Docker @ CDS

André Schaaff[1], François-Xavier Pineau[1], Gilles Landais[1], Laurent Michel[2]

[1]Centre de Données astronomiques de Strasbourg, [2]SSC-XMM-Newton

Paul Trehiou

Université de technologie de Belfort-Montbéliard

IVOA, Shanghai, 14-19/05/2017

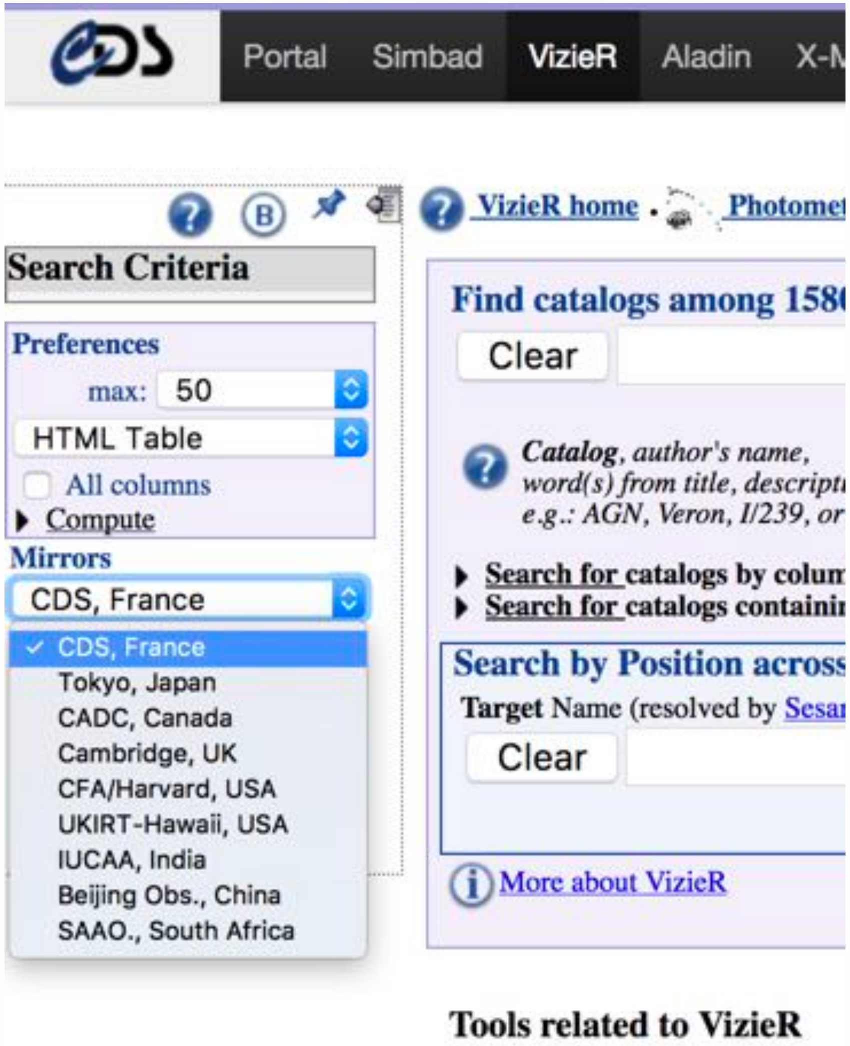
CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

Asterics Astronomy ESFRI & Research Infrastructure Cluster

H2020-Astronomy ESFRI and Research Infrastructure Cluster (Grant Agreement number: 653477).

Docker for VizieR deployement

- VizieR is deployed on several mirrors
- Hosts with different Linux distributions, kernels, etc.
- Docker as a solution to deploy “quick and easy” ?



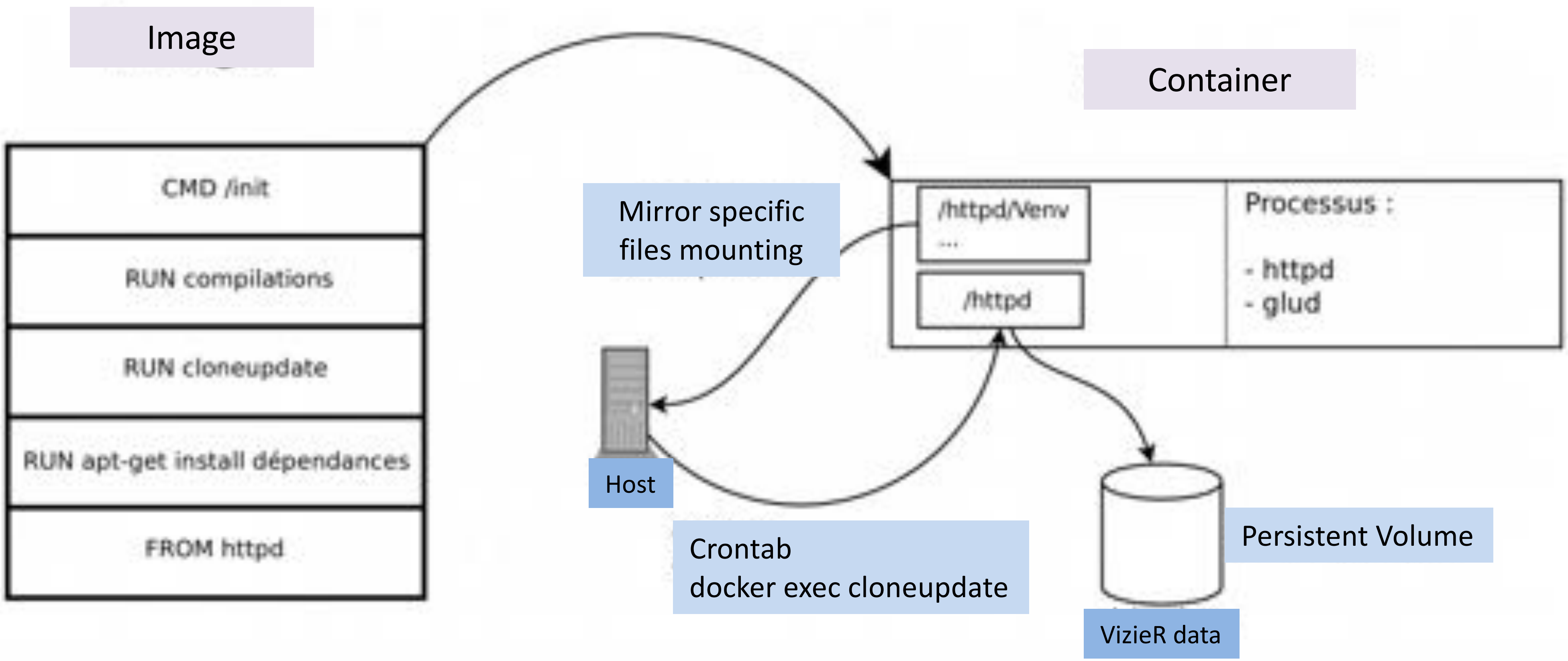
VizieR mirror update process

- Typical installation of VizieR
  - Dependencies and CGI scripts transfer through Rsync and scp
  - Dependencies installation with the package manager
  - Compilation of the dependencies only available as sources (developed at CDS)
  - CGI files copy and Apache configuration
  - Apache start

To a VizieR Docker image…

- Prototyping following the previous process
  - Resolution of missing packages (like gcc, make, rsync, .., not present in the Apache image)
- To an optimized version
  - Sources directly from CVS
  - CGI scripts and static files in a Docker volume (between the host and the container), these files will be updated through scripts (executed at each container start and via a cronjob service (on the host))
  - Path corrections
  - Also use of Portainer (Simple management UI)

VizieR & Docker at the End



And now

- Used in a first step inside CDS
  - To test Docker use on a mid-term basis
  - To install local “VizieR” to test it before update
- Needs discussion and agreement with at least one host (in a first step) to start to use it “in production” all over the world !

Experiments with Spark - status

André Schaaff, François-Xavier Pineau
Centre de Données astronomiques de Strasbourg

Osman Aidel
Centre de Calcul – IN2P3 Villeurbanne

Noémie Wali, Paul Trehiou
Université de technologie de Belfort-Montbéliard

IVOA, Shanghai, 14-19/05/2017

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

Asterics
Astronomy ESFRI & Research Infrastructure Cluster

H2020-Astronomy ESFRI and Research Infrastructure Cluster (Grant Agreement number: 653477).

Outline

Apache Spark

Use case & data

Test beds, experiments and what we learned

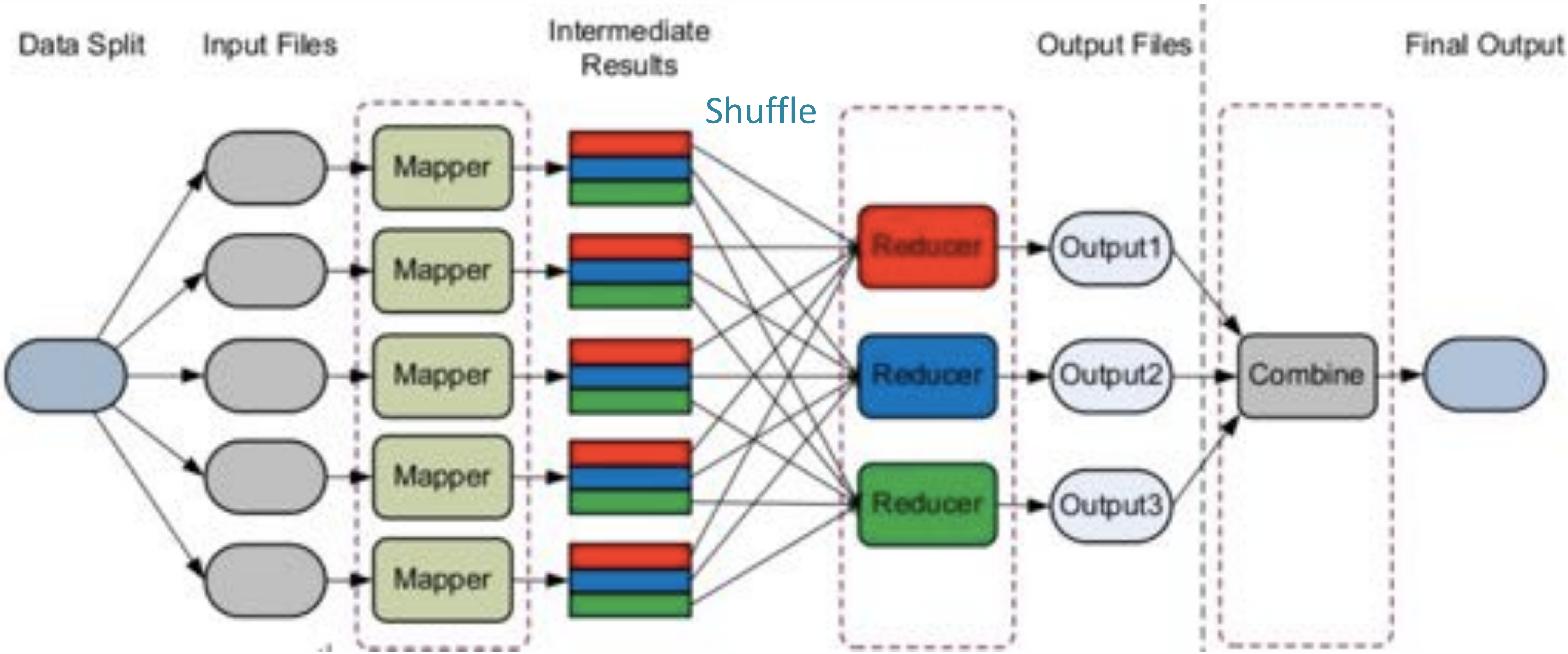
Perspectives

Apache Spark

- "Apache Spark is a cluster computing platform designed to be fast and general purpose."

- It extends the MapReduce model to support more types of computations (interactive queries, stream processing, etc.) and it offers APIs for Scala, Java, Python, R,...

MapReduce



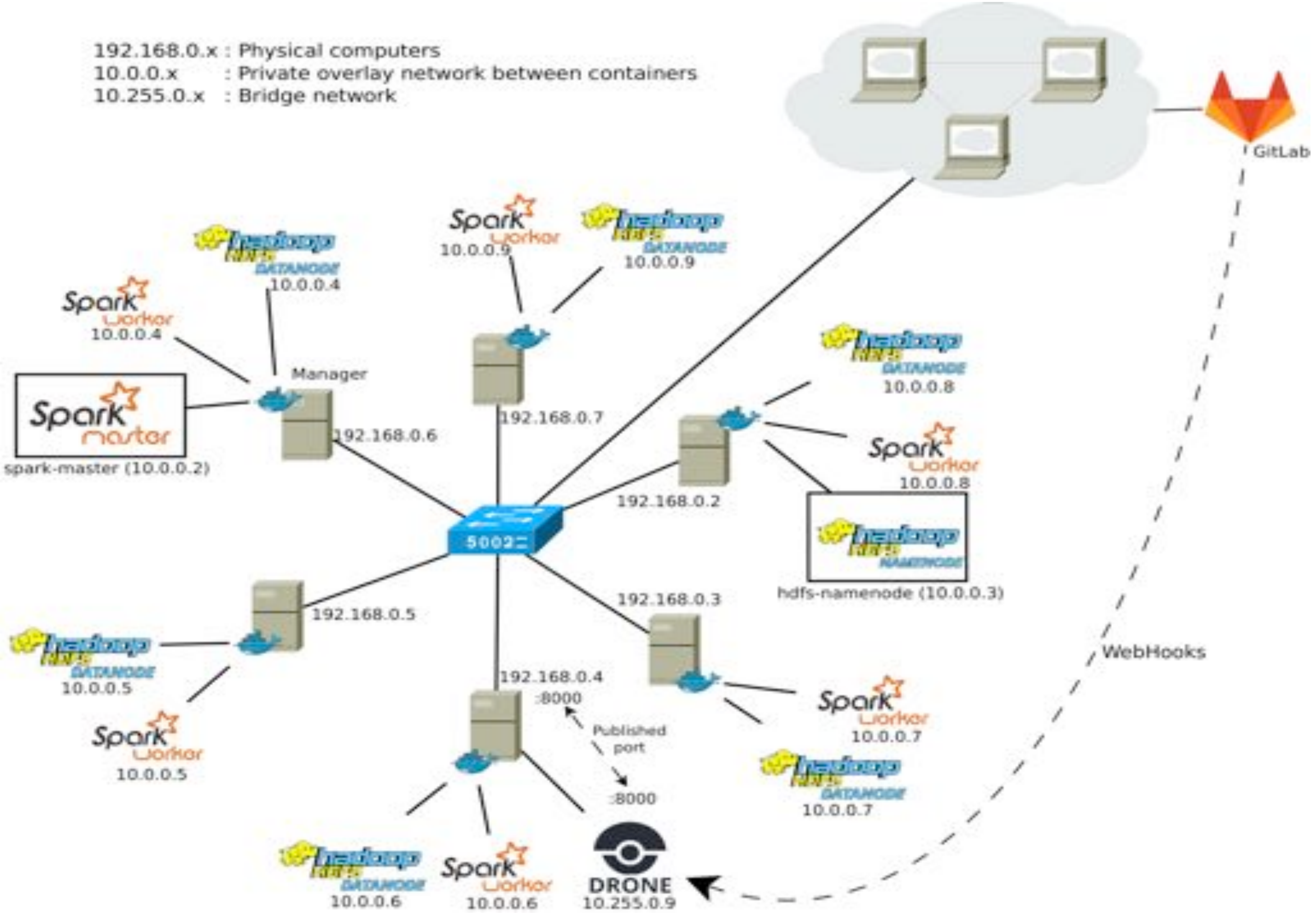
Credit: G. Fedak, INRIA

Apache Spark, so quick ?

- Computations in memory (as much as possible, otherwise pilling to the disks)
- Introduction of data models
  - RDD (Resilient Distributed Datasets)
    - Immutable distributed collection of elements
    - Operations: Transformations (map, filter, etc.), Actions (reduce, count, etc.)
  - Datasets to represent tabular data, queryiable via SQL
- It uses mainly Hadoop Distributed File System (HDFS).

Other **technical aspects**

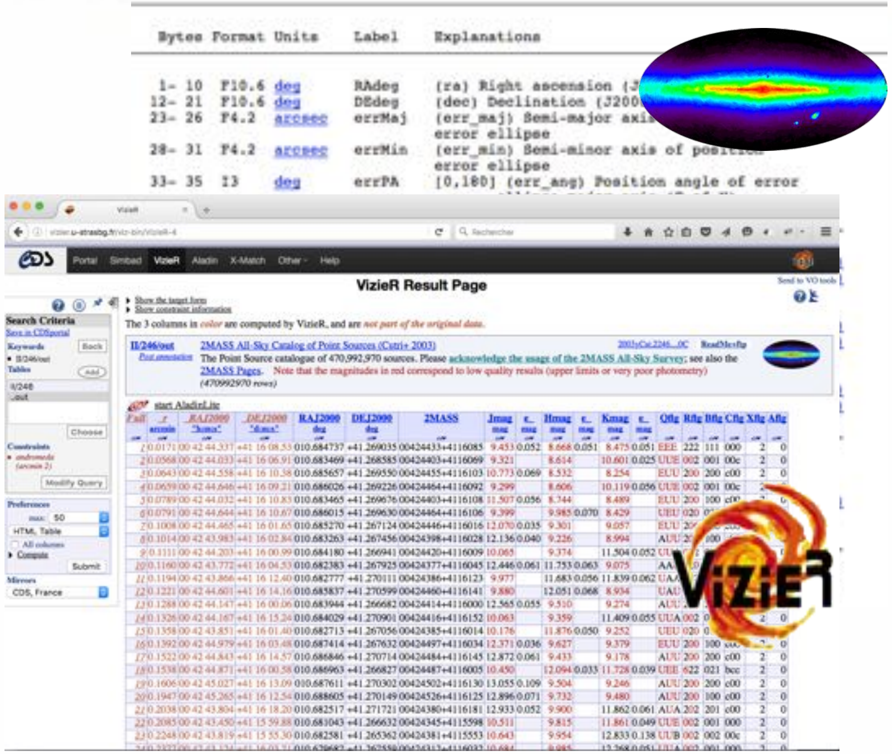
- Introduction of Docker (components) and Drone (continuous integration) to "automate" the deployment process and to focus mainly on the development side. It is becoming easy to migrate to external resources when needed.
- Use of Scala which is native in Spark (a part of the Java API is "experimental").

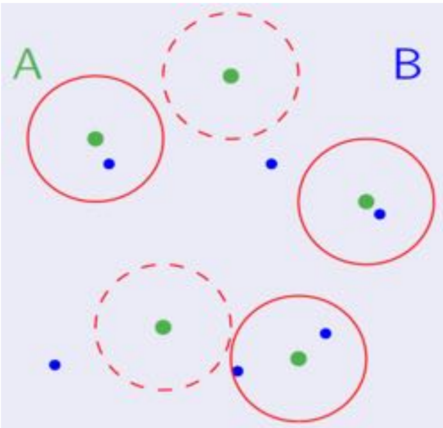


192.168.0.x : Physical computers
10.0.0.x : Private overlay network between containers
10.255.0.x : Bridge network
GitLab
WebHooks
Spark worker 10.0.0.4
10.0.0.4
Manager
spark-master (10.0.0.2)
192.168.0.6
10.0.0.9
10.0.0.9
192.168.0.7
10.0.0.8
Spark worker 10.0.0.8
192.168.0.2
hdfs-namenode (10.0.0.3)
192.168.0.3
192.168.0.5
10.0.0.5
Spark worker 10.0.0.5
192.168.0.4
:8000
Published port
:8000
Spark worker 10.0.0.7
10.0.0.7
10.0.0.6
Spark worker 10.0.0.6
DRONE
10.255.0.9

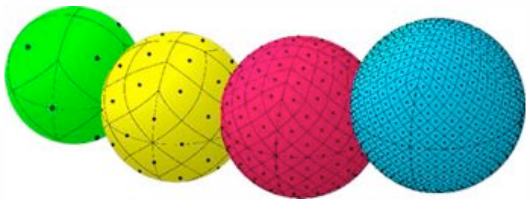
Use case & data

- The "cross-match" of (large) source catalogues.
- Examples:
  - 2MASS[1], 470,992,970
  - SDSS[2] DR9, 469,053,874





Full sky: all the sources
A cone: only the sources which are at a certain angular distance from a given position
A HEALPix cell

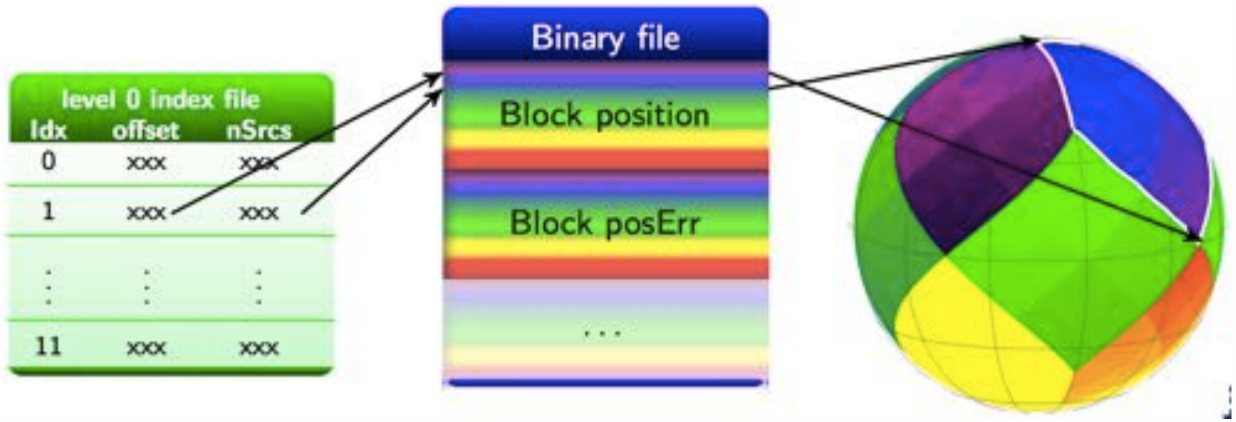


Credits: http://healpix.jpl.nasa.gov/

Fuzzy join between 2 tables (A and B) of several hundred millions of data

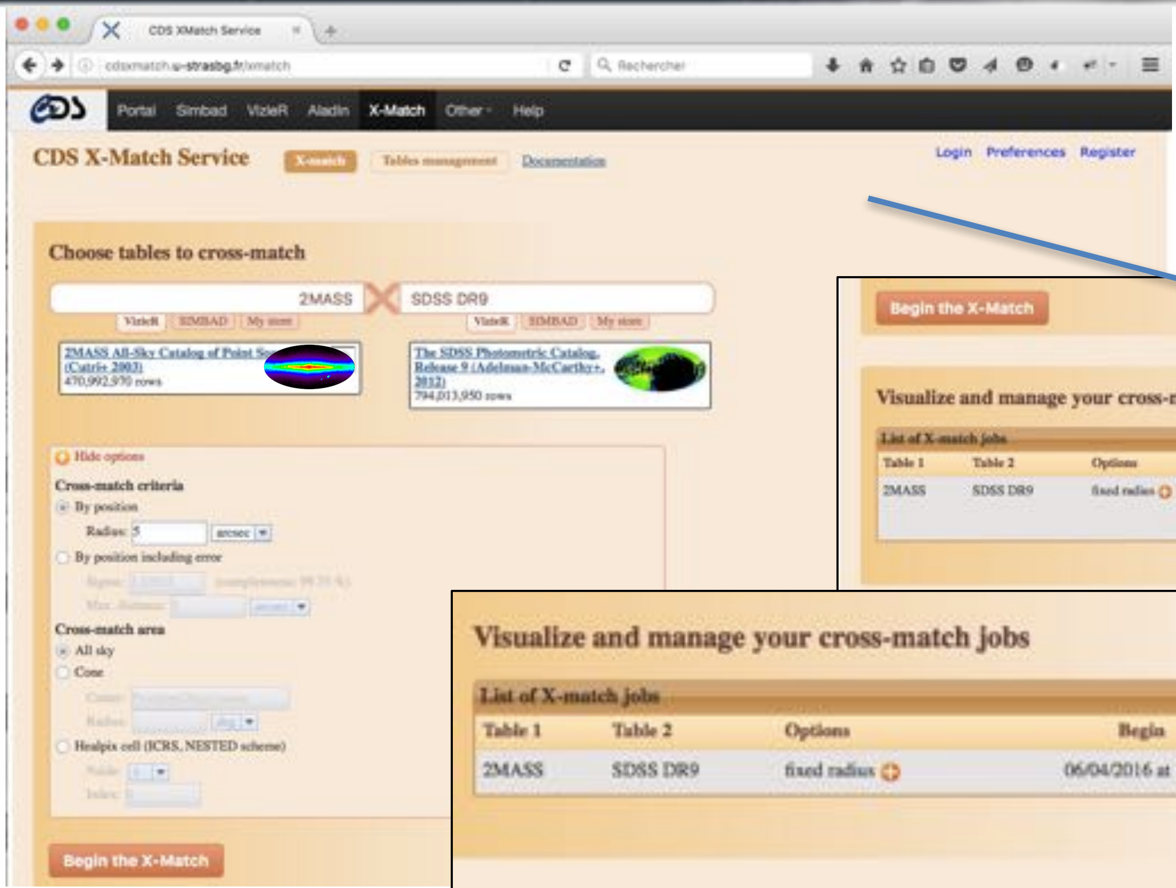
Not distributed but…

- organised and stored on one server (2x10 cores, 64GB, 12TB (15k tours))



The sky is cut into diamonds of the same size, pixels, each source or sky object is a numbered pixel.

Illustration: X-Match frontend

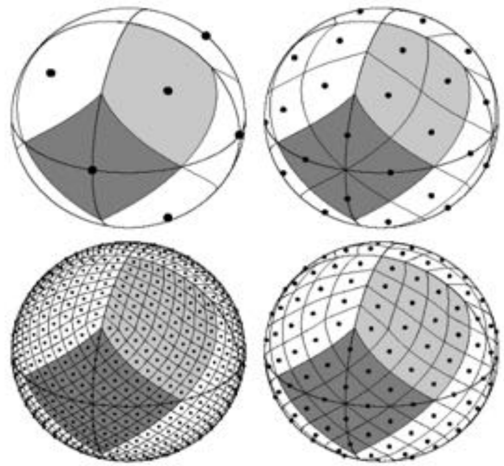


X-Match of
2MASS & SDSS DR9
(over 10,000 catalogues + own catalogue upload)

GAIA DR1 X SDSS DR9 (1 arcsec) in 17’ (100.10^6 matchs, 34 GB)

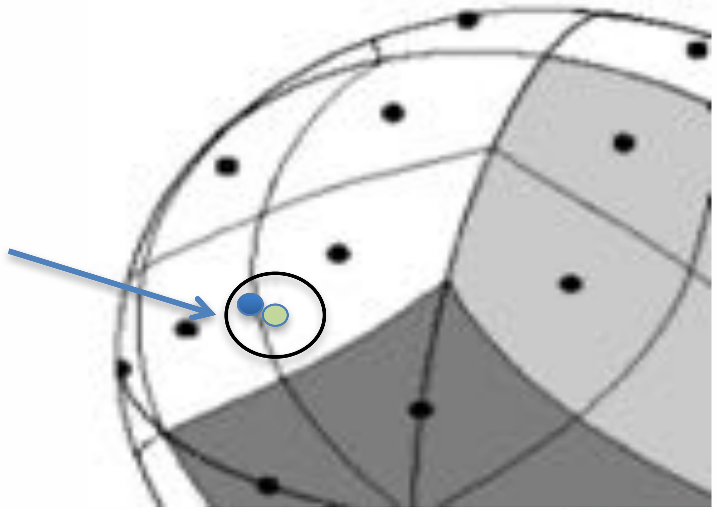
Illustration

- A X-Match implementation in MapReduce, couples (Key = pixel number, Value)



HEALPix sky cutting

- Side effects
  - Fuzzy join
  - Source duplication in the neighbour cells if needed



Credits: HEALPix – arXiv:astro-ph/0409513

☐ Test beds: hardware & software

- External resources (Experiment 1)
  - 12 nodes, 4 cores, 32GB, Raid 2*2TB, Ubuntu
  - Configuration was defined "ad hoc" and low cost
- Collaboration with IN2P3 (Experiment 2)
  - 9 nodes (only one VM per physical server, CentOS / OpenStack, remote data storage), 24 threads (-> 216 threads), 64GB (-> 576 GB) -> possible X-Match of billion sources
- Internal resources to prototype, (Experiment 3)
  - 6 nodes (4 cores, 16GB, 1TB + 500GB SSD), Ubuntu
- Apache distributions of Spark and Hadoop, Java, Scala, Docker, Drone, …

Experiment 1 (12 nodes)

- Input data (SDSS DR7 (primary sources) and 2MASS): 54GB and 58GB file size; 357 175 411 and 470 992 970 elements
- Output data: 49 208 820 elements

X-Match service reference time was: 10 minutes

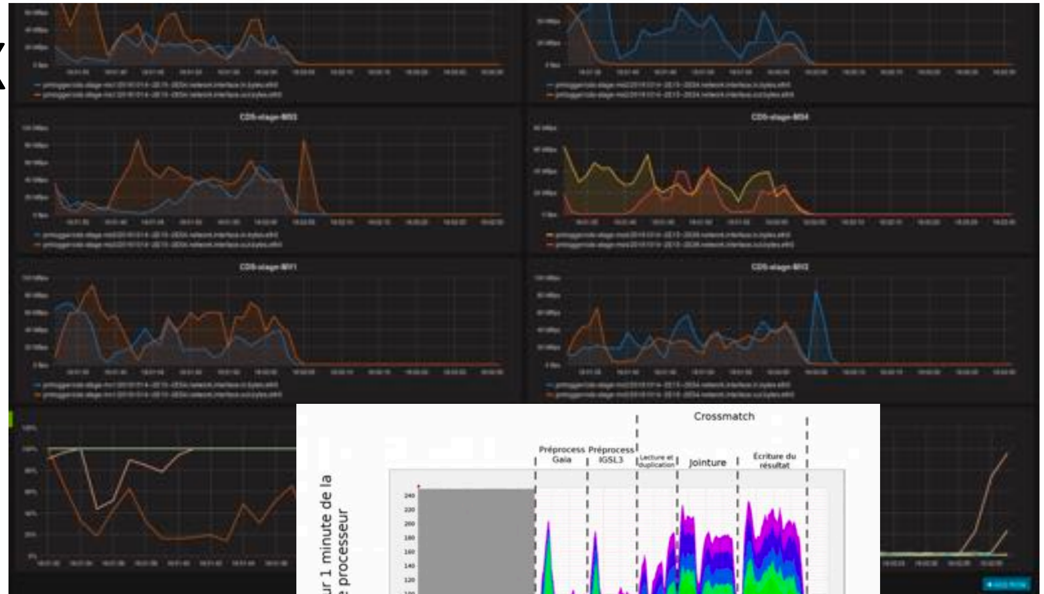
Cross-Match (source duplication done in phase 2 with all the data as output)					
HDFS block size= 128MB for the input files ; sdss7.csv and t 2mass.csv replicated 2 times					
HashPartitioner	60 partitions				
HDFS output files size	32MB				
Number of nodes Spark/HDFS	5	7	9	10	11
Phase 1: prepare	**23,0**	**16,0**	**14,0**	**14,0**	**13,0**
mapToPair (sdss7.csv)	5,1	4,9	4,9	4,8	4,7
saveAsHadoopFile (sdss7.bin)	5,7	2,7	2,0	2,3	1,5
mapToPair (2mass.csv)	5,7	5,2	5,2	5,1	5,0
saveAsHadoopFile (2mass.bin)	6,5	3,6	1,9	1,6	1,4
Phase 2: join	**31,0**	**21,0**	**13,0**	**11,0**	**9,9**
mapToPair (sdss7.bin)	7,2	4,7	3,5	3,0	[illegible]
flatMapToPair (2mass.bin)	11,8	8,3	5,5	4,9	4,3
saveAsTextFile (crossMatch_D.txt)	12,0	7,6	3,4	2,4	2,3
TOTAL	**54,0**	**37,0**	**27,0**	**25,0**	**22,9**

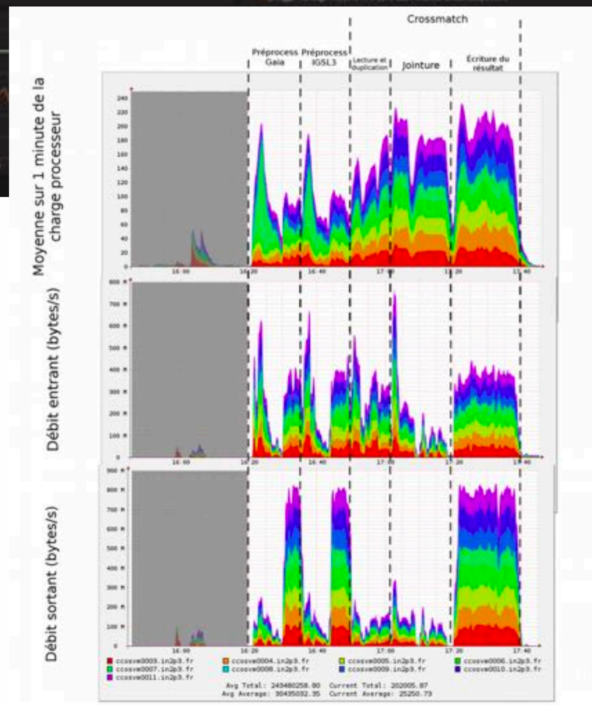
What we have learned

- Time was similar to the X-Match service from 11 nodes
- Keys common to 2 RDDs are not necessarily on the same node
  - It implies a transfer overhead between the nodes during the join => impact on the performances
  - We had clearly a bottleneck in the join phase ("shuffle")
  - "block affinity groups" is an on-going work at Apache.
- We spent time on the "data co-location"
- We found a solution to do it "manually" via Python scripts (=> Experiment 3, with SSD).

Experiment 2: IN2P3 cluster

- Gaia (1.1 billion sources) X IGSL3 (1.2 billions)
- 1.6 billion associations in 15 minutes (30 minutes for the production X-Match Server)*
- Debriefing: Bottleneck is
  - Network for Loading phase
  - Mainly CPU during the xmatch phase (can be improved)





* Rebuild Join not included

Experiment 3: SSD

- Network 1GB/s, SSD 400-450 MB/s
- We were able to experiment the Python Script to move the data blocks « manually » (=> manual colocation of data)
  - With the HD, no gain (Network and HDD have a similar speed)
  - With the SSD, a gain of 30% on the loading phase, as theoretically predicted (SSD faster than network)
  - And similar performances in or out of Docker

Perspective and conclusion

- Apache Spark quick to install, easy to use for common tasks
- But not easy to understand what happens, how it works when you have particular use cases
- Real interest in several communities
- Ongoing experiments, probably with clouds
- Docker use will continue with an application to the X-Match service

Links

- Apache Spark, http://spark.apache.org/
- Apache Hadoop, http://hadoop.apache.org/
- Spark : Cluster Computing with Working Sets, Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, University of California, Berkeley, http://static.usenix.org/legacy/events/hotcloud10/tech/full_papers/Zaharia.pdf
- Optimizing Shuffle Performance in Spark, Aaron Davidson, Andrew Or, UC Berkeley, http://www.cs.berkeley.edu/~kubitron/courses/cs262a-F13/projects/reports/project16_report.pdf
- Resilient Distributed Datasets : A Fault-Tolerant Abstraction for In-Memory Cluster Computing, Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica, University of California, Berkeley, https://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf
- JavaSpark Api, http://spark.apache.org/docs/latest/api/java/
- HEALPix, http://healpix.jpl.nasa.gov/

H2020-Astronomy ESFRI and Research Infrastructure Cluster (Grant Agreement number: 653477).