Hadoop & Spark, « cross-match » of source catalogues

André Schaaff, François-Xavier Pineau
CDS, Centre de Données astronomiques de Strasbourg
Noémie Wali
UTBM, Université de technologie de Belfort-Montbéliard

Special thanks to Julien Nauroy, Université de Paris Sud

IVOA Cape Town, GWS session 2

Outlines

Context

Motivation

The data and the « cross-match » service

Test beds

Study phase

(current) Results

Conclusion and perspectives

Context

A continuous exploration of new technologies, especially in the « Big Data » field

During a Sydney GWS session:

- Hadoop (HDFS), Spark presentation and use cases
- Discussion
- In the frame of "bringing the computation to the data" ?

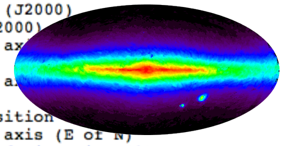
Motivation (of this study)

- We whished to evaluate what Hadoop / Spark could bring by studying an appropriate use case, the « cross-match » of source catalogues:
  - Improvement of the existing service, the up to scale capability (data volumes, hardware, deployments, etc.)
  - For which cost (budget, manpower, performances (better ?))

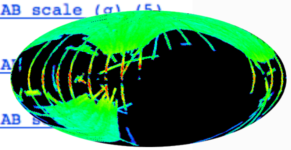
Which data ?

- Data from source catalogues
- Examples (number of sources):
  - 2MASS[1], 470,992,970
  - SDSS[2] DR9, 469,053,874

Bytes	Format	Units	Label	Explanations
1- 10	F10.6	deg	RAdeg	(ra) Right ascension (J2000)
12- 21	F10.6	deg	DEdeg	(dec) Declination (J2000)
23- 26	F4.2	arcsec	errMaj	(err_maj) Semi-major ax error ellipse
28- 31	F4.2	arcsec	errMin	(err_min) Semi-minor a error ellipse
33- 35	I3	deg	errPA	[0,180] (err_ang) Position ellipse major axis (E of N)
37- 53	A17	---	2MASS	(designation) Source designation (1)



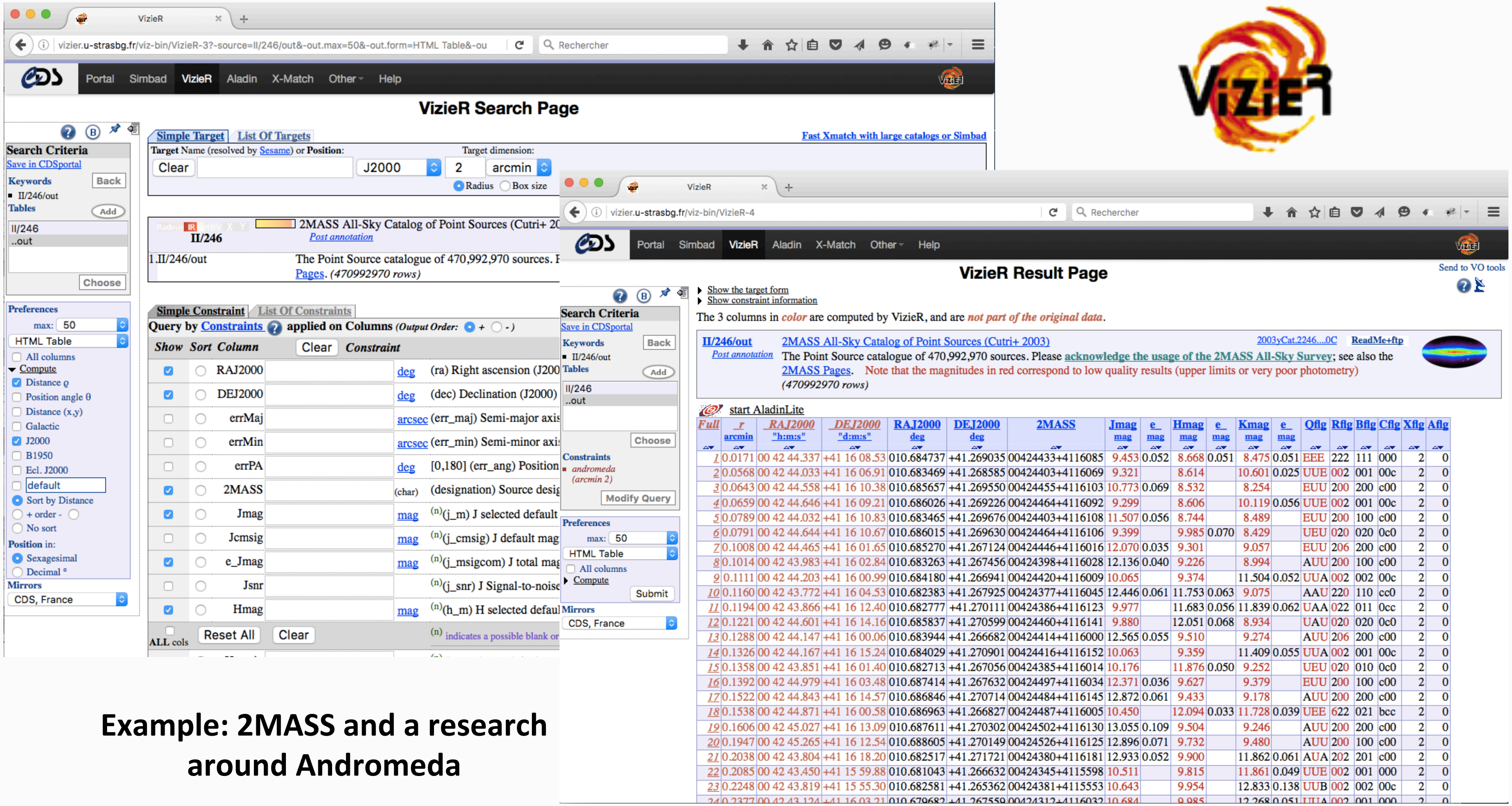
Bytes	Format	Units	Label	Explanations
1	I1	---	mode	[1,2] 1: primary (469,053,874 sources), 2: secondary (324,960,076 sources).
2	A1	---	q_mode	[+] '+' indicates clean photometry (105,969,748 sources with mode 1+)
3	I1	---	cl	Type (class) of object (3=galaxy, 6=star) (1)
5- 23	A19	---	SDSS9	SDSS-DR9 name, based on J2000 position
24	A1	---	m_SDSS9	[*] The asterisk indicates that 2 different SDSS objects share the same SDSS9 name
27- 47	A21	---	SDSS-ID	[0-9 -] SDSS object identifier (2)
49- 67	I19	---	objID	SDSS unique object identifier (2)
70- 84	A15	---	Sp-ID	Spectroscopic Plate-MJD-Fiber identifier (7)
86-104	I19	---	SpObjID	Pointer to the spectrum of object, or 0 (7)
106-124	I19	---	parentID	Pointer to parent (if object deblended)
126-141	A16	---	flags	[0-9A-F] Photo Object Attribute flags (3)
143-150	A8	---	Status	[0-9A-F] Hexadecimal status (4)
153-162	F10.6	deg	RAdeg	Right Ascension of the object (ICRS) (ra)
163-172	F10.6	deg	DEdeg	Declination of the object (ICRS) (dec)
174-178	F5.3	arcsec	e_RAdeg	Mean error on RAdeg (raErr)
180-184	F5.3	arcsec	e_DEdeg	Mean error on DEdeg (decErr)
186-194	F9.4	yr	ObsDate	Mean Observation date
196	I1	---	Q	[0/5] Quality of the observation (0=unknown): 1=bad 2=acceptable 3=good 4=missing 5=hole (6)
198-203	F6.3	mag	umag	? Model magnitude in u filter, AB scale (u) (5)
204	A1	---	---	[:]
205-209	F5.3	mag	e_umag	? Mean error on umag (err_u)
211-216	F6.3	mag	gmag	? Model magnitude in g filter, AB scale (g) (5)
217	A1	---	---	[:]
218-222	F5.3	mag	e_gmag	? Mean error on gmag (err_g)
224-229	F6.3	mag	rmag	? Model magnitude in r filter, A
230	A1	---	---	[:]
231-235	F5.3	mag	e_rmag	? Mean error on rmag (err_r)
237-242	F6.3	mag	imag	? Model magnitude in i filter, AB s
243	A1	---	---	[:]
244-248	F5.3	mag	e_imag	? Mean error on imag (err_i)
250-255	F6.3	mag	zmag	? Model magnitude in z filter, AB scale (z) (5)
256	A1	---	---	[:]
257-261	F5.3	mag	e_zmag	? Mean error on zmag (err_z)



Example of a ReadMe file associated to source catalogues available through the VizieR service

[1]2MASS, Two Micron All Sky Survey,
[2]SDSS, Sloan Digital Sky Survey

Which data ? (in VizieR)

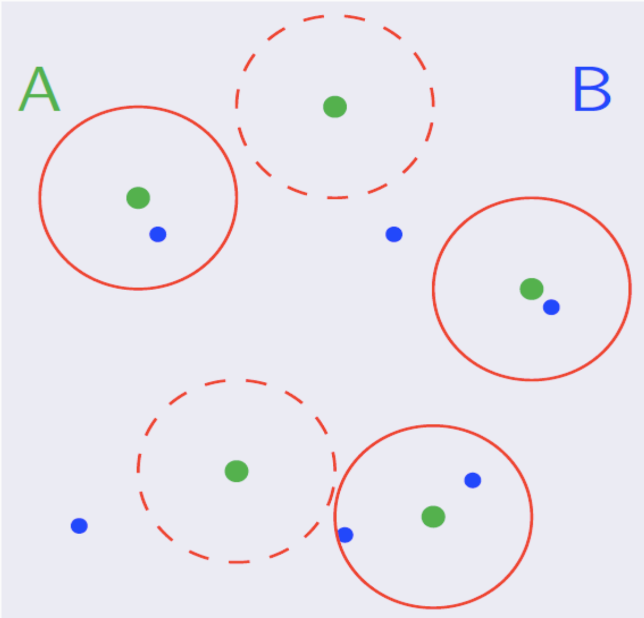


Example: 2MASS and a research around Andromeda

…and the CDS « cross-match » service

- The « cross-match » service does a cross correlation of sources between (very) large catalogues (current size: 10^9).

Fuzzy join between 2 tables of several hundred millions of lines

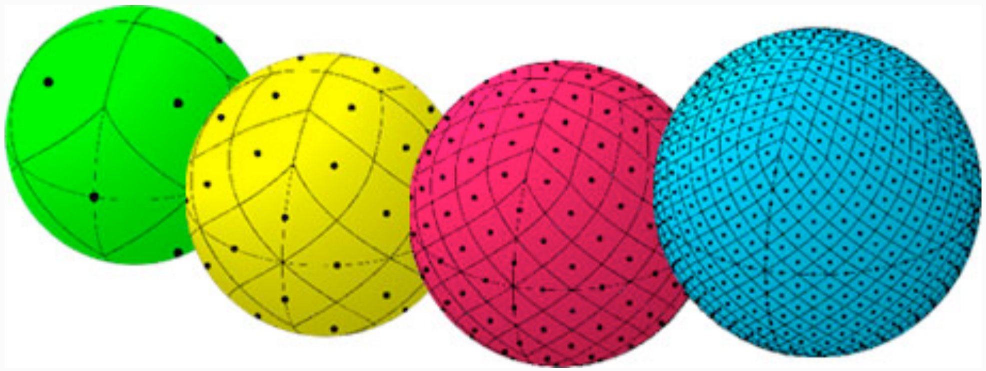


…and the CDS « cross-match » service (2)

- It is possible to do it with catalogues proposed by the CDS but also to upload your own data (a table with positions) to cross-match it with these catalogues.
- It is based on optimized developments and implemented on a well-sized server (enabling on-line use).

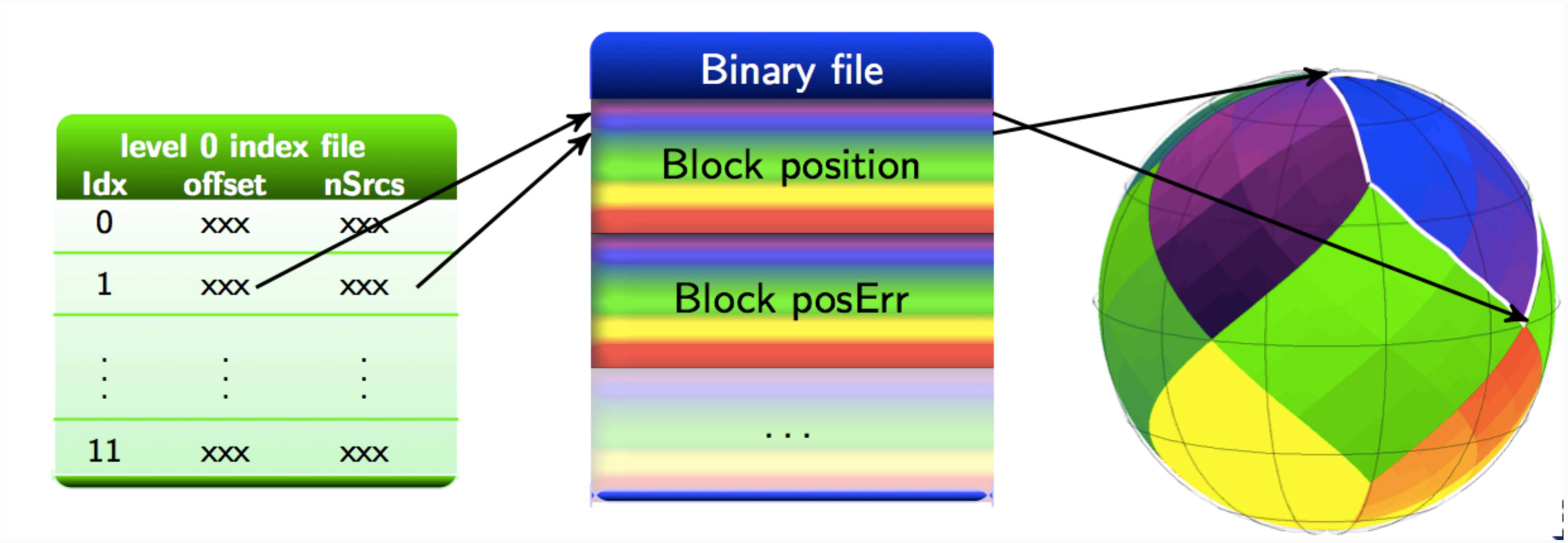
…and the CDS « cross-match » service (3)

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



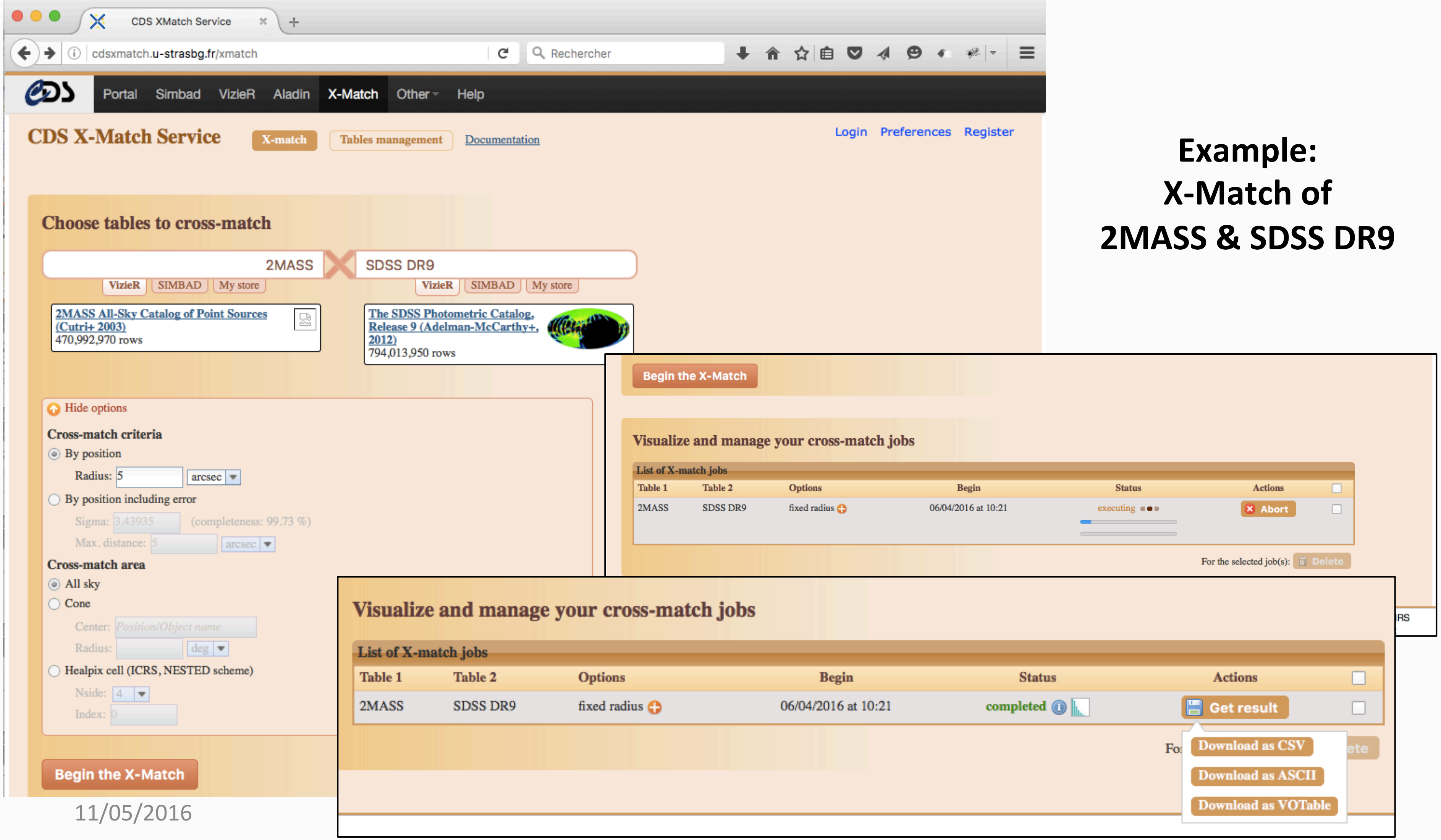
…and the CDS « cross-match » service (4)

- Data is not distributed but organised and stored on one server



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

Illustrations



Example:
X-Match of
2MASS & SDSS DR9

Illustrations (2)

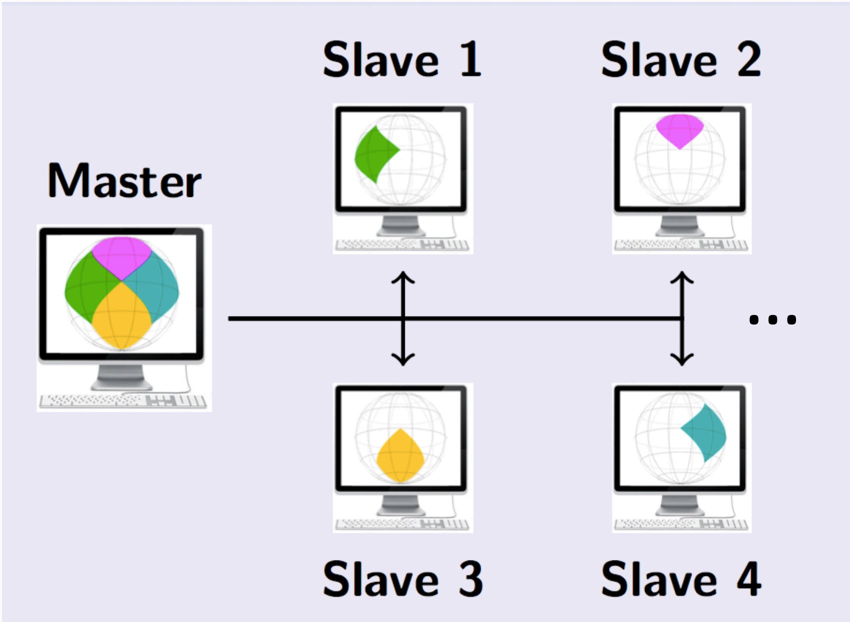
**Example:
An "excerpt"
of the result
in CSV**

Téléchargements — vi 1459930879752A.csv — 108×36


```
angDist,2MASS,RAJ2000,DEJ2000,errHalfMaj,errHalfMin,errPosAng,Jmag,Hmag,Kmag,e_Jmag,e_Hmag,e_Kmag,Qfl,Rfl,X,
MeasureJD,SDSS9,RAdeg,DEdeg,errHalfMaj,errHalfMin,errPosAng,umag,gmag,rmag,imag,zmag,e_umag,e_gmag,e_rmag,e_
imag,e_zmag,objID,cl,q_mode,flags,Q,ObsDate,pmRA,e_pmRA,pmDE,e_pmDE,SpObjID,zsp,e_zsp,f_zsp,spType,spCl,subC
lass
0.305453,02595905+0000200,44.996055,+0.005565,0.170,0.160,76,16.376,15.770,15.258,0.097,0.140,0.141,ABB,222,
0,2451084.8062,J025959.06+000020.2,44.996116,+0.005624,0.002,0.002,90,19.548,18.186,17.619,17.379,17.241,0.0
28,0.006,0.007,0.007,0.013,1237663784217084122,6,1,0000201090020010,3,2003.8857,13,3,-5,3,0,,,,,,
0.080507,03000116+0001113,45.004857,+0.019806,0.060,0.060,90,12.529,11.954,11.874,0.024,0.030,0.029,AAA,222,
0,2451084.8062,J030001.17+000111.2,45.004879,+0.019802,0.061,0.060,90,17.398,15.191,14.183,16.934,13.777,0.0
11,0.005,0.003,0.018,0.006,1237663784217083948,6,0,0000F81090060010,3,2003.8857,20,4,24,4,0,,,,,,
1.331290,03000116+0001113,45.004857,+0.019806,0.060,0.060,90,12.529,11.954,11.874,0.024,0.030,0.029,AAA,222,
0,2451084.8062,J030001.08+000110.8,45.004509,+0.019681,0.062,0.057,0,24.566,25.148,17.596,13.890,22.827,2.39
3,1.716,0.032,0.001,2.226,1237663784217083950,3,0,0001F80092061110,3,2003.8857,,,,,0,,,,,,
4.789590,03000116+0001113,45.004857,+0.019806,0.060,0.060,90,12.529,11.954,11.874,0.024,0.030,0.029,AAA,222,
0,2451084.8062,J030001.01+000115.5,45.004220,+0.020974,0.002,0.002,90,21.956,19.689,18.110,16.886,16.261,0.1
41,0.014,0.008,0.006,0.008,1237663784217083949,6,1,0000201812060010,3,2003.8857,,,,,0,,,,,,
0.116926,03000100+0001154,45.004193,+0.020956,0.060,0.060,90,14.845,14.223,14.016,0.056,0.077,0.055,AAA,222,
0,2451084.8062,J030001.01+000115.5,45.004220,+0.020974,0.002,0.002,90,21.956,19.689,18.110,16.886,16.261,0.1
41,0.014,0.008,0.006,0.008,1237663784217083949,6,1,0000201812060010,3,2003.8857,,,,,0,,,,,,
4.728929,03000100+0001154,45.004193,+0.020956,0.060,0.060,90,14.845,14.223,14.016,0.056,0.077,0.055,AAA,222,
0,2451084.8062,J030001.08+000110.8,45.004509,+0.019681,0.062,0.057,0,24.566,25.148,17.596,13.890,22.827,2.39
3,1.716,0.032,0.001,2.226,1237663784217083950,3,0,0001F80092061110,3,2003.8857,,,,,0,,,,,,
4.833083,03000100+0001154,45.004193,+0.020956,0.060,0.060,90,14.845,14.223,14.016,0.056,0.077,0.055,AAA,222,
0,2451084.8062,J030001.17+000111.2,45.004879,+0.019802,0.061,0.060,90,17.398,15.191,14.183,16.934,13.777,0.0
11,0.005,0.003,0.018,0.006,1237663784217083948,6,0,0000F81090060010,3,2003.8857,20,4,24,4,0,,,,,,
0.084417,02595132+0002369,44.963851,+0.043587,0.220,0.170,95,16.476,16.057,15.564,0.113,0.175,,BCU,220,0,245
1084.8062,J025951.33+000236.9,44.963874,+0.043591,0.002,0.002,90,20.998,18.942,18.088,17.765,17.573,0.071,0.
009,0.007,0.008,0.016,1237663784217084100,6,1,0000001010000000,3,2003.8857,6,3,-3,3,0,,,,,,
0.267343,02595881+0002175,44.995074,+0.038204,0.380,0.310,0,16.746,15.814,16.125,0.134,0.140,0.324,BBD,222,0
,2451084.8062,J025958.80+000217.3,44.995029,+0.038145,0.009,0.008,90,24.543,21.773,20.167,18.857,18.180,0.69
0,0.055,0.021,0.012,0.024,1237663784217084474,6,1,0000001010000000,3,2003.8857,0,5,1,5,0,,,,,,
0.120901,03001158+0002539,45.048281,+0.048329,0.180,0.070,0,13.354,12.874,12.699,0.025,0.030,0.030,AAA,222,0
,2451813.9014,J030011.58+000253.8,45.048268,+0.048298,0.001,0.000,90,17.725,15.789,14.967,14.983,14.450,0.01
1,0.004,0.005,0.001,0.005,1237663784217084036,6,0,0001981294061048,3,2003.8857,-4,3,-15,3,0,,,,,,
@
```

Distribution ?

- With Hadoop / Spark, the data is distributed over several nodes
- Distribution ?
- How to optimise it ?



Test beds

- Data: >10k catalogues (SDSS, 2MASS, etc.)
  - Up to ~ 60 GB and several millions of elements on the output side (examples: 2MASS 58GB, SDSS DR9 54GB, ~49 10^6 output elements)

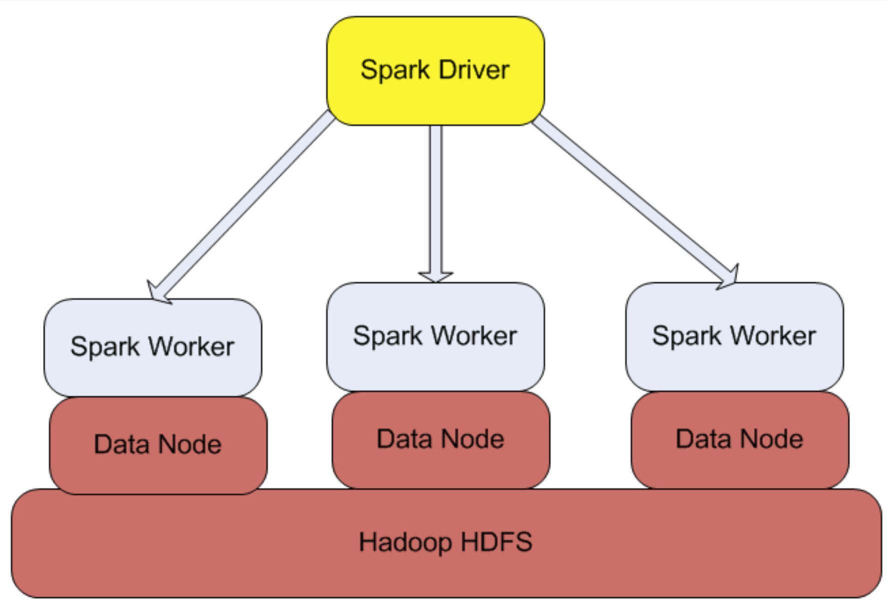
- Internal resources
  - Up to 6 nodes (4 cores, 16GB, 1 TB), common desktop machines under Ubuntu 14.04

- External resources from a provider
  - 12 nodes (dedicated servers, 4 cores, 32GB, Raid 2*2TB), Ubuntu 14.04

X-Match server (2*6 cores, 32GB, 12TB (15k tours))

Test beds (2)

- Common architecture using the Apache distributions (Spark 1.5.0 for Hadoop 2.6) + Java (no Cloudera, Hortonworks, …
- Standalone mode (with Spark own's cluster manager)
  - Without Apache Yarn, Mesos, …
  - Quick add of new nodes



Credits : BigHadoop

Study phase – Data preparation

- Before the execution the **input files are stored into HDFS**.
- These files are, in a first step, **loaded into 2 RDDs** ((Resilient Distributed Dataset, a distributed data collection) where each line of the RDD is an element containing the information about an object in the Sky.
- **Each RDD is then transformed in a PairRDD** (RDD containing a (Key, Value) pair): a **key** representing the **source pixel number** is attributed to each element of the RDD based on the HEALPix tessellation of the Sky.
- The elements of the PairRDDs are then **(Key, Value)** couples where the Value contains all the information whose the source (ra, dec) coordinates in the equatorial system.

Study phase – Data preparation (2)

- The **PairRDDs** are then **distributed** over the **cluster nodes**.
- This **distribution** is done following a **hash partitioning** where the PairRDDs are split in partitions which will be stored over the nodes.
- The hash partitioning consists in **grouping all the elements having the same key (same pixel number) in the same partition**.
- The partitions are stored into different nodes
  - **The elements with the same key are on the same nodes**
  - This data distribution is essential to the second programme phase.
- At the end the **PairRDDs** are stored **into HDFS as binary files following a method preserving the structure (Key, Value)**.

Study phase – Join

- The binary files stored previously are directly loaded into two PairRDDs.

- A treatment is applied on the second PairRDD: a **duplication** of some **sources** in the **neighbour pixels**.

- The **2 PairRDDs** are then **joined following the Key**. This join generates a new PairRDD where the elements are (Key, Value1, Value2) triples.

- As the join is done following the Key (cell number), 2 near sources can be in the different cells and so they are not joined (=> **duplication of sources in the neighbour cells to limit the side effects**).

Study phase – Join (2)

- The **duplication** steps
  - A circle with a fixed radius is drawn around the source
  - If neighbour pixels are partially in this circle, the source is then duplicated in the neighbour cells.
- The **joined elements** are then **filtered**
  - Only the joined elements which distance between the two sources is under a **given threshold** are conserved.
- The **final result is stored in HDFS** in a text format for a later visualization and use.

Illustration

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

HEALPix sky cutting

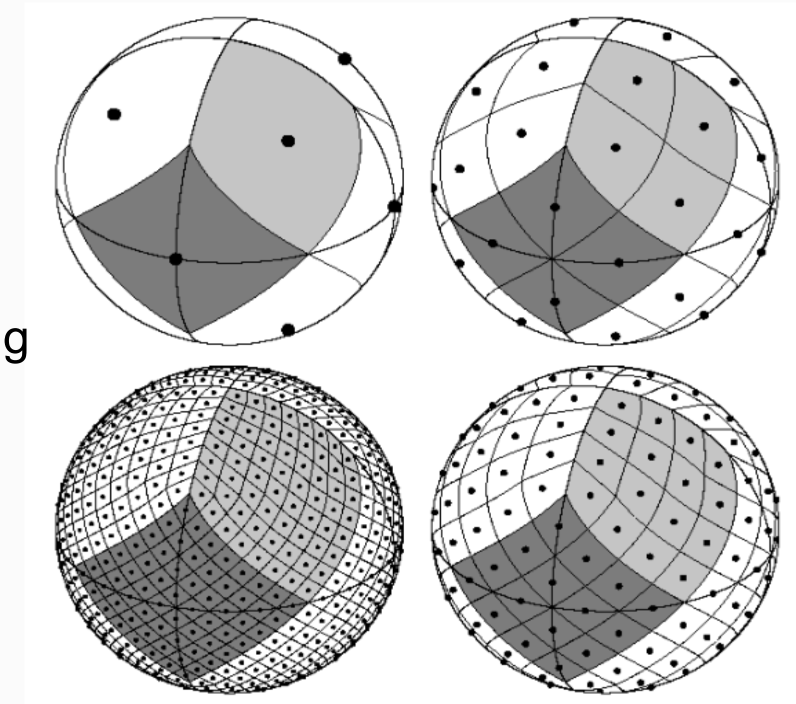
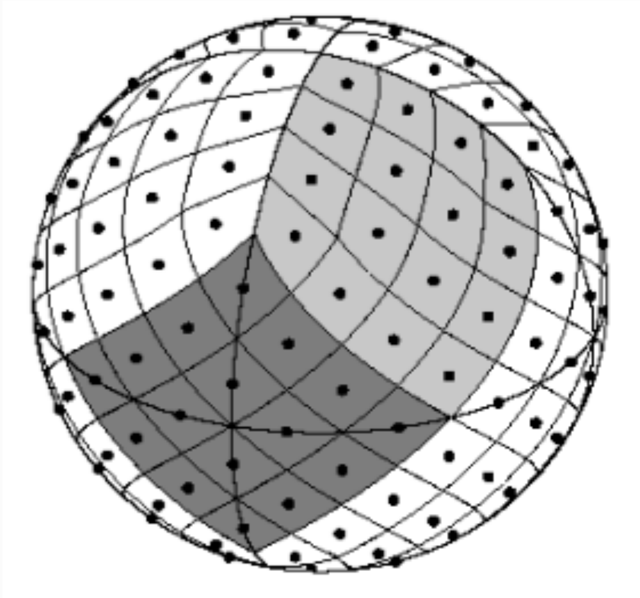


Illustration (2)

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



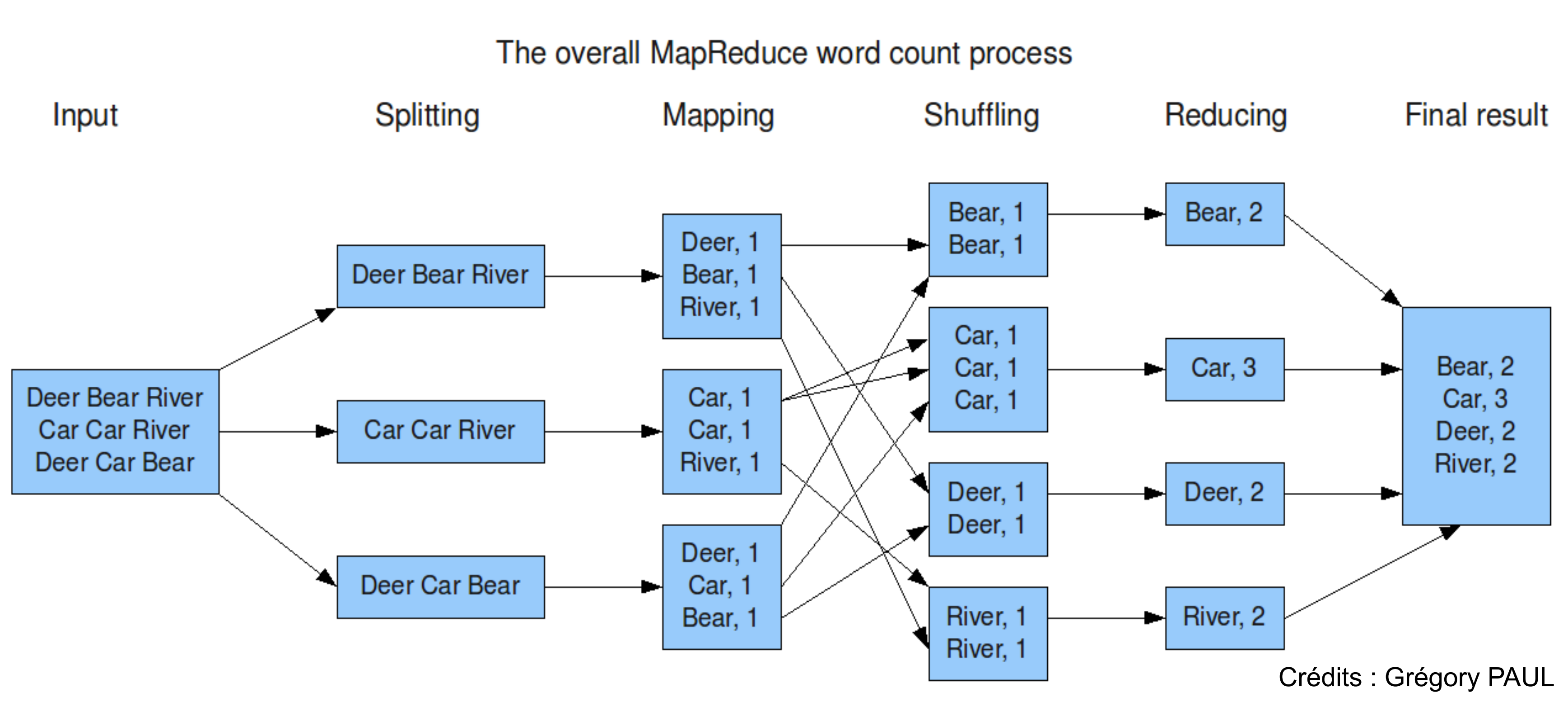
Credits : HEALPix – arXiv:astro-ph/0409513

Study phase – Co-location

- During the hash partitioning of the RDDs, the elements having the same key are stored on the same nodes for a given RDD.
- **This means not that the keys which are common to 2 RDDs are also on the same nodes.** In this case, it implies a transfer **overhead** between the nodes during the join, which has an impact on the **performances** (=> Shuffle + no co-location of the Data).

« Shuffle » phase

- Re-distribution on the nodes

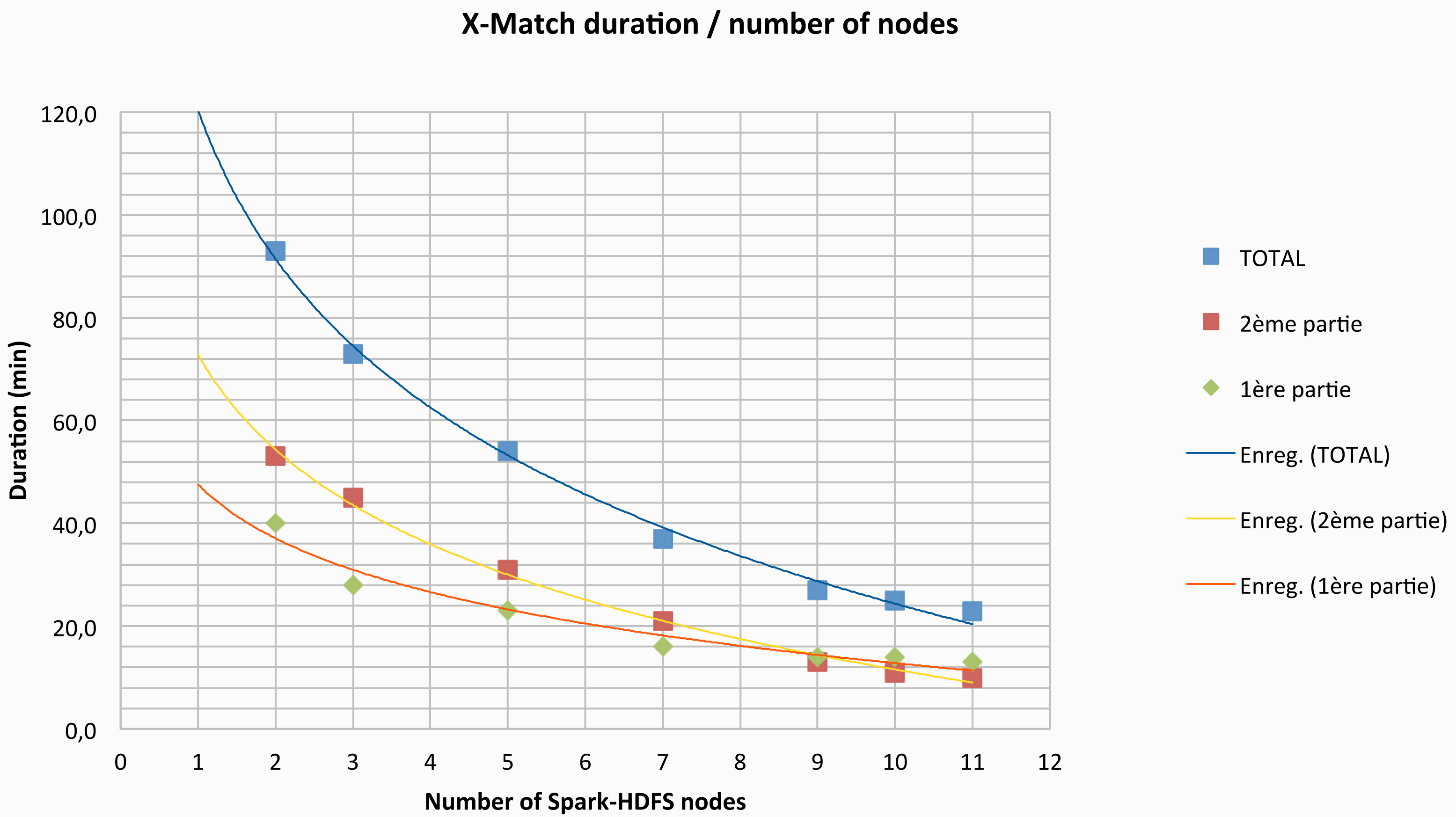


Results

- 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

Cross-Match (duplication des sources faite dans la 2e partie ; avec toutes les données en sortie)											
Taille des blocs HDFS = 128MB pour les fichiers en entrée ; sdss7.csv et 2mass.csv répliqués 2x											
HashPartitioner	60 partitions										
Taille des blocs HDFS en sortie	32MB										
Nombre de nœuds Spark/HDFS	1	2	3	4	5	6	7	8	9	10	11
1ère partie : préparation des données		**40,0**	**28,0**		**23,0**		**16,0**		**14,0**	**14,0**	**13,0**
mapToPair (sdss7.csv)		7,8			5,1		4,9		4,9	4,8	4,7
saveAsHadoopFile (sdss7.bin)		10,0			5,7		2,7		2,0	2,3	1,5
mapToPair (2mass.csv)		8,5			5,7		5,2		5,2	5,1	5,0
saveAsHadoopFile (2mass.bin)		13,0			6,5		3,6		1,9	1,6	1,4
2ème partie : jointure		**53,0**	**45,0**		**31,0**		**21,0**		**13,0**	**11,0**	**9,9**
mapToPair (sdss7.bin)					7,2		4,7		3,5	3,0	2,9
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		**93,0**	**73,0**		**54,0**		**37,0**		**27,0**	**25,0**	**22,9**

Results (2)



The CDS X-Match service needs 15 minutes of computation for the same data (which is split in multiple files in HDFS), It corresponds to the second part (data is already prepared)

Conclusion and perspective

- Our current results:
  - We have reached an execution time better than the X-Match service
  - From 8 nodes it could be an alternative to the current architecture
  - Concerning the cost the dedicated server set (rent) is interesting (example: 8*60*12, around 6000 euros / an)
- Remark: following recent discussions, the test beds could be optimized (RAM per core, no RAID, etc.)

Conclusion and perspective (2)

- Bottleneck: « shuffle »
  - Optimisation (?) of the code using the « data co-location », « block affinity groups » is an on-going work at Apache
- We wish to do new tests with more nodes but also with other configurations (RAM, Hard disks, etc.)
- On going collaboration with Julien Nauroy (Université Paris Sud) who has deployed a Spark architecture in his University

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/