(Apache) Spark for physicists

C. Arnault, G. Barrand, J.E. Campagne, J. Peloton and S. Plaszczynski

LAL, Univ. Paris-Sud, CNRS/IN2P3, Université Paris-Saclay, Orsay, France

October 18, 2018

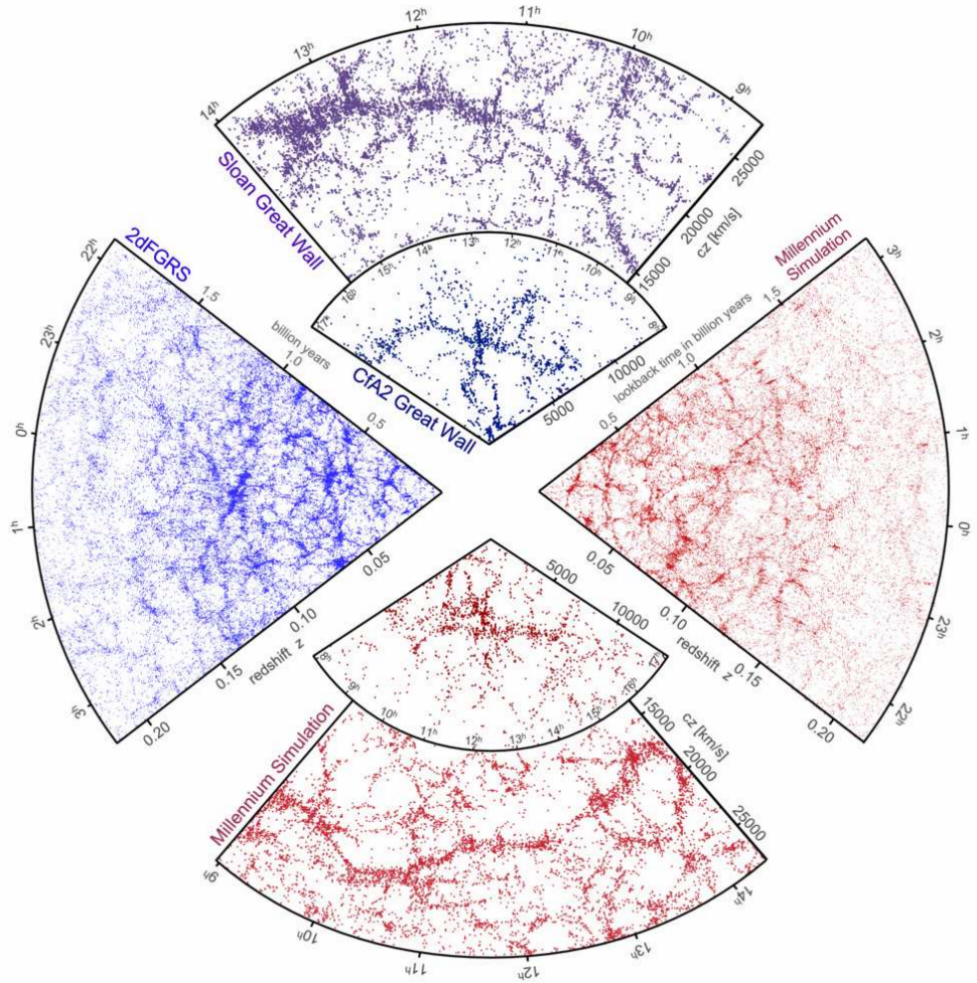
Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

The rise of big data computing

- 2004 `Google`: `mapReduce` programming model foundation of *distributed computing*
- 2006 `Hadoop` open-source framework (ecosystem) `HDFS`,`Hive`, `YARN`...
- 2004 `scala` (`java` ecosystem)
- 2009 `Spark`: research project at UC. Berkeley
- 2015 `Spark` SQL (dataframes)
- today: (`Apache`) `Spark` used by $\gtrsim$ 1000 companies

Meanwhile in cosmology...

Springel et al. (2006)



SDSS

BOSS $O(10^6)$ galaxies, DESI $O(10^7)$

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

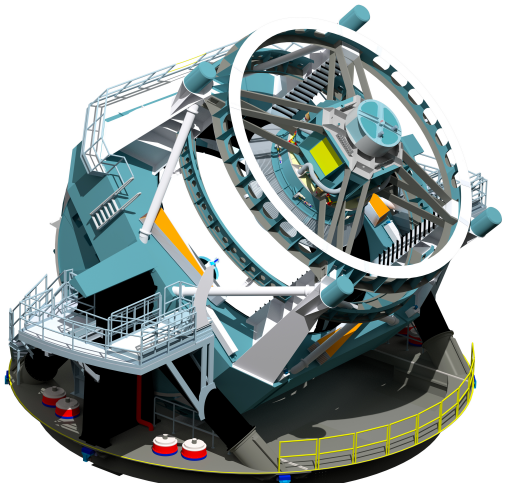
DES

$\mathcal{O}(10^7)$

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

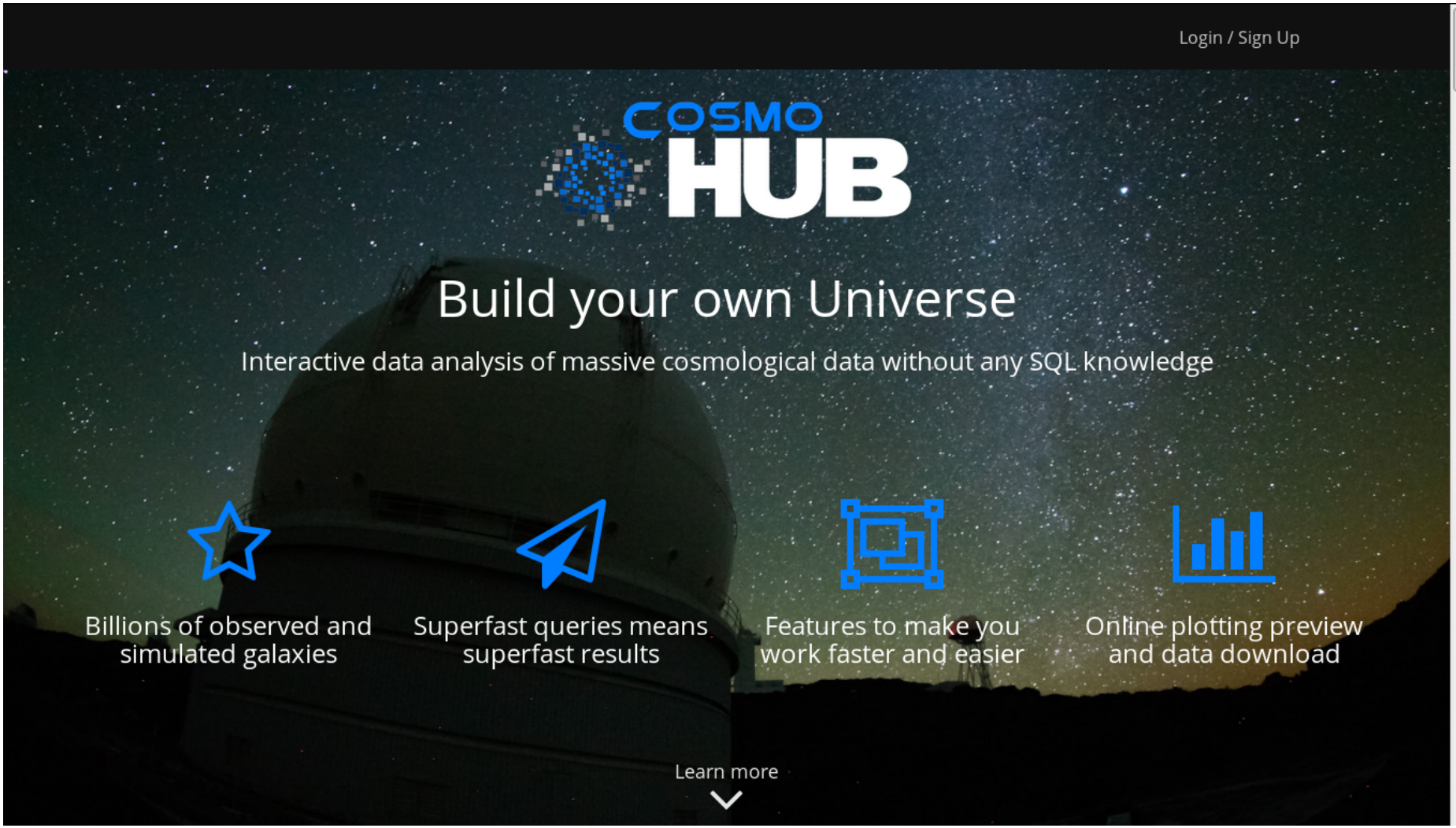
LSST

- start 2022 for 10 years
- 8.4m primary mirror
- 3.2 Gpixels camera
- 18000 deg^2
- 15 TB raw data/night
- +mocks...→big data

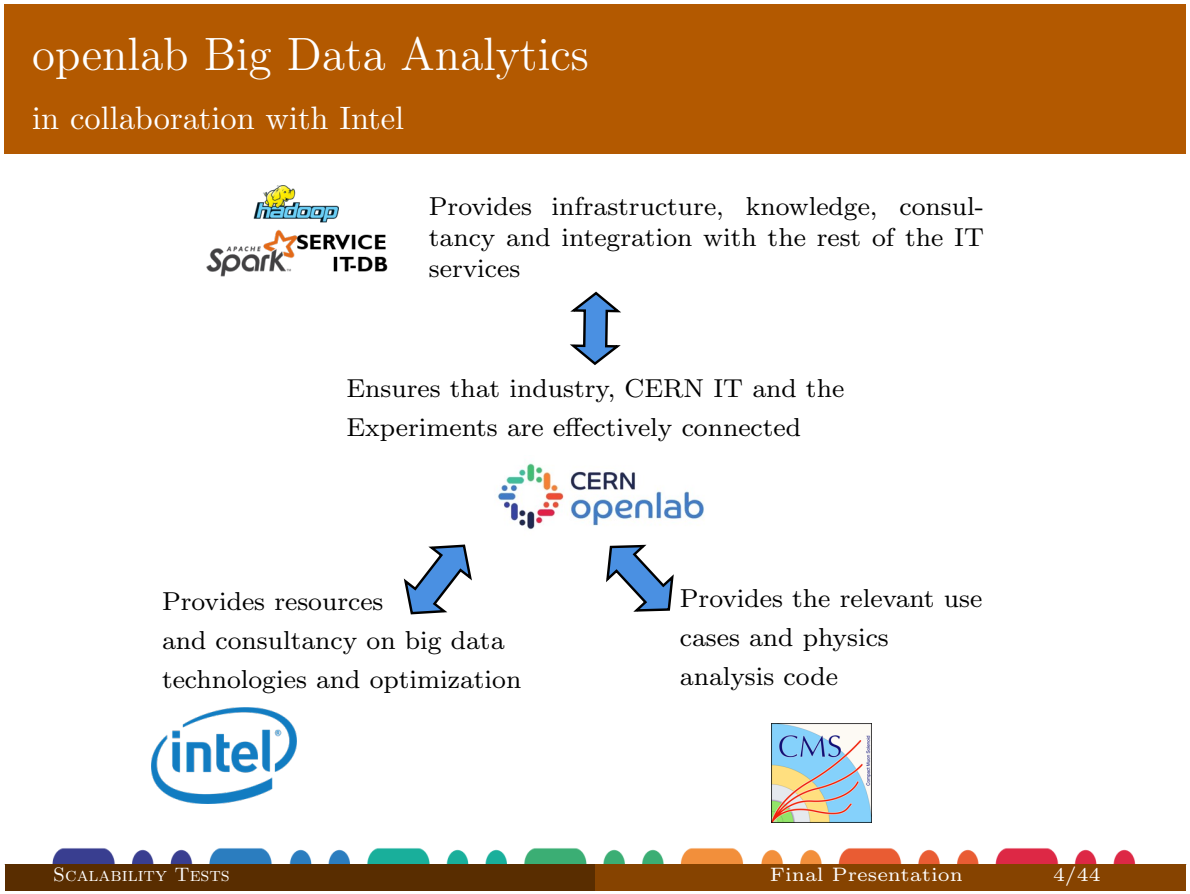


(may 2018, Cerro Pachón)

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT



Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT



OpenLab@CERN:
https://cernbox.cern.ch/index.php/s/6B89Z3wQgfZci7h

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

CMS Data Reduction Facility - Motivation

Why Data Analytics & Reduction with Spark?

- Investigate new ways to analyse physics data
- Improve resource utilization and time-to-physics
- Adopt new technologies widely used in the industry
  - Open the HEP field to a larger community
  - Improve chance of researchers on the job market outside academia

So what is Spark about?

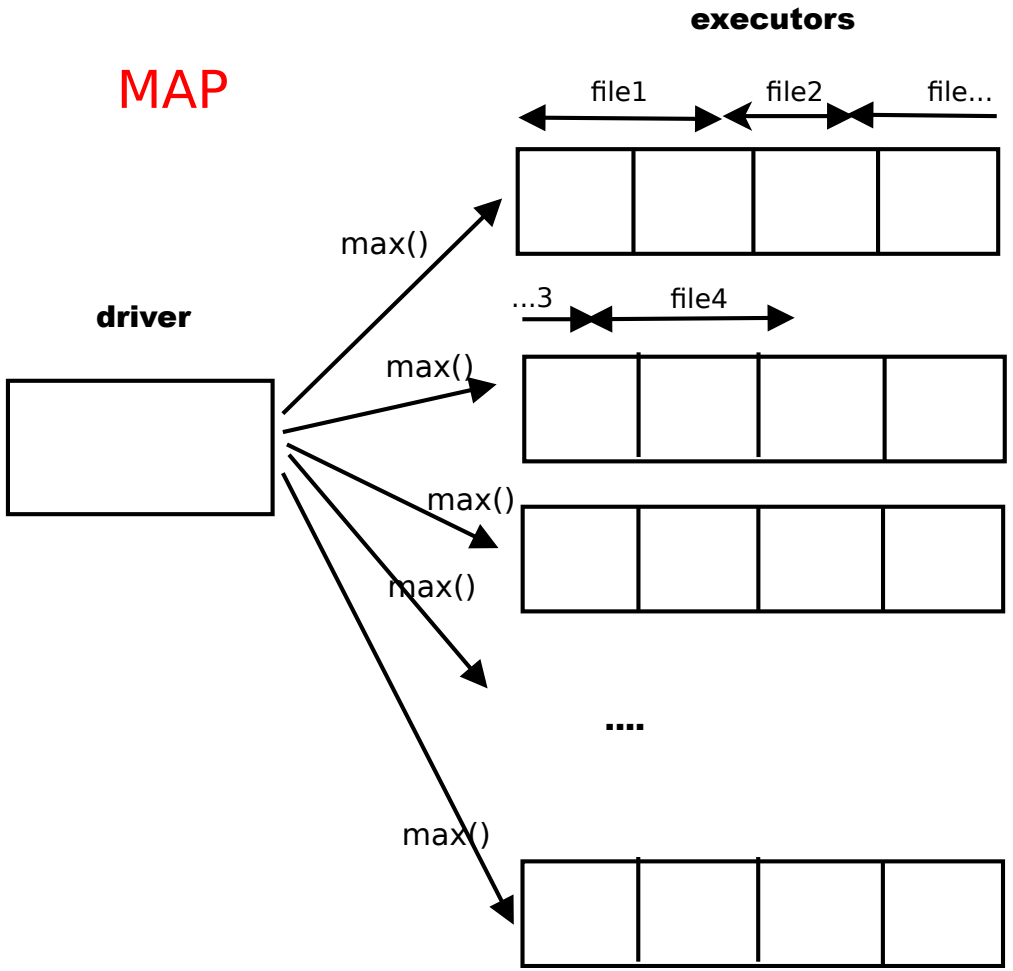
For a **large** volume of data it is more efficient
to **move the computation to the data** than the other way.

Spark = a framework to do it efficiently on distributed architecture
→scala, (java), python, R

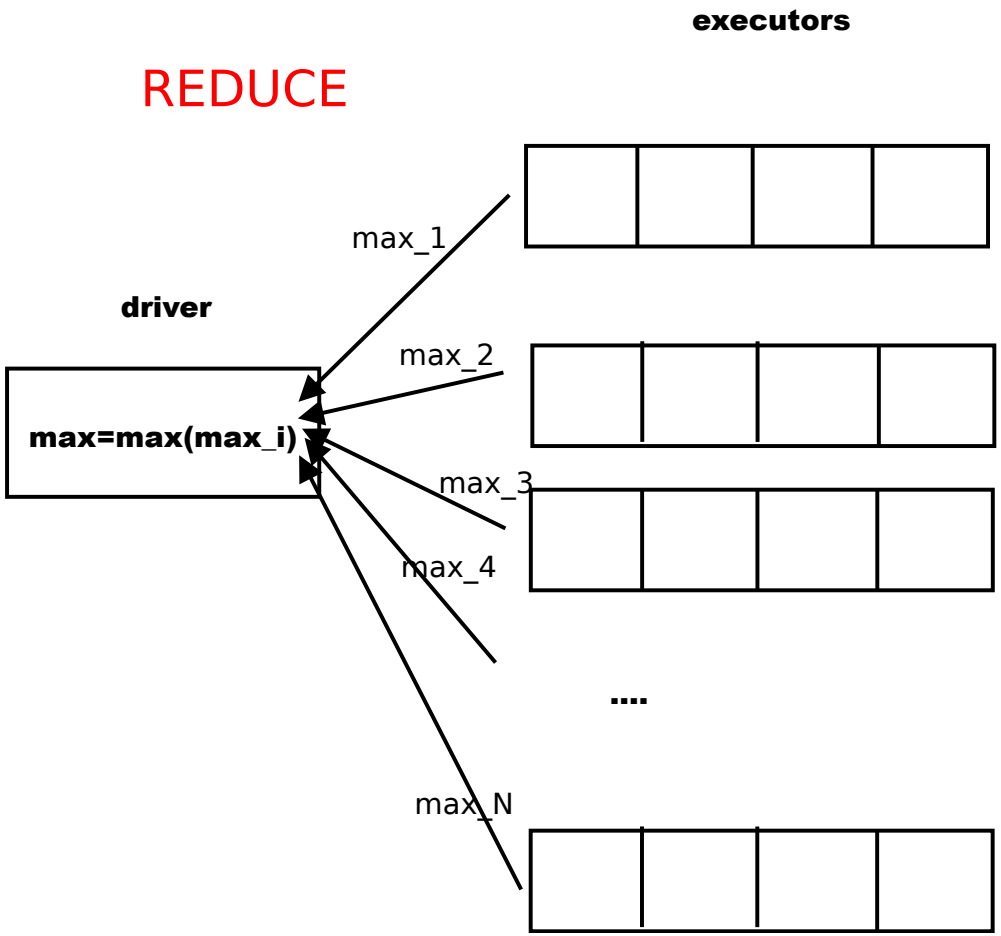
```
dataframe.transform1().transform2()....action()
```

This is Functional Programming (but you don't need to know it!)

Distributed computing



Advantage 1= coarse-grain parallelization



Advantage 2 ?

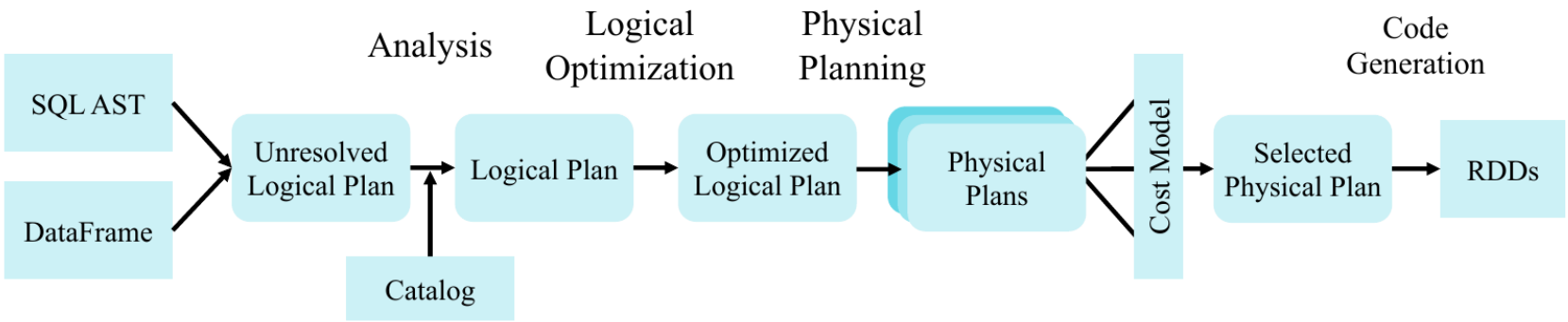
```
> data=spark.read.format("fits")\
              .load("path/to/110GB/of/fits/files")
> data.show(5)
+---------+---------+---------+---------+
|       RA|      Dec|        z|     zrec|
+---------+---------+---------+---------+
|225.80168|18.519966|2.4199903| 2.414322|
|225.73839|18.588171|2.4056022|2.2913096|
|225.79999|18.635067| 2.396816|2.3597262|
|225.49783|18.570776|2.4139786|2.3434482|
|225.57983|18.638515|2.3995044|2.3826954|
+---------+---------+---------+---------+
```

5s !/?

Lazy evaluation →optimization

- you are used to **imperative** languages (C/C++/FORTRAN...)
- here **lazy evaluation**: code is an 'expression-language' that allows to build a Direct Acyclic Graph (DAG)
- transformations (`load,map,filter..`) →update DAG
- actions (`count,collect,show..`) →optimize DAG (`Catalyst`) and run

Advantage 2= Automatic pipeline optimization



the Machine does it better than you! →`Spark` reason of success

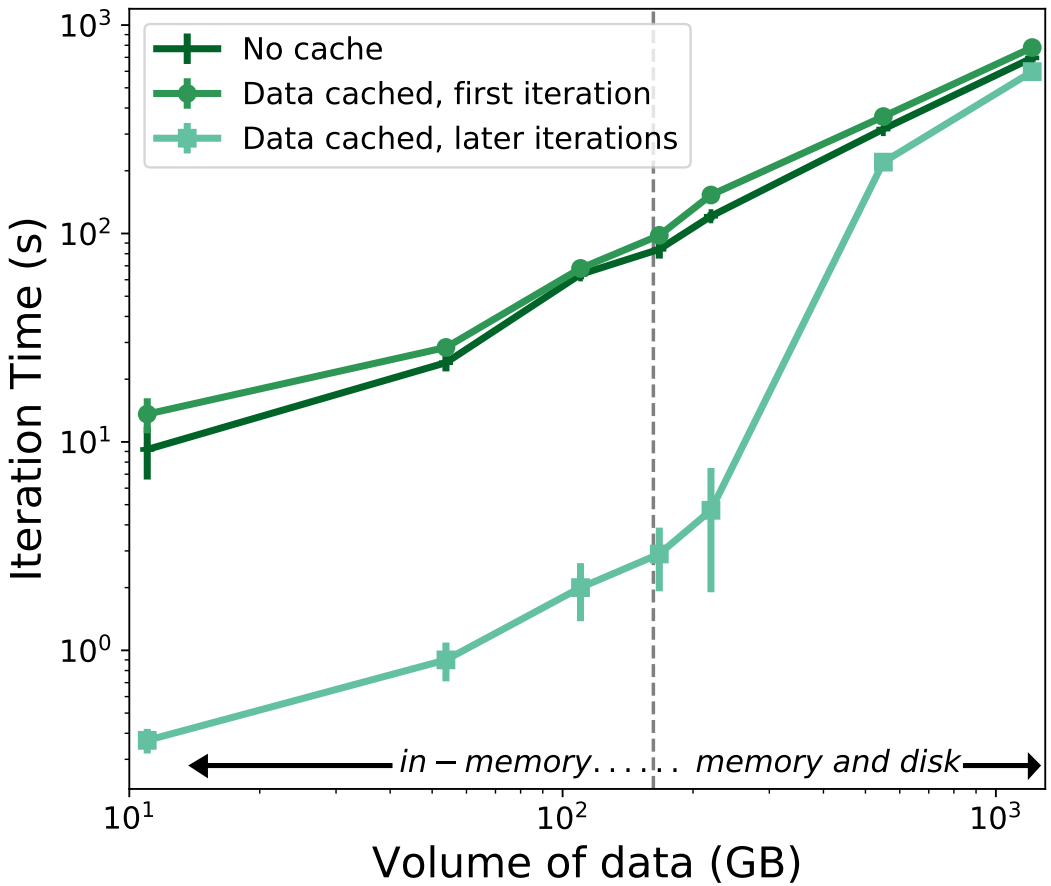
Advantage 3= in memory work (cache)

Put the data in cache as if you had a **huge** RAM

- ex: 110 GB on a small cluster (8 workers)
- 1TB at NERSC

Then you can work interactively

Advantage 4= scaling



A use-case in cosmology

- generate LSST 10Y of galaxies with fast sim
  `https://github.com/damonge/CoLoRe.git`
- →110GB of FITS files. 6. 10^9 galaxies

- goal is to have a quick *interactive* look at what was generated (`python`)
- this is different from *developing* software (`scala`)

The U-PSUD cluster

- 9 machines: 18 cores+ 32 GB RAM each
- cache = 0.6 TOTmem=144 GB →enough to hold the data
- HDFS

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

Data sources

- Spark was rather developed to ~~spy~~ study your habits: poorly structured data (text, although `avro, parquet`)
- need to develop support for more complex structures
- popular formats in astronomy: FITS, HDF5
- but no good native FITS/HDF5 Spark reader exists...

`spark-fits` high performance connector (+lib) Peloton et al. (2018)

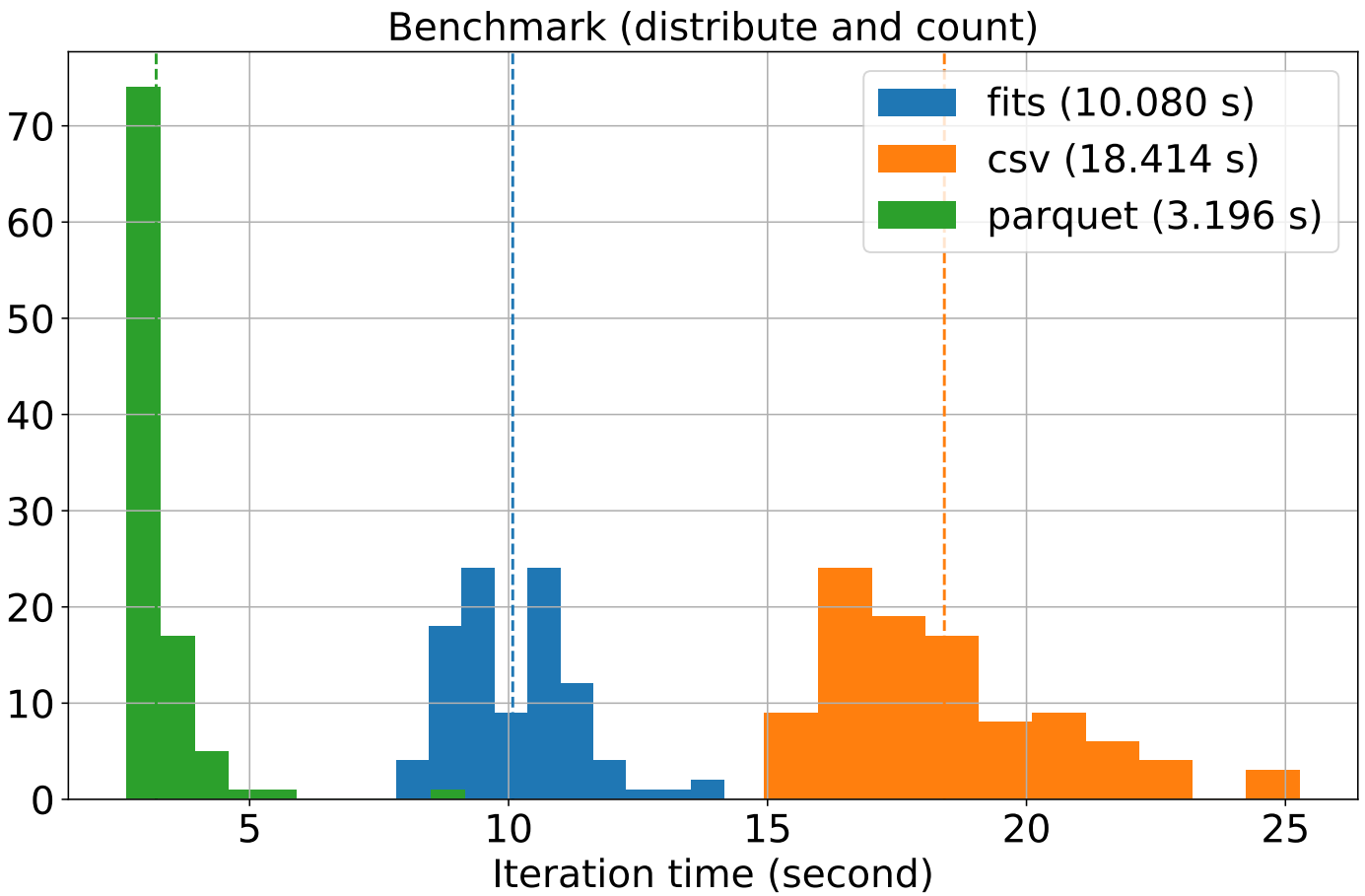
Reading a FITS file

Nothing to do on the user-side: just copy your standard FITS file to your cluster and then

```
> df=spark.read.format("fits").option("hdu",1)\
      .load("hdfs:path/to/fits/dir/")

> df.printSchema()

root
 |-- TYPE: integer (nullable = true)
 |-- RA: float (nullable = true)
 |-- DEC: float (nullable = true)
 |-- Z_COSMO: float (nullable = true)
 |-- DZ_RSD: float (nullable = true)
```

Dataframe similar to R/pandas
('n-tuple' in HEP since the 70's, 'binTable' in FITS since the 80's...)

Performances



Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

1. Selecting columns

```
> gal=df.select("RA","Dec", \
      (col("Z_COSMO")+col("DZ_RSD")).alias("z"))

> gal.show(5)

+---------+----------+----------+
|       RA|       Dec|         z|
+---------+----------+----------+
| 265.1168| -79.96222| 0.5590986|
| 258.0575| -79.84589|0.55854577|
|261.24503|-80.293274|0.56063706|
|279.49026| -80.23766|0.56124765|
| 285.2853| -79.96391|0.56244487|
+---------+----------+----------+
```

2. Put them in cache

```
> gal.cache().count()

5926764680
```

≃100s

3. Basic stats

```
> gal.describe('z').show()

+-------+--------------------+
|summary|                   z|
+-------+--------------------+
|  count|            11713638|
|   mean|0.27755870587729775|
| stddev| 0.1639140896858911|
|    min|       -0.0017737485|
|    max|           0.5673544|
+-------+--------------------+
```

3s

Approx median (1/1000 rel. prec): 30s

4.Histograms (the distributed way)

starting from z add a column of bin numbers

```
> zbin=gal.select("z",
        ((gal['z']-zmin-dz/2)/dz).astype('int')\
          .alias('bin'))

+----------+---+
|         z|bin|
+----------+---+
| 0.5590986|  98|
|0.55854577|  97|
|0.56063706|  98|
|0.56124765|  98|
|0.56244487|  98|
|0.55902207|  98|
...
```

Histograms (the distributed way) 2

GroupBy this column

```
> zbin=zbin.groupBy("bin")...
+----------+----------+--------------------+----+
|         z|     group                     | bin|
+----------+----------+--------------------+----+
| {0.5590986,0.56063706,0.56124765,...}    | 98 |
| {0.55854577. ...                         } | 97 |
...
```

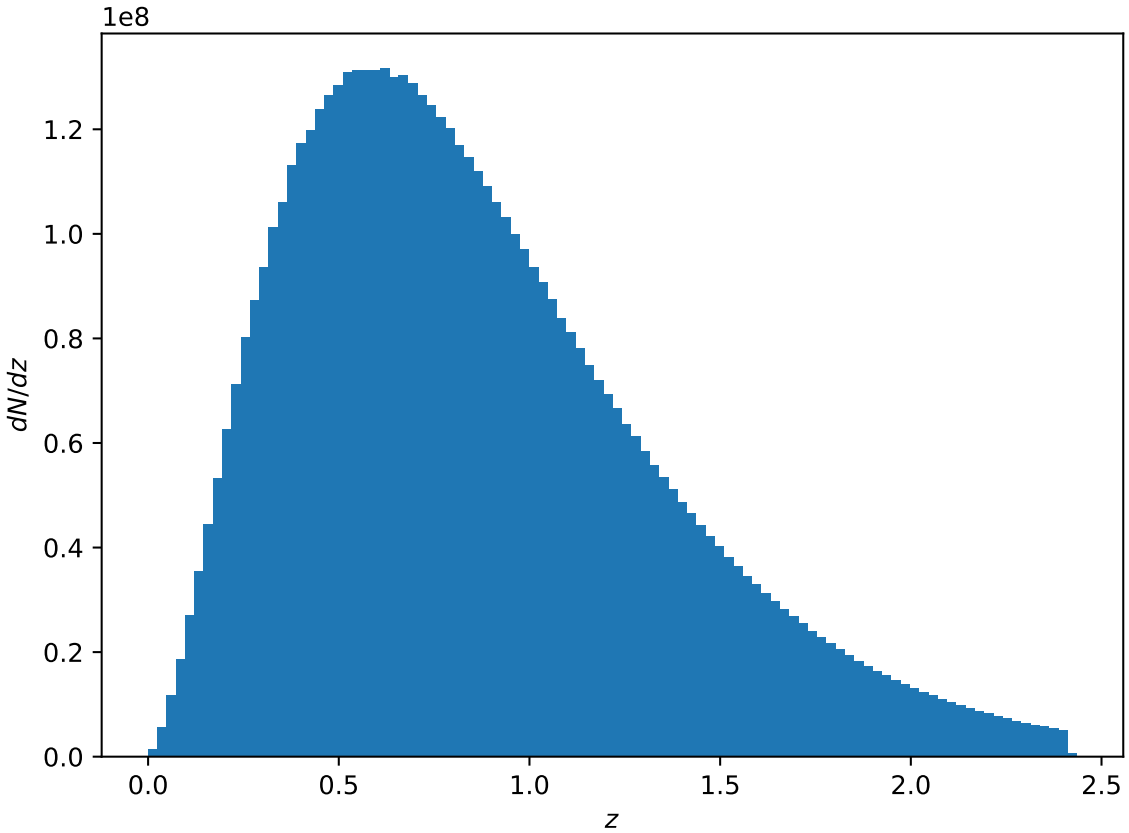
and count by group

```
> zbin=zbin.groupBy("bin").count()
+---+------+
|bin| count|
+---+------+
| 98|116607|
| 97|117410|
```

Histograms (the distributed way) 3

- sort in ”bin” ascending order
- add locations (bin centers)

```
> h=zbin.sort("bin",ascending=True)
> histo=h.select((zmin+dz/2+h['bin']*dz)\
                    .alias('zbin')\
                  ,"count")
+--------------------+------+
|                zbin| count|
+--------------------+------+
|0.0010718921013176 44|237445|
|0.0067631733976304 53|178469|
| 0.012454454693943 26|132612|
| 0.018145735990256 07|102854|
| 0.023837017286568 88| 96153|
...
```



≃11s

on 6.10^9 data! imperative way (sequential): 45 mins

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

5. User-Defined Functions (UDF)

```
binNum=udf(lambda z: int((z-zmin-dz/2)/dz))
zbin=gal.select(gal.z,\
     binNum(gal.z).alias('bin'))
```

115s !

```
@pandas_udf("float", PandasUDFType.SCALAR)
def binNumber(z):
    return pandas.Series((z-zmin)/dz)

zbin=gal.select(gal.z,\
     binNumber("z").astype('int').alias('bin'))
```

40s

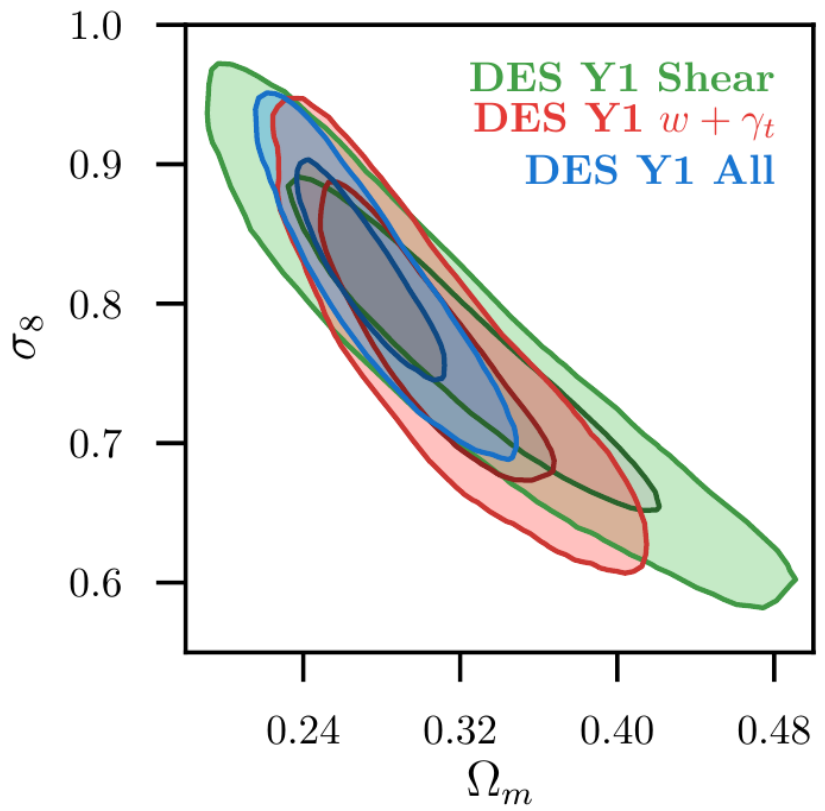
Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

6. Tomography

- compute over-densities in redshift regions (shells)
- project onto a map (`HEALPix`)
- compute cross/auto power-spectra
- clustering+ weak lensing= powerful probe for cosmology

DES Y1

Parameter	Prior
Cosmology	
Ω_m	flat (0.1, 0.9)
A_s	flat $(5 \times 10^{-10}, 5 \times 10^{-9})$
n_s	flat (0.87, 1.07)
Ω_b	flat (0.03, 0.07)
h	flat (0.55, 0.91)
$\Omega_\nu h^2$	flat$(5 \times 10^{-4}, 10^{-2})$
w	flat $(-2, -0.33)$
Lens Galaxy Bias	
$b_i (i = 1, 5)$	flat (0.8, 3.0)
Intrinsic Alignment	
$A_{\rm IA}(z) = A_{\rm IA}[(1+z)/1.62]^{\eta_{\rm IA}}$	
$A_{\rm IA}$	flat $(-5, 5)$
$\eta_{\rm IA}$	flat $(-5, 5)$
Lens photo-z shift (red sequence)	
$\Delta z_{\rm l}^1$	Gauss $(0.001, 0.008)$
$\Delta z_{\rm l}^2$	Gauss $(0.002, 0.007)$
$\Delta z_{\rm l}^3$	Gauss $(0.001, 0.007)$
$\Delta z_{\rm l}^4$	Gauss $(0.003, 0.01)$
$\Delta z_{\rm l}^5$	Gauss $(0.0, 0.01)$
Source photo-z shift	
$\Delta z_{\rm s}^1$	Gauss $(-0.001, 0.016)$
$\Delta z_{\rm s}^2$	Gauss $(-0.019, 0.013)$
$\Delta z_{\rm s}^3$	Gauss $(+0.009, 0.011)$
$\Delta z_{\rm s}^4$	Gauss $(-0.018, 0.022)$
Shear calibration	
$m^i_{\rm METACALIBRATION} (i = 1, 4)$	Gauss $(0.012, 0.023)$
$m^i_{\rm IM3SHAPE} (i = 1, 4)$	Gauss $(0.0, 0.035)$

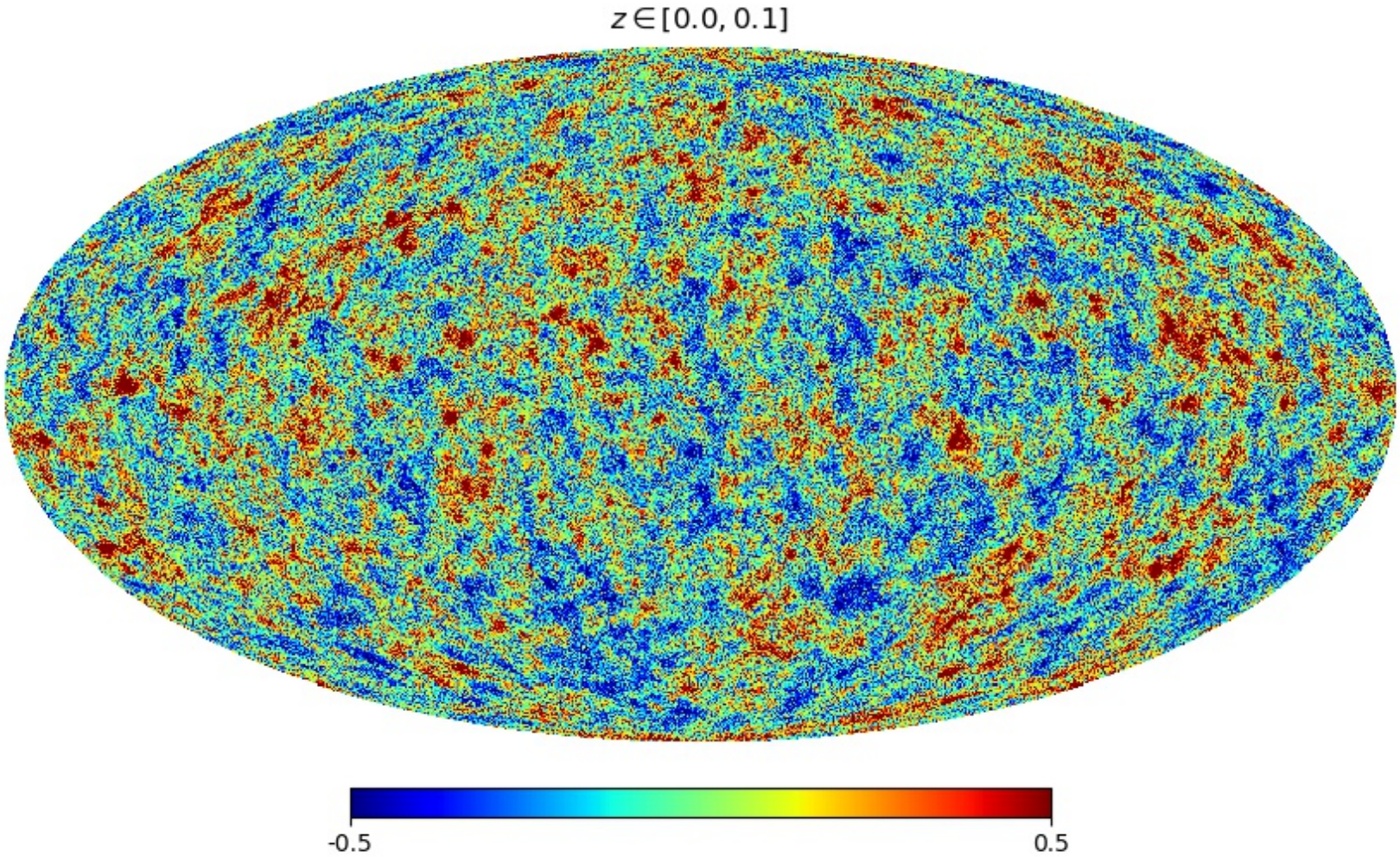


Troxel et al. (2018)

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

Implementation

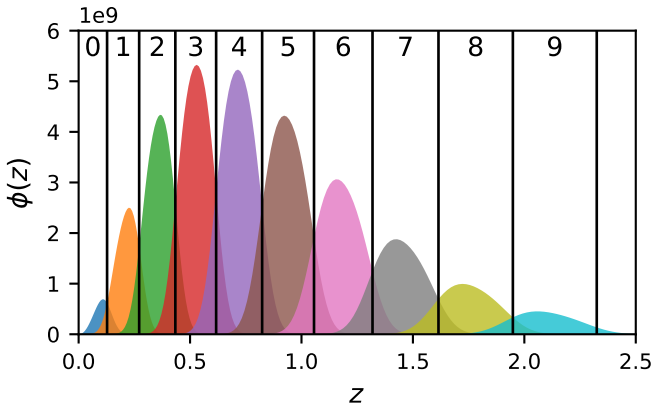
```
@pandas_udf('int', PandasUDFType.SCALAR)
def Ang2Pix(ra,dec):
    theta=np.radians(90-dec)
    phi=np.radians(ra)
    return pandas.Series(\
           healpy.ang2pix(nside,theta,phi)
           )

shell=gal.filter(gal['z'].between(z1,z2))

map=shell.select(Ang2Pix("RA","Dec")\
    .alias("ipix"))\
    .groupBy("ipix").count()
```

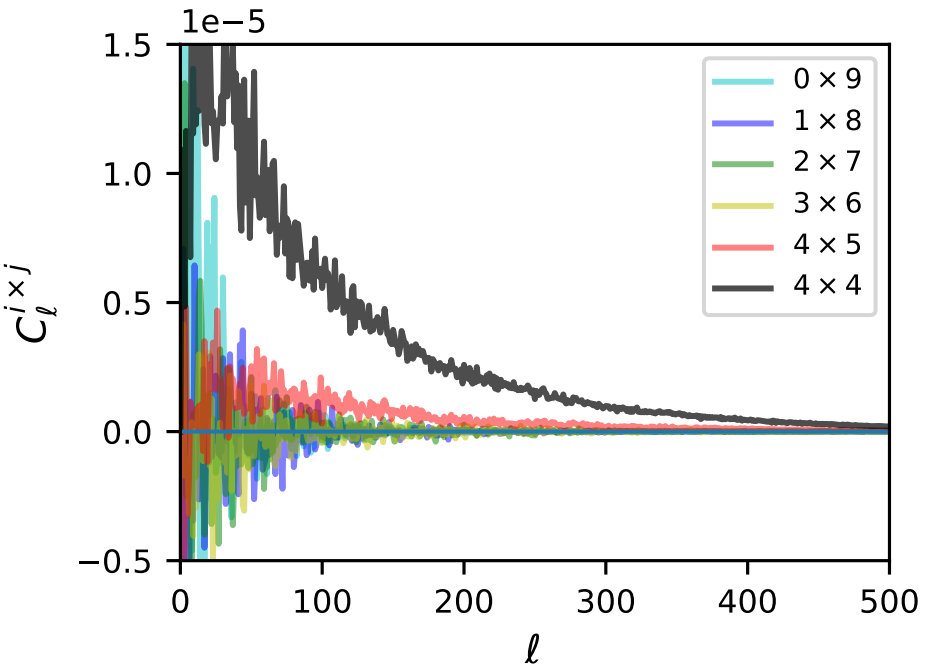


$$\frac{N_i - \bar{N}}{\bar{N}}$$

Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

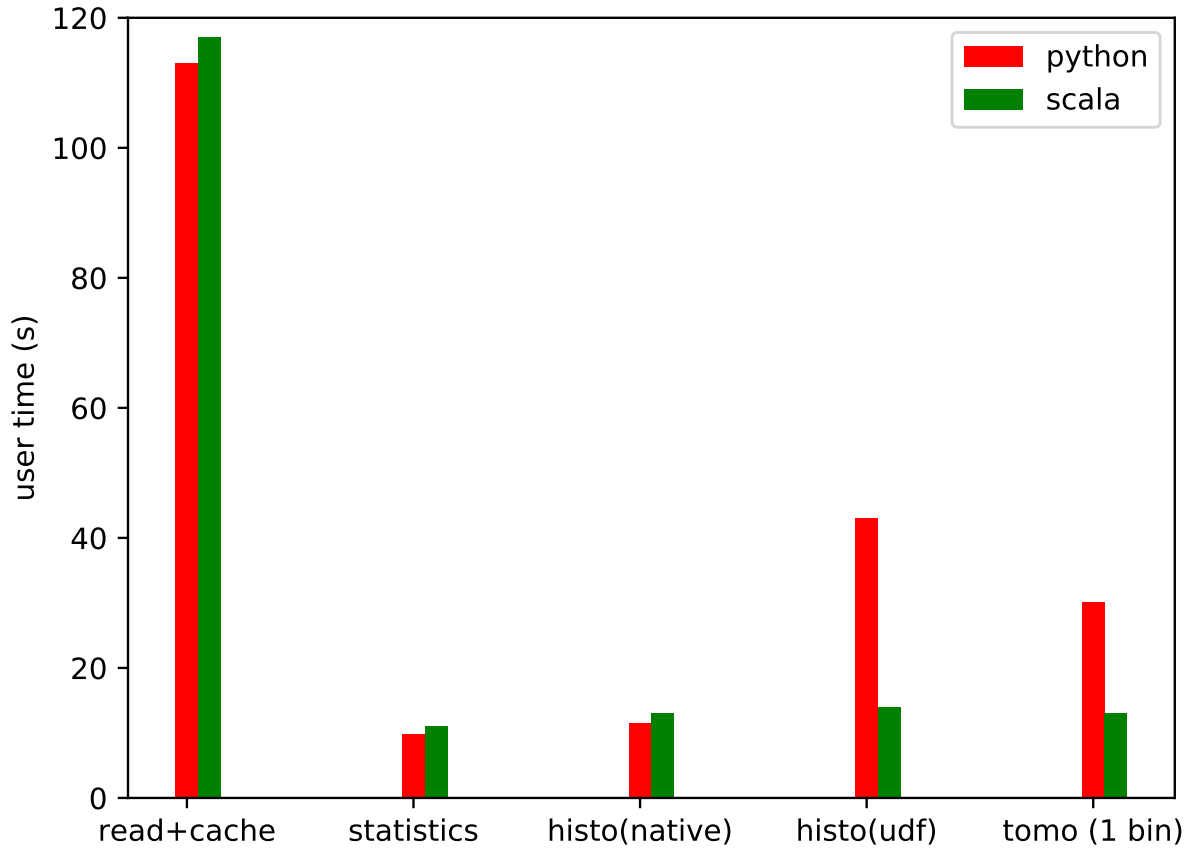
Power spectra





≃30 s/shell

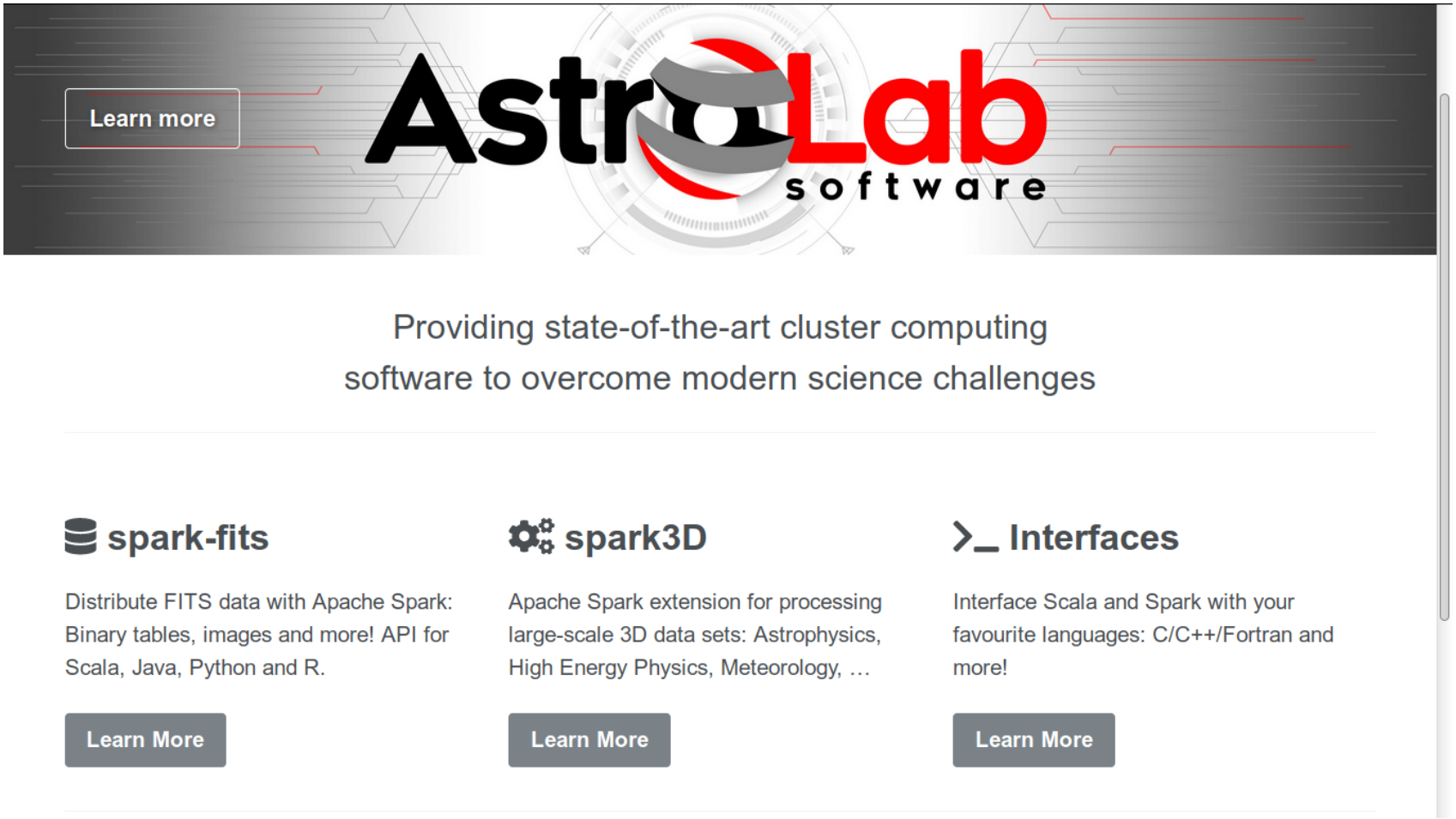
python or scala?



User conclusions

- you may perform (simple) analyzes with `Spark` in `python` (or `R`,`scala`).
- Open/read any FITS file.
- use Dataframes (not RDDs)
- use as much as possible native `Spark SQL` functions
- interface to external `python` code with `pandas_udf` (for `C/Fortran` wait a bit...)
- details in Plaszczynski et al. (2018)

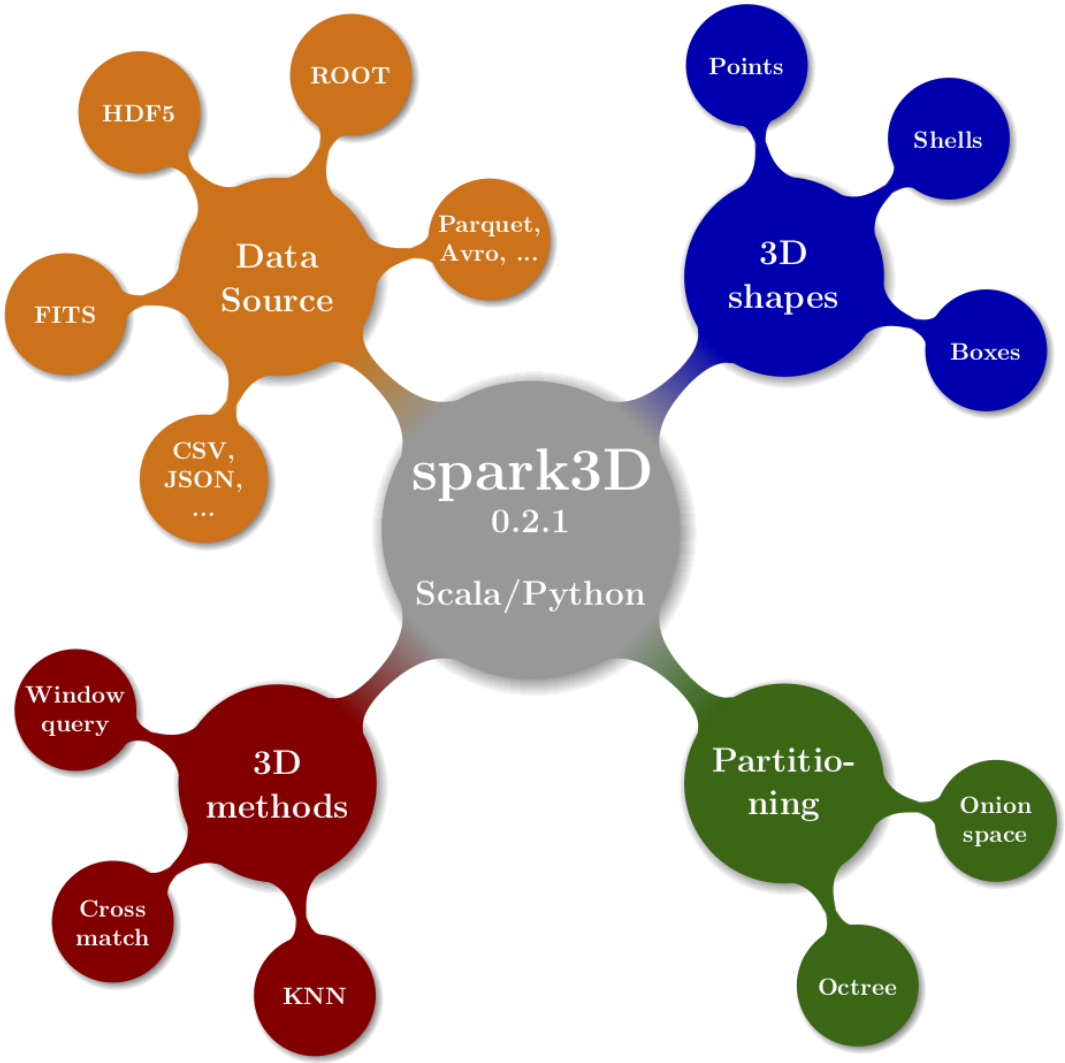
https://astrolabsoftware.github.io



→(re)appropriate a project born in a public lab.

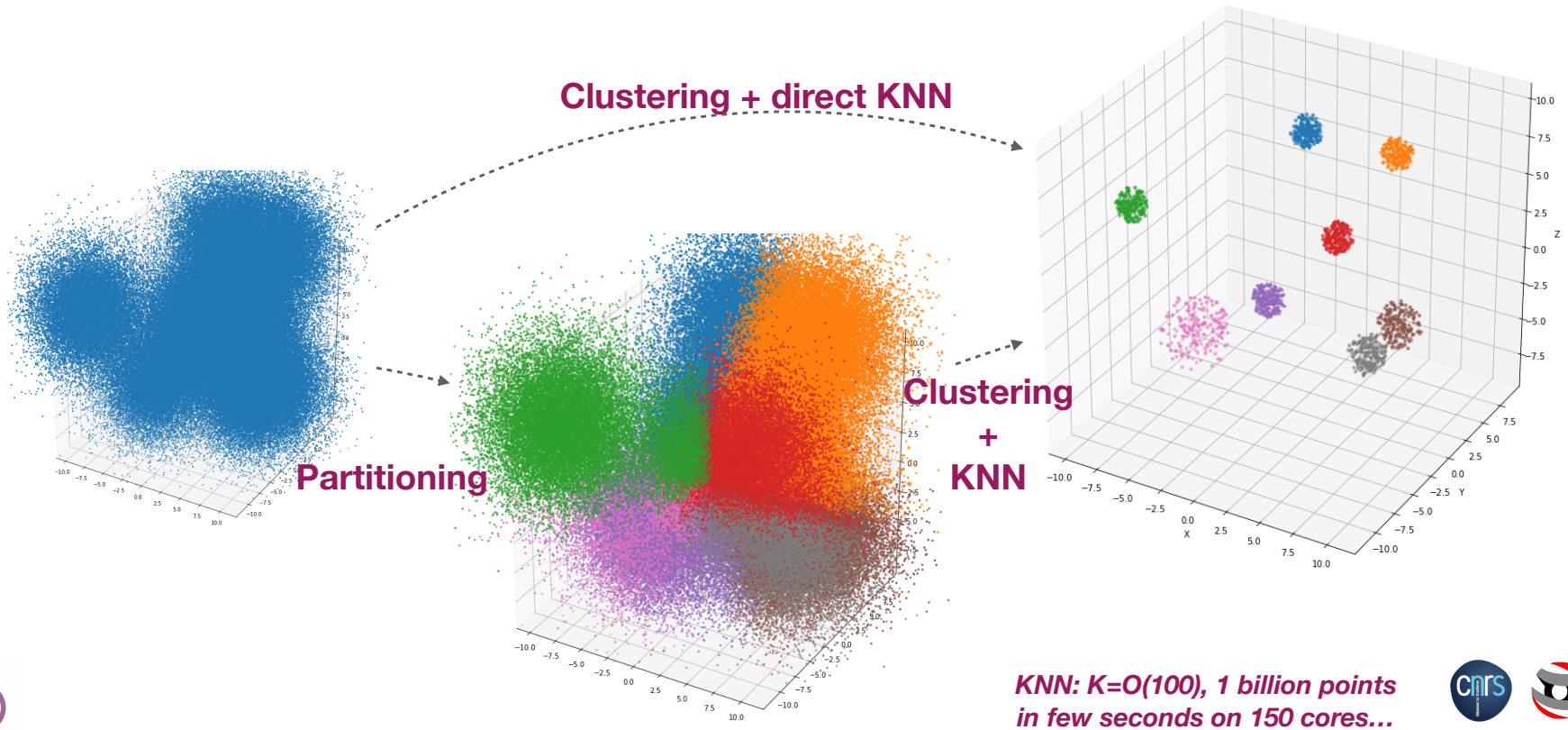
Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

Spark-3D



Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

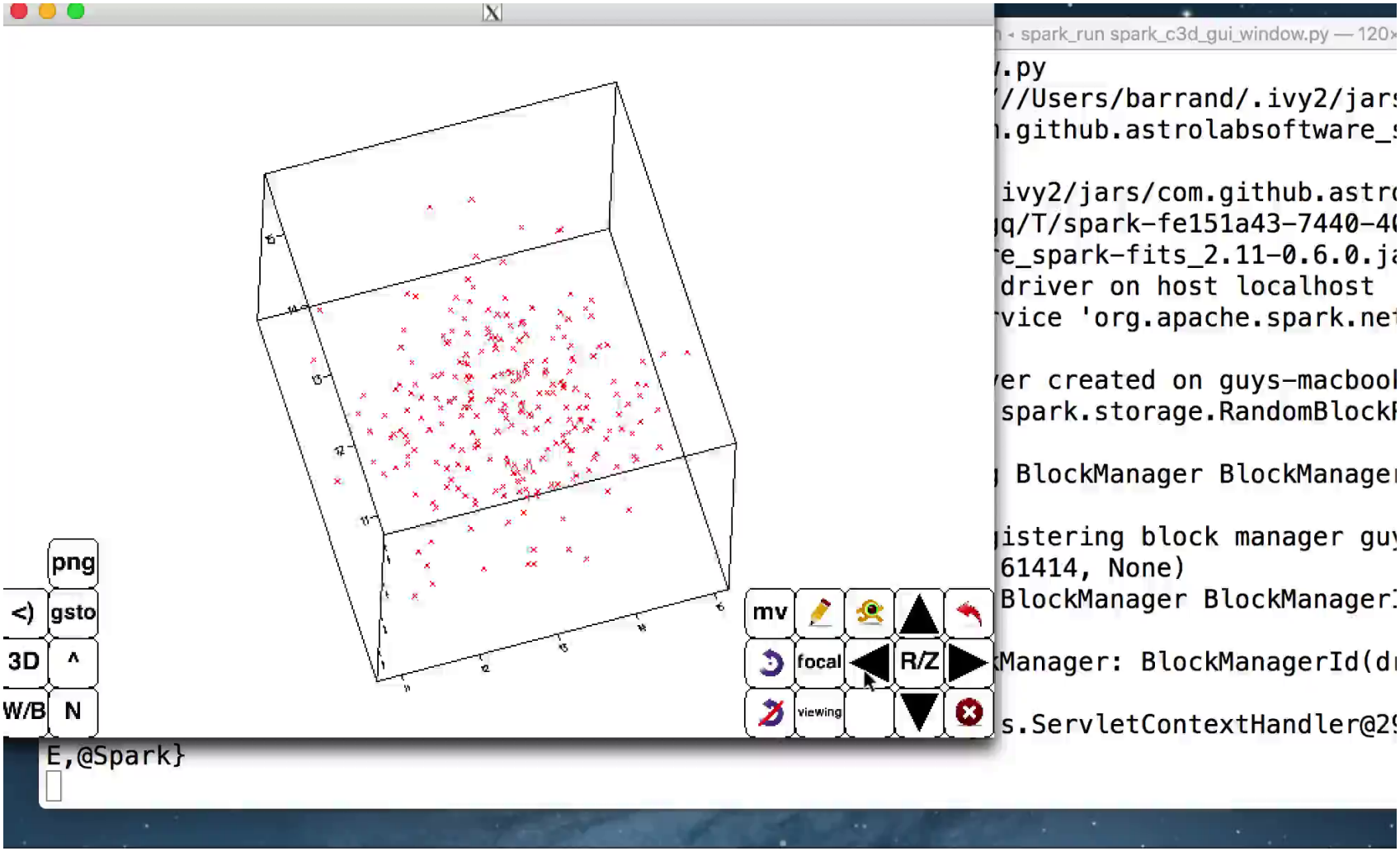
spark3D: K Nearest Neighbours



KNN: K=O(100), 1 billion points in few seconds on 150 cores…

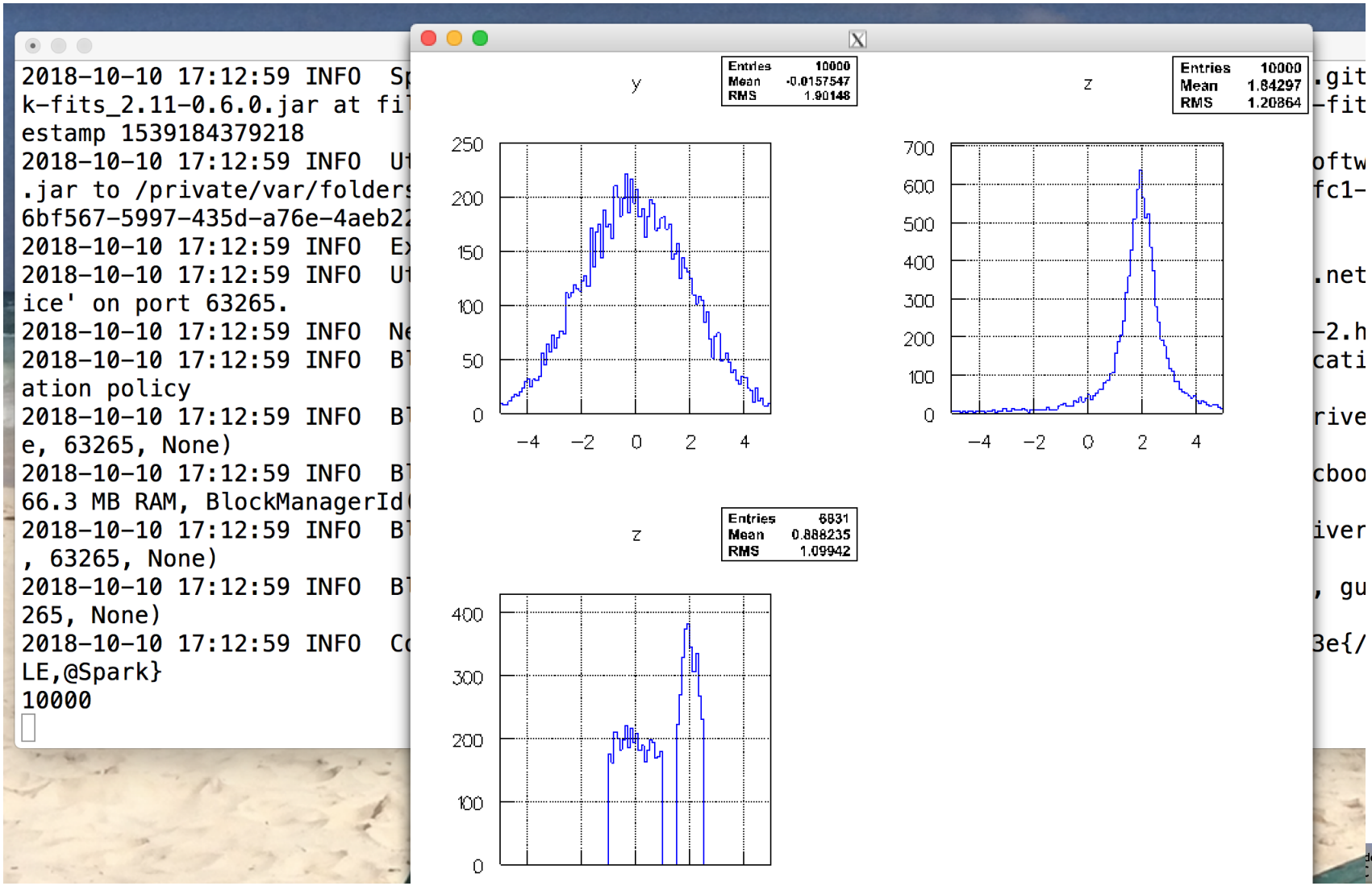
clustering algorithm =k-Means →DBSCAN

3D visualization



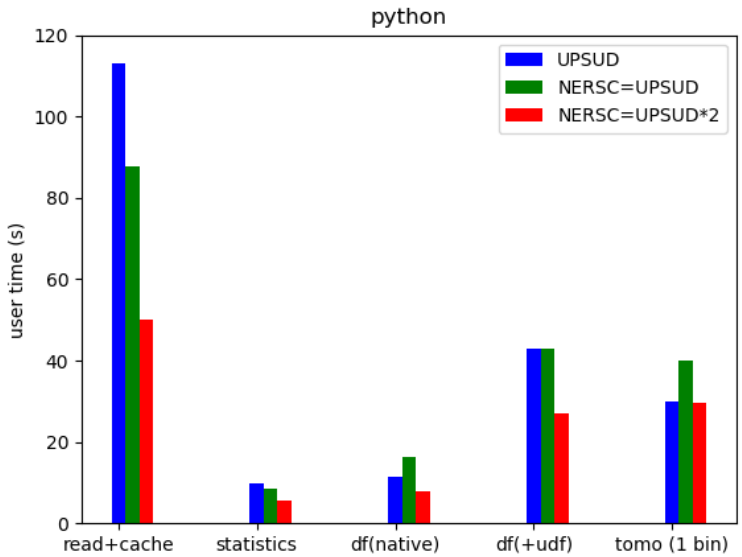
based on `inlib/exlib (C++), GL-ES+X11, SWIG`

Client/Server architecture



Spark & HPC

- Spark and HPC can be combined →goldmine
- sometimes you cannot avoid communication between executors (*shuffle*)→better on HPC
- you can interface `scala` →`C(++)/Fortran` with `JNA` (hot topic)
- Spark available @NERSC (as shifter image): no sizable loss.



Refondation des Labos
CSNSM - IMNC - IPNO - LAL - LPT

Conclusions

if you are having issues with

- analyzing easily large set of data
- algorithmic performances on large data sets

contact us.

FP

- concepts from math logic theory ($\lambda-$calculus): Curry-Howard correspondence
- *imperative* languages rather developed (Turing machines)
- Lisp, Haskell..
- `scala` used by some scientific communities (genomics)

main ideas:

- *functions* are fundamental and basic types
- Referential transparency $\Longrightarrow f(x)+f(x)=2f(x)$
- no idea of 'state' (but monades)
- immutability (no side-effect), recursivity

quite clear /concise/robust codes but not very used until `scala`/`Spark`

DBSCAN

- *density-based spatial clustering of applications with noise*
- 2 parameters (ϵ, *minPts*): FoF=DBSCAN(minPts=0)
- can find clusters of arbitrary morphology, no filtering needed
- pb: until recently no decent distributed version available (orders of mag wrt FoF HPC)
- *Song, H. and Lee, J., "RP-DBSCAN: A Superfast Parallel DBSCAN Algorithm Based on Random Partitioning," In Proc. 2018 ACM Int'l Conf. on Management of Data (SIGMOD), Houston, Texas, June 2018.*
- RP-DBSCAN (cell-based Random Partitioning): scales now
- we are testing it →promising

Full refs

Peloton, J., Arnault, C., & Plaszczynski, S. 2018, ArXiv e-prints, arXiv:1804.07501

Plaszczynski, S., Peloton, J., Arnault, C., & Campagne, J. E. 2018, ArXiv e-prints, arXiv:1807.03078

Springel, V., Frenk, C. S., & White, S. D. M. 2006, Nature, 440, 1137, arXiv:astro-ph/0604561

Troxel, M. A., MacCrann, N., Zuntz, J., et al. 2018, Phys. Rev. D, 98, 043528, arXiv:1708.01538