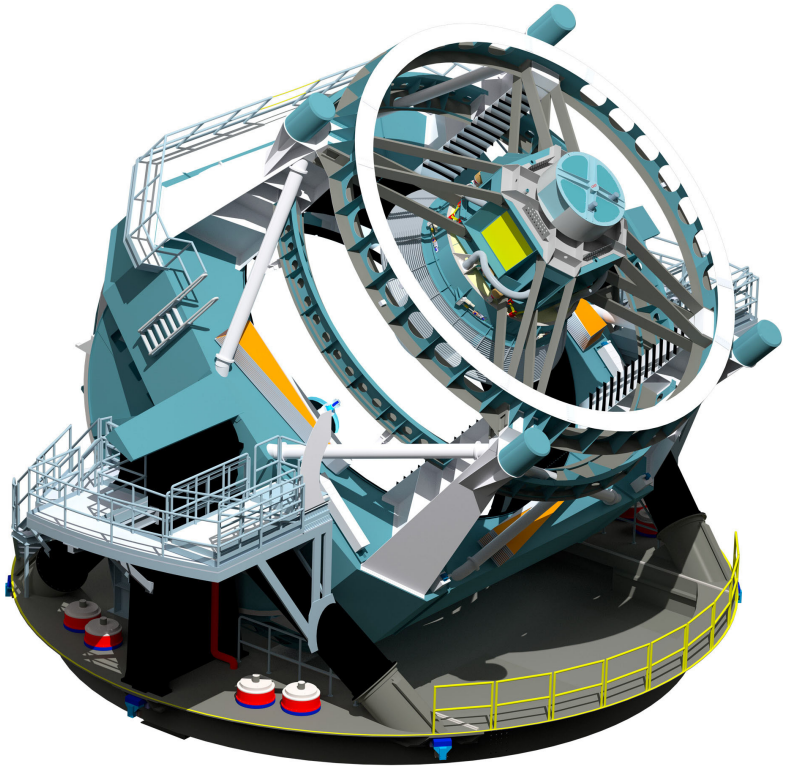
Analyzing astronomical data with Apache Spark

Julien Peloton
with Christian Arnault, Stéphane Plaszczynski & Jean-Eric Campagne

Laboratoire de l'Accélérateur Linéaire

July 16, 2018

The Large Synoptic Survey Telescope (LSST)



- Start observation in 2022, 10 year survey of the sky from Chile.
- Goal: among several, understanding Dark Energy.
- Collaboration: International, 1000+ contributors.

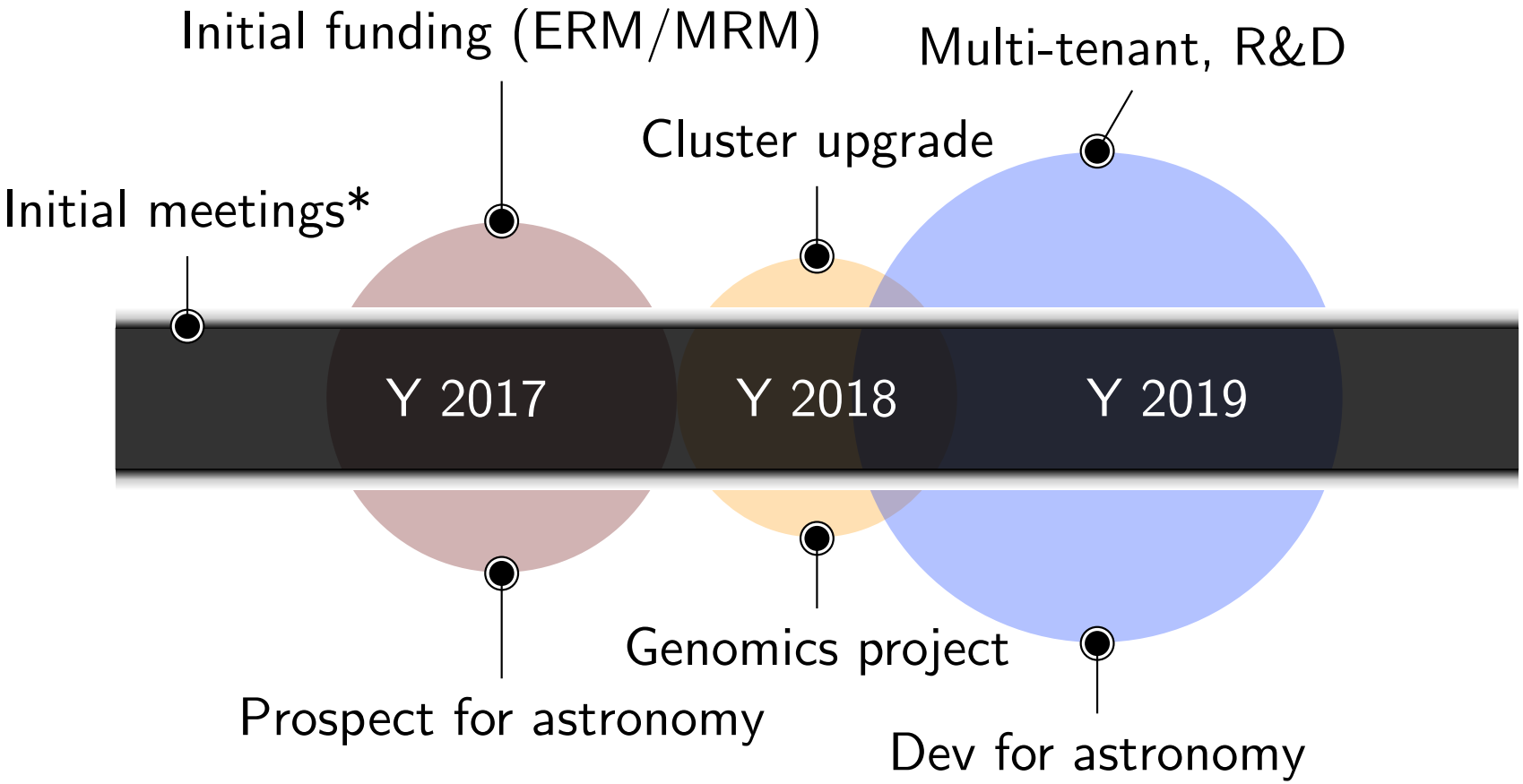
Entering a big data era

- 3200 megapixel camera (each image has the size of 40 full moons), O(15) TB every night.
- Catalogs of 37 billion stars and galaxies at the end ($>$ 2 orders of magnitude wrt current ones) + mocks.
- Traditional tools are not always adapted for the analysis:
  - A popular file format to store and manipulate astro data is FITS: not serialized.
  - Standard database tools: data ingestion takes a long time.
  - Standard analysis tools: reproducibility of the analysis is challenging, interactivity becomes impossible, scalability will be a nightmare.

$\rightarrow$ what could bring current big data technology in this context?

Spark@LAL: evolution 2016–now

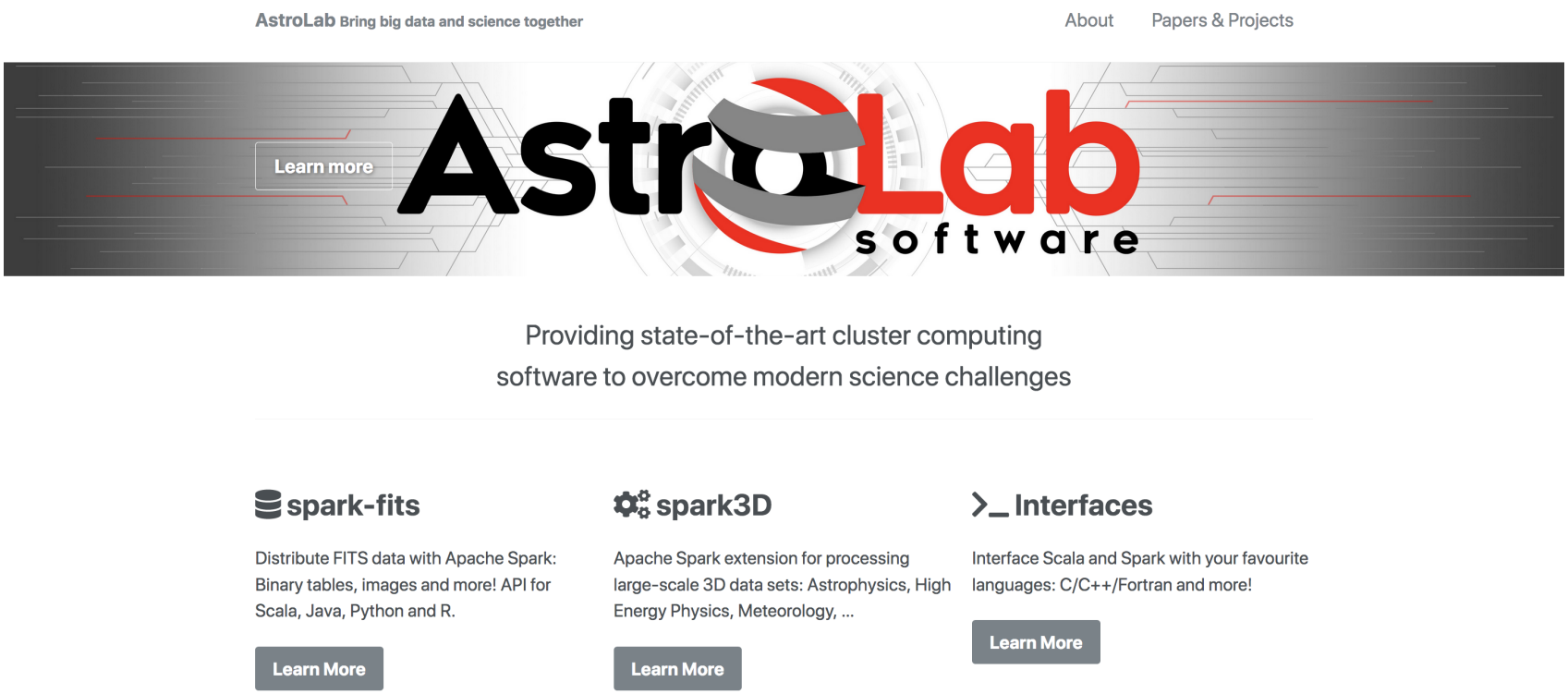
Goal: provide a realistic experimental environment, integrated in the cloud@VirtualData platform, to overcome big data challenges.



* Spark@u-psud Aug 2015, Spark@VirtualData Feb 2016

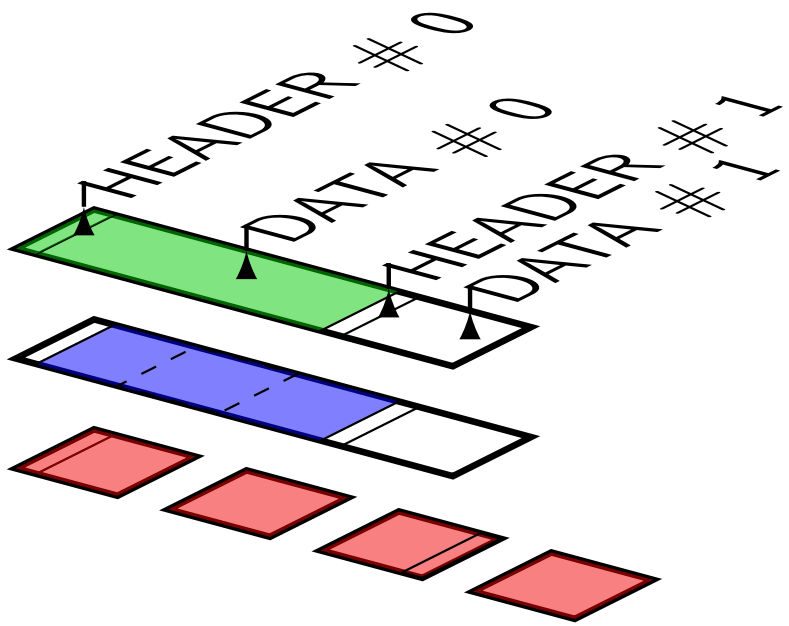
AstroLab Software

- AstroLab Software is a project started at LAL to promote the development and the use of big data solutions in astrophysics.



spark-fits: Interfacing FITS format with Spark

- FITS: standardized for more than 30 years (backward compatible).
- spark-fits is a native Spark connector, similar to what spark-root does.
- Written in Scala. API for Scala, Python, Java and R.
- Support FITS data source for Spark SQL and DataFrames.




```
// Define a DataFrame with the
// data from the first HDU
val df = spark.read
         .format("fits")
         .option("hdu", 1)
         .load("hdfs://...")
```

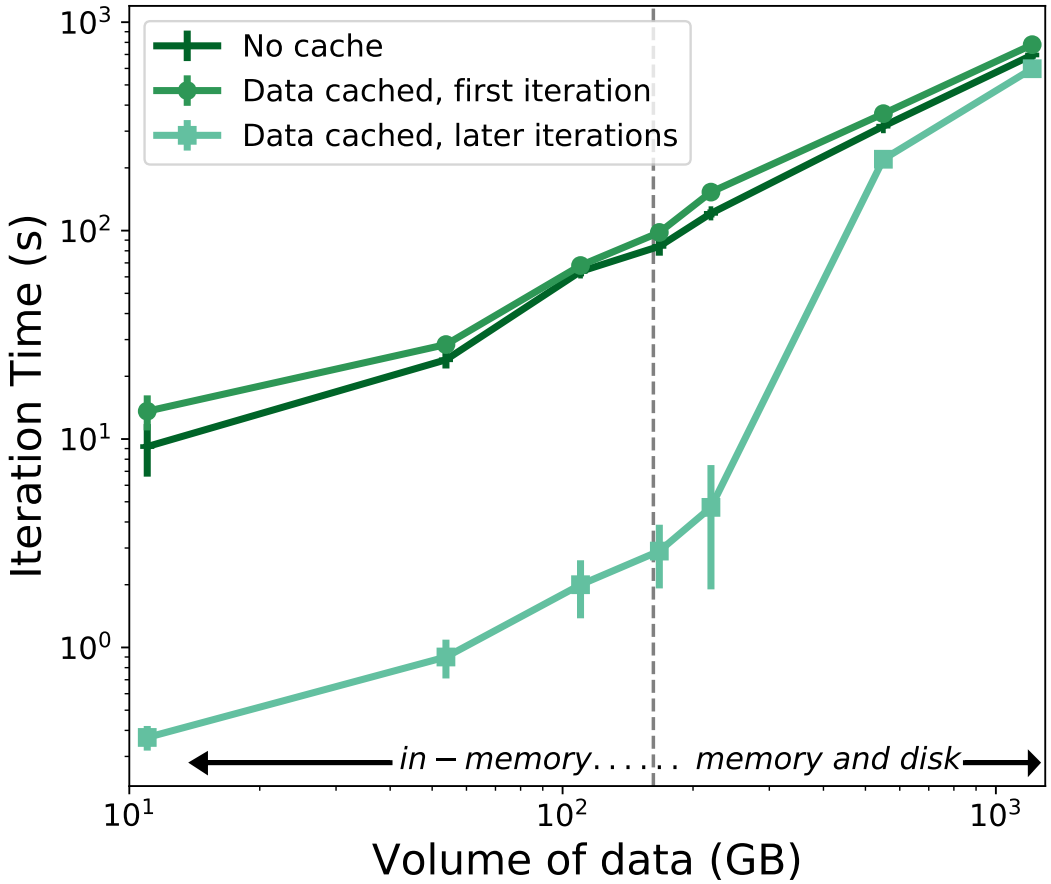
`spark-fits`: Interfacing FITS format with Spark

Some problems encountered

- There is no FITSIO in Scala. Solution: write a new library to handle FITS format in Scala (and make it serializable!).
- FITS organised by blocks (header+data): all executors must know the header. Solution: register it in the `org.apache.hadoop.conf.Configuration`.
- Scala not so much used in the Astro community. Solution: capitalize on the Apache Spark tools to have API for Python, Java, and R.

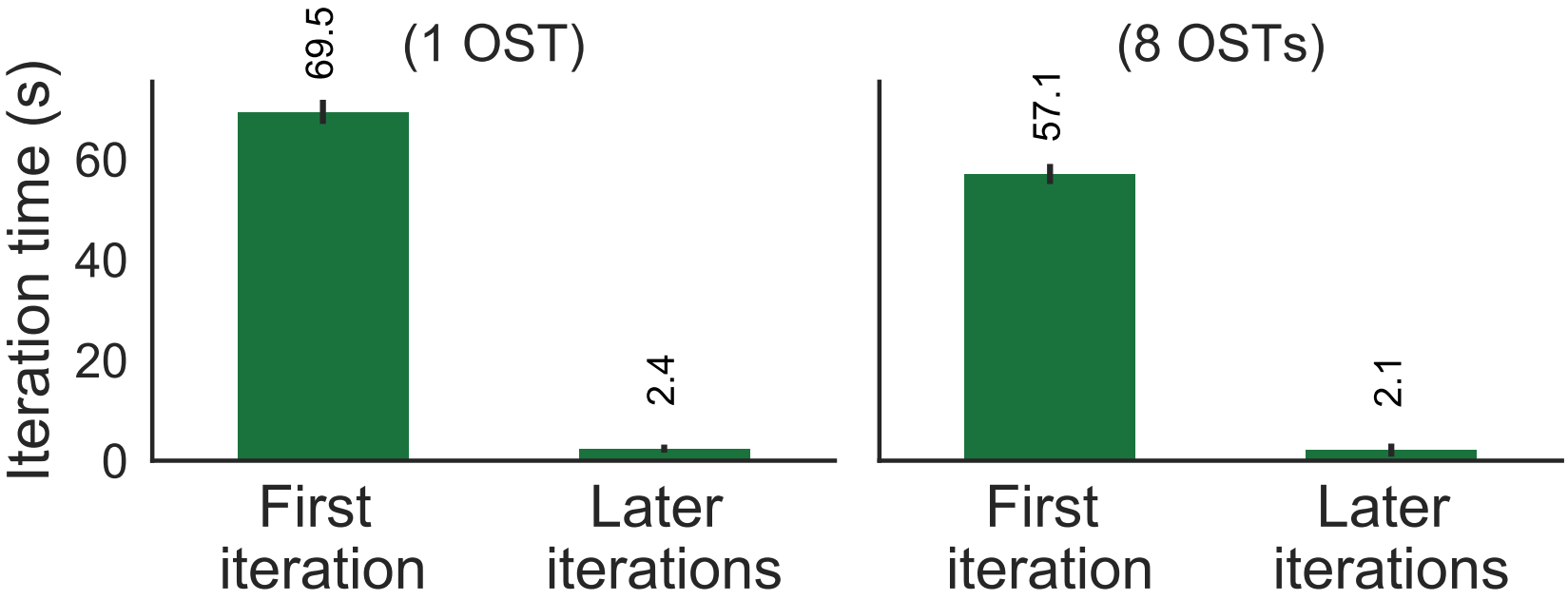
spark-fits: Interfacing FITS format with Spark

- Proof-of-concept: Peloton et al. (arXiv:1804.07501).
- Most benchmarks run on Spark@LAL (153 cores, 150 GB memory for cache).



spark-fits: Interfacing FITS format with Spark

- Apache Spark on state-of-the-art HPC infra (DFS: Lustre): Distribute up to 1.2 TB in 1 minute over 1280 cores ($\sim$ 20 GB/s), and then manipulate them in few seconds.

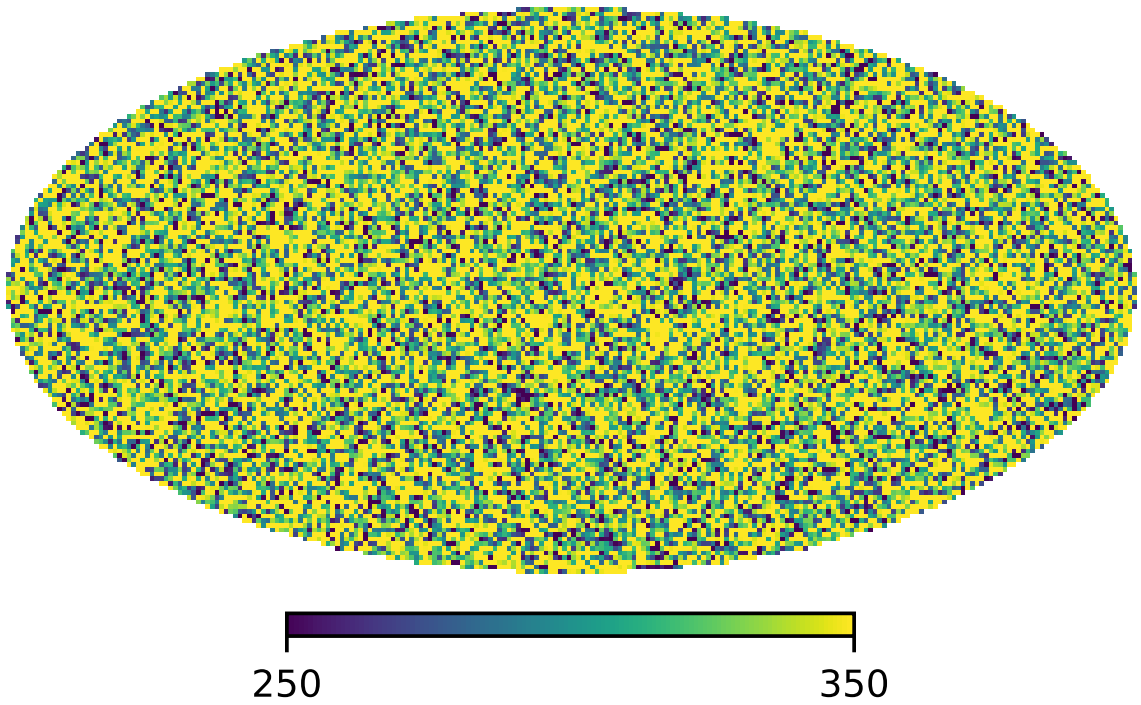


1.2 TB, 1280 cores (@NERSC)

Apache Spark for astronomers

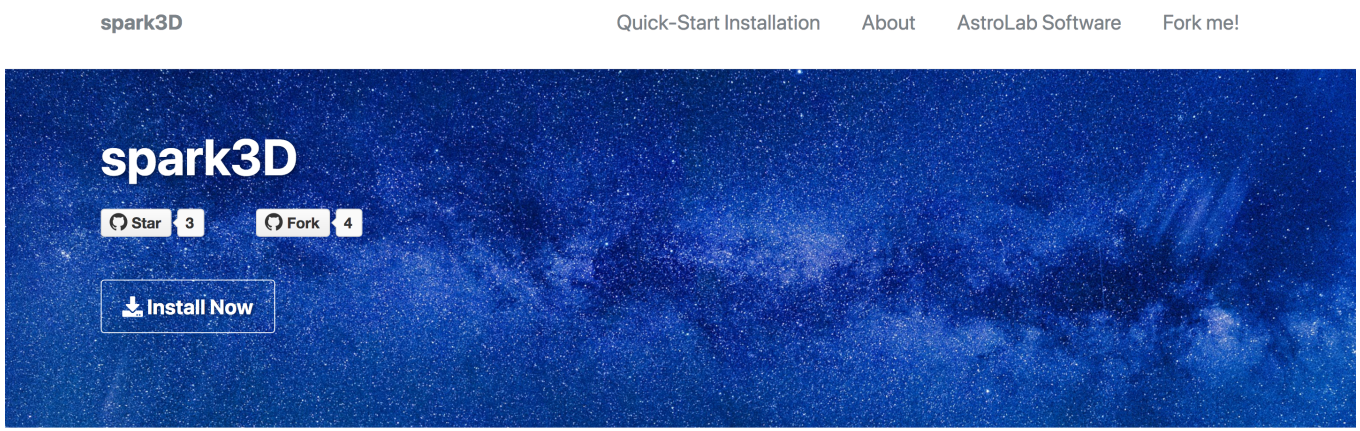
- Apache Spark for astronomers: Plaszczynski et al. (arXiv:1807.03078).
- Simulate 10 years of LSST data (112 GB, 6.10^9 galaxies); analyze them interactively: statistics, histograms and tomography within seconds.

$z \in [0.61, 0.82]$



spark3D: Manipulating 3D data sets

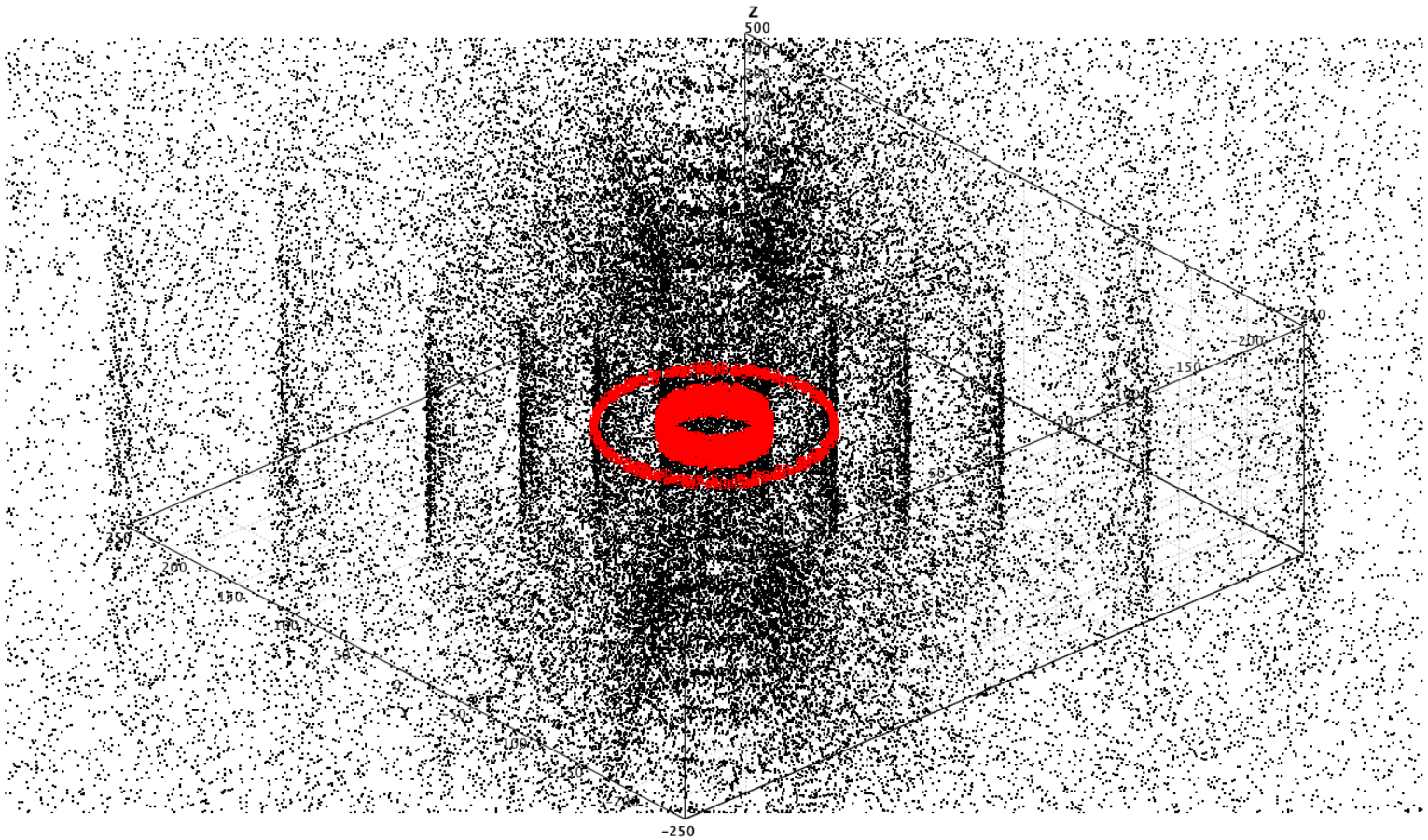
- Develop methods specific to 3-dimensional data sets.
  - Google Summer of Code 2018 project (CERN-HSF).
  - Space partitioning, cross-match, neighbour search, queries, ...
  - Optimized for large data sets, and support a wide range of data sources: FITS, ROOT ($\leq$6.11*), HDF5 + all the v2 Spark data sources (CSV, JSON, TXT, parquet, ...)



*forward-compatibility issue. thx @vkhristenko.

spark3D: Manipulating 3D data sets

K nearest neighbours* using TrackML challenge data



* Preliminary. Brute force method (distributed) takes $<$ 1s for 100,000 points and K=5000. For larger dataset ($>$ 10^9 points) need to fix substantial increase in wait time during garbage collection... Re-partitioning (Tree) and indexing to come. ** z axis has not the same scale as x and y axes.

Binding or native implementation?

Typical question: dev should be made in Scala or Python?

- For simple tasks, the Scala and Python API perform the same.
- For complex tasks, the Scala API often outperforms the Python one. But the Python scientific ecosystem is much wider than the Scala one.

$\rightarrow$ To me, which language to use in priority remains an open question :-(

Beware though....

- Big data is also about handling the IO correctly in a code.
- Interfacing outside world code can turn out to be a nightmare just because of the way IO is handled internally... and OO is not always your best friends!

Interfaces: Bring Scala and others together

- R&D around language binding.
- Spark works best with Scala under the hood, but scientific libraries mostly in imperative languages rather declarative...
- API already exist for Python, Java, and R, but one would like ideally to be able to use of our beloved C/C++/Fortran libraries within Spark.
- Currently, this is more testing what is available (e.g. `JEP`, `JNA`) rather than developing new tools from scratch.

END

Thanks

Spark cluster: current numbers

Main cluster

- CentOs7
- 1+1+9 virtual machines (openstack)
- 18+18+162 physical cores
- IO bandwidth of 2.3 GB/s
- 35 TB of storage (volume cinder HDFS)
- 300 GB of memory

Overview of components

- DFS: HDFS (Hadoop 2.8.4) on Ceph.
- Cluster manager: YARN
- Framework: Apache Spark (2.3.0)
- Monitoring: Ganglia 3.7 (+Spark/YARN UI)
- Alert: Monitoring linked to Slack (webhook)

Current + future plans (and problems to solve...)

- Multi-tenant system:
  - How to decouple efficiently data from computation?
  - batch, interactive (SWAN?)
  - In test currently: Hue (Hadoop User Experience)
- Finer control and monitoring of the resources.
- DFS: other as S3?
- Hosting vs streaming the data? What would be the best choice for experiments (and us)?
- Apache Kafka?
- High availability: Zookeeper?

Links

- AstroLab Software:
  https://astrolabsoftware.github.io/
- `spark-fits`:
  https://astrolabsoftware.github.io/spark-fits/
- `spark3D`:
  https://astrolabsoftware.github.io/spark3D/