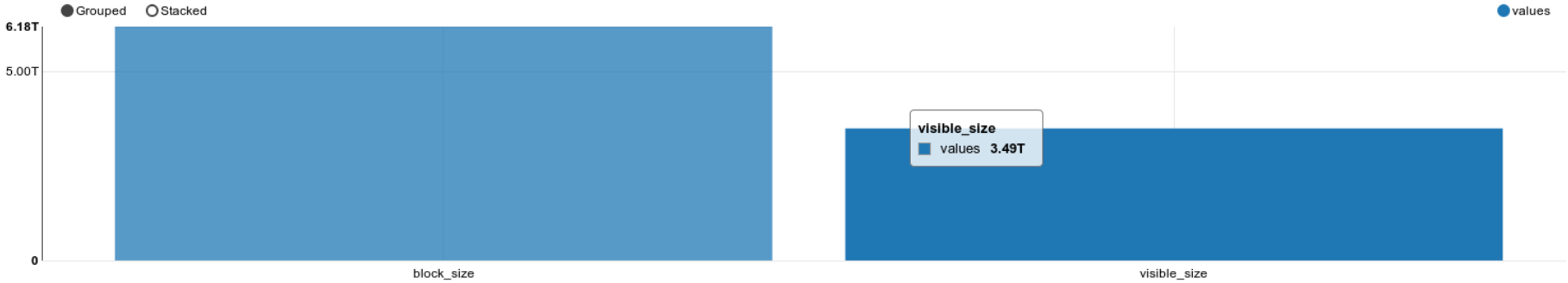


PBS Home folders

File extensions repartition in # of file

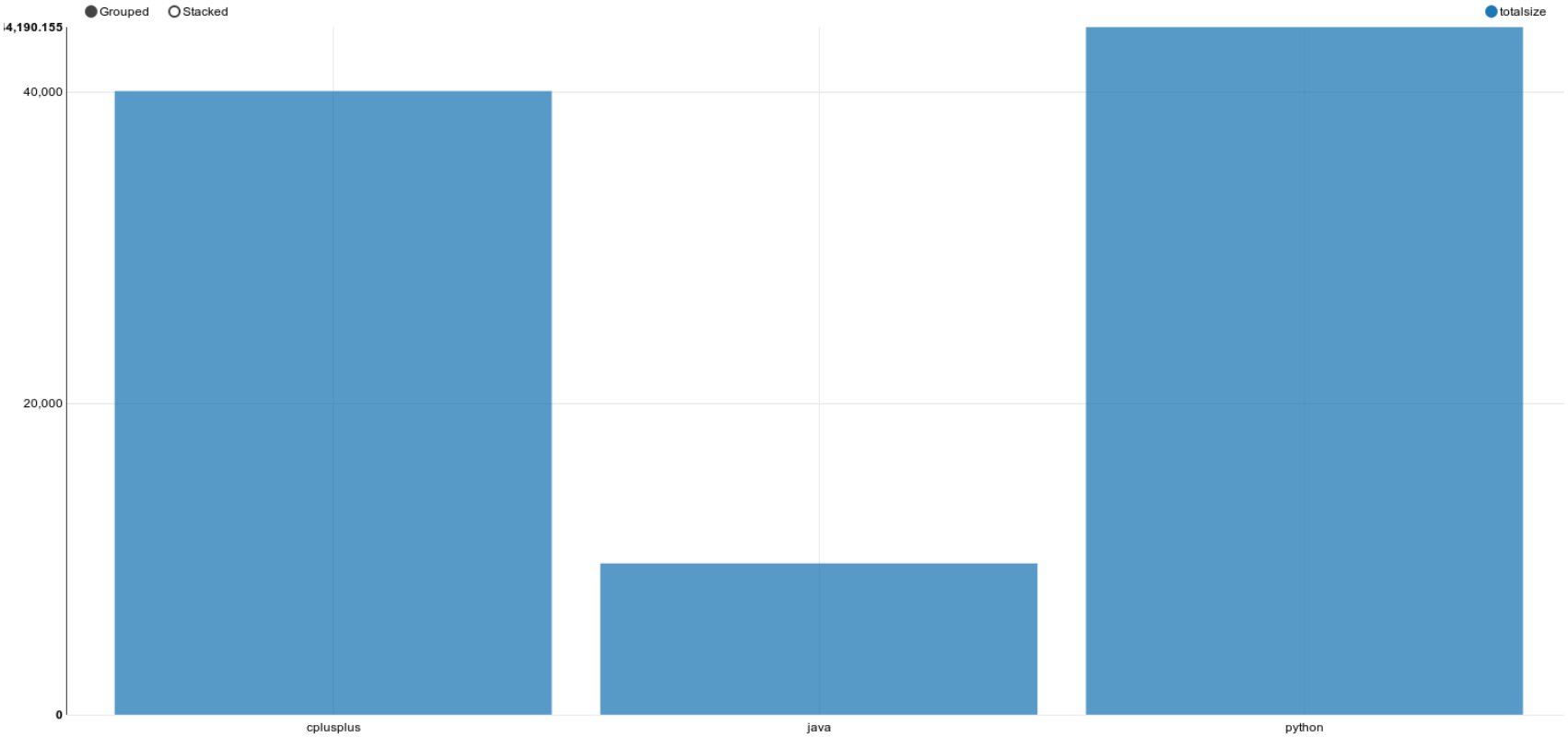
First results (3/4)

File size vs real block size in home directory



First results (4/4)

Most used programming language in home directory




Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

Conclusion

Conclusion

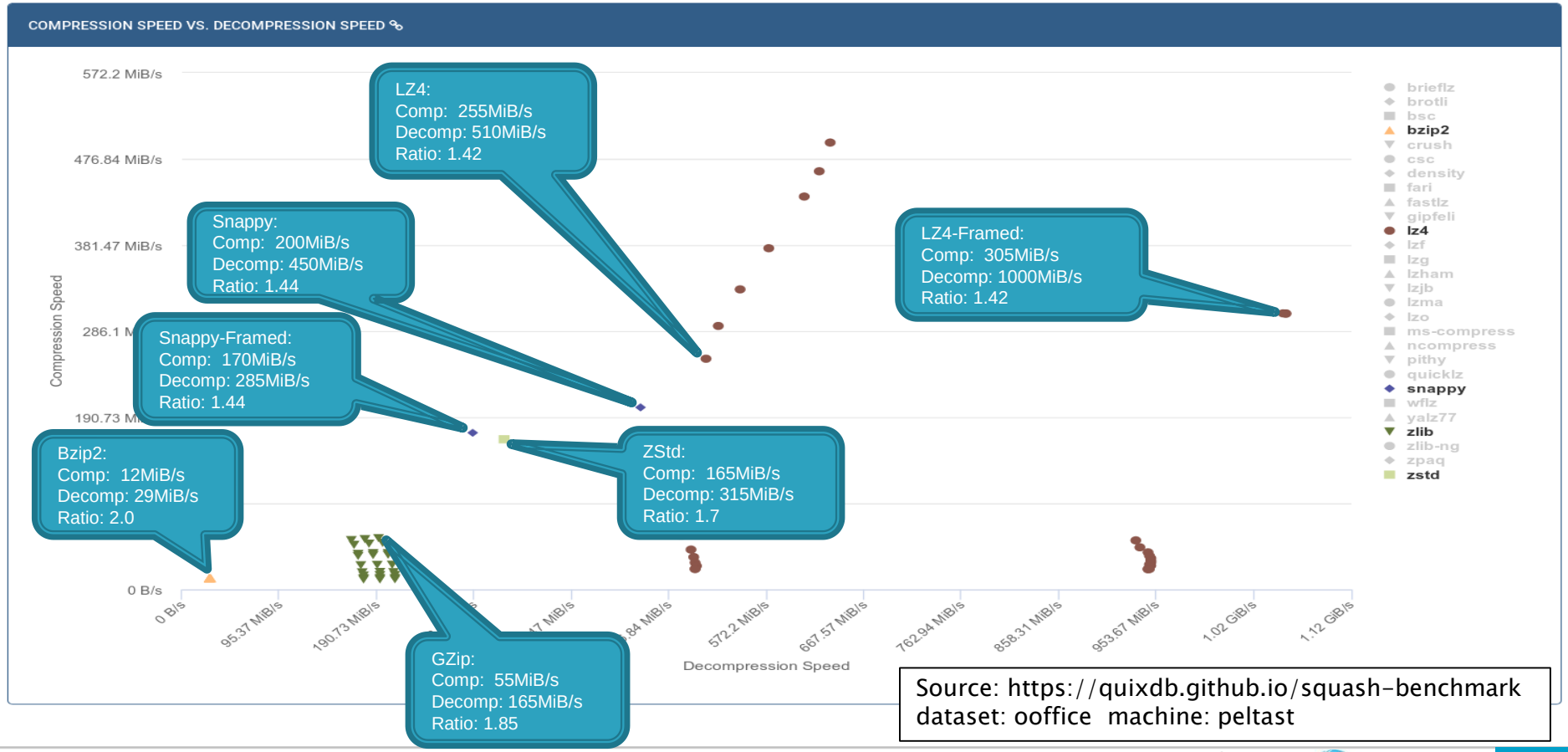
- This project is at an early stage:
  - Integration all storage elements.
  - Determine stats to compute on a regular basis.
  - Hive setup
  - Kerberos integration

- But we also have ideas for the future:
  - Add metadata for internal storage elements (TSM, database…).
  - Validate compliance of Data Management Plans.
  - Job file access/CPU analysis.

Thank you !

Questions and comments are welcome !

Compression algorithm



Source: https://quixdb.github.io/squash-benchmark
dataset: ooffice machine: peltast

Compression algorithm

- Compression matters :
  - Compressed data = less network transfer.
  - Compressed data = less storage.
  - Choose the correct algorithm for the correct task

Compression	Splitable	Hadoop/Spark native support
Z-Standard	No	Yes
LZ4	No	Yes
LZ4-Framed	Yes	No
Snappy	No	Yes
Snappy-Framed	Yes	No
GZip	No	Yes
BZip2	Yes	Yes

Metadata conversion csv to parquet benchmark

format	size	time
Bz2	620 MB	2900s
GZip	1900 GB	11000s

Hadoop ecosystem