Cas d'utilisation de Spark en Astrophysique: Cross Match de catalogues de sources

André Schaaff, François-Xavier Pineau
CDS, Centre de Données astronomiques de Strasbourg
Noémie Wali, Paul Trehiou
Elèves-ingénieurs UTBM, Université de technologie de Belfort-Montbéliard
Julien Nauroy
DI Paris-Sud

Journées Plateforme 6 & 7 octobre 2016, Clermont Ferrand

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

IVOA

Asterics
Astronomy ESFRI & Research Infrastructure Cluster

Le fil d'Ariane

Contexte

Apache Spark

La motivation

Les données et le service de « cross-match »

Les bancs de test

Première phase d'étude & Résultats

Deuxième phase (en cours)

Perspectives

Contexte

Une exploration continue de nouvelles technologies, notamment « Big Data » (… et diversifiée)

Une première phase (9/2015 -> 2/2016 + prolongations)
« Pour se faire la main, identifier les verrous, débuter des collaborations, etc. »

Une deuxième phase en cours depuis début septembre 2016 pour 6 mois
« Pour approfondir, introduire de nouvelles technologies (Docker) »

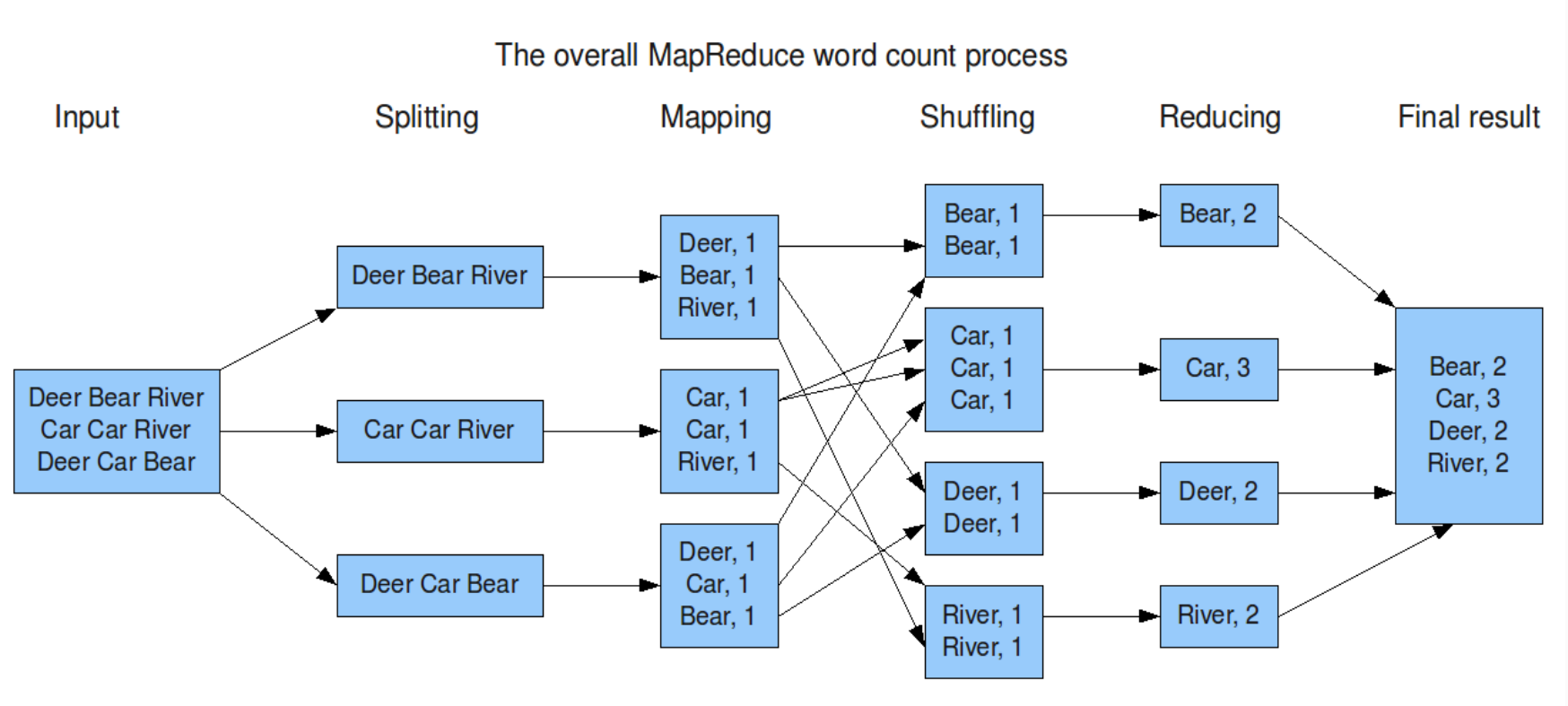
Apache Spark

- 2009, premiers développements à l'AMPLab de l'UC de Berkeley (Matei Zaharia)...
- 2013, reprise par la fondation Apache
- 2014, V 1.0 avec pour particularité notable des performances élevées (par rapport à MapReduce)
- … mi-2016 V2.0, unification des Datasets et des Dataframes

Apache Spark (2)

- Spark peut fonctionner avec divers systèmes de fichiers dont HDFS (Hadoop Distributed File System)
- Spark apporte des améliorations à MapReduce notamment au niveau du Shuffle (phase durant laquelle les données sont déplacées vers les noeuds sur lesquels s'effectuera la partie « reduce »)
- Il utilise un stockage en mémoire pour les étapes intermédiaires et réduit les accès disques -> très intéressant lorsque l'on travaille de façon itérative avec le même jeu de données
- Si la mémoire est insuffisante, Spark utilise l'espace disque

Illustration



Crédits : Grégory PAUL

Apache Spark (3)

- Spark introduit des modèles de données lui permettant de gérer une persistance des données durant les traitements
  - RDD (Resilient Distributed Dataset) pour stocker des objets (au sens OO), ils sont manipulables via des opérateurs ensemblistes type map ou reduce
  - Datasets pour représenter les données tabulaires manipulables via:
    - des commandes SQL
    - des fonctions SparkQL
    - des opérations ensemblistes
- API Spark: Scala, Java, Python, …

La motivation (de cette étude)

- Nous souhaitions évaluer ce que Hadoop / Spark pouvait apporter en étudiant un cas d'utilisation précis, le « cross-match » de catalogues de sources:
  - Remplacement ou amélioration de l'existant, notamment au niveau du passage à l'échelle (jeux de données de taille croissante (exponentielle…), souplesse au niveau matériel, déploiements, etc.)
  - Pour quel coût (budget, main d'oeuvre, performances (améliorées ?))
- Mais également nous familiariser avec Hadoop / Spark

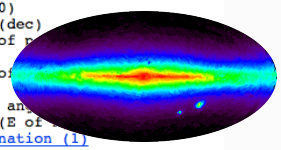
Les données…

- Données issues des catalogues de sources
- Exemples (nb sources):
  - 2MASS[1], 470,992,970
  - SDSS[2] DR9, 469,053,874

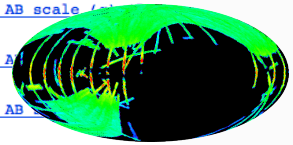
Exemple de fichiers ReadMe associés aux catalogues de sources accessibles via le service VizieR

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

Bytes	Format	Units	Label	Explanations
1- 10	F10.6	deg	RAdeg	(ra) Right ascension (J2000)
12- 21	F10.6	deg	DEdeg	(dec) Declination (J2000) (dec)
23- 26	F4.2	arcsec	errMaj	(err_maj) Semi-major axis of p… error ellipse
28- 31	F4.2	arcsec	errMin	(err_min) Semi-minor axis of… error ellipse
33- 35	I3	deg	errPA	[0,180] (err_ang) Position an… ellipse major axis (E of …
37- 53	A17	---	2MASS	(designation) Source designation (1)
55- 60	F6.3	mag	Jmag	?(j m) J selected default magnitude (2)
62- 66	F5.3	mag	Jcmsig	?(j cmsig) J default magnitude uncertainty (3)
68- 72	F5.3	mag	e_Jmag	?(j msigcom) J total magnitude uncertainty (4)
74- 83	F10.1	---	Jsnr	?(j_snr) J Signal-to-noise ratio



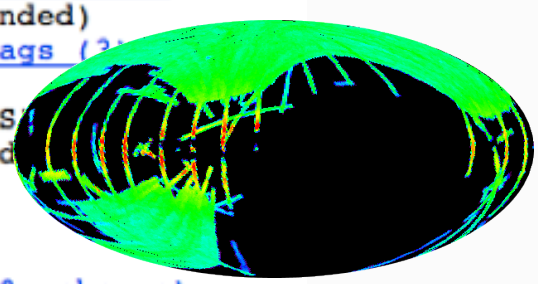
Bytes	Format	Units	Label	Explanations
1	I1	---	mode	[1,2] 1: primary (469,053,874 sources), 2: secondary (324,960,076 sources).
2	A1	---	q_mode	[+] '+' indicates clean photometry (105,969,748 sources with mode 1+)
3	I1	---	cl	Type (class) of object (3=galaxy, 6=star) (1)
5- 23	A19	---	SDSS9	SDSS-DR9 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 …
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 …
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)



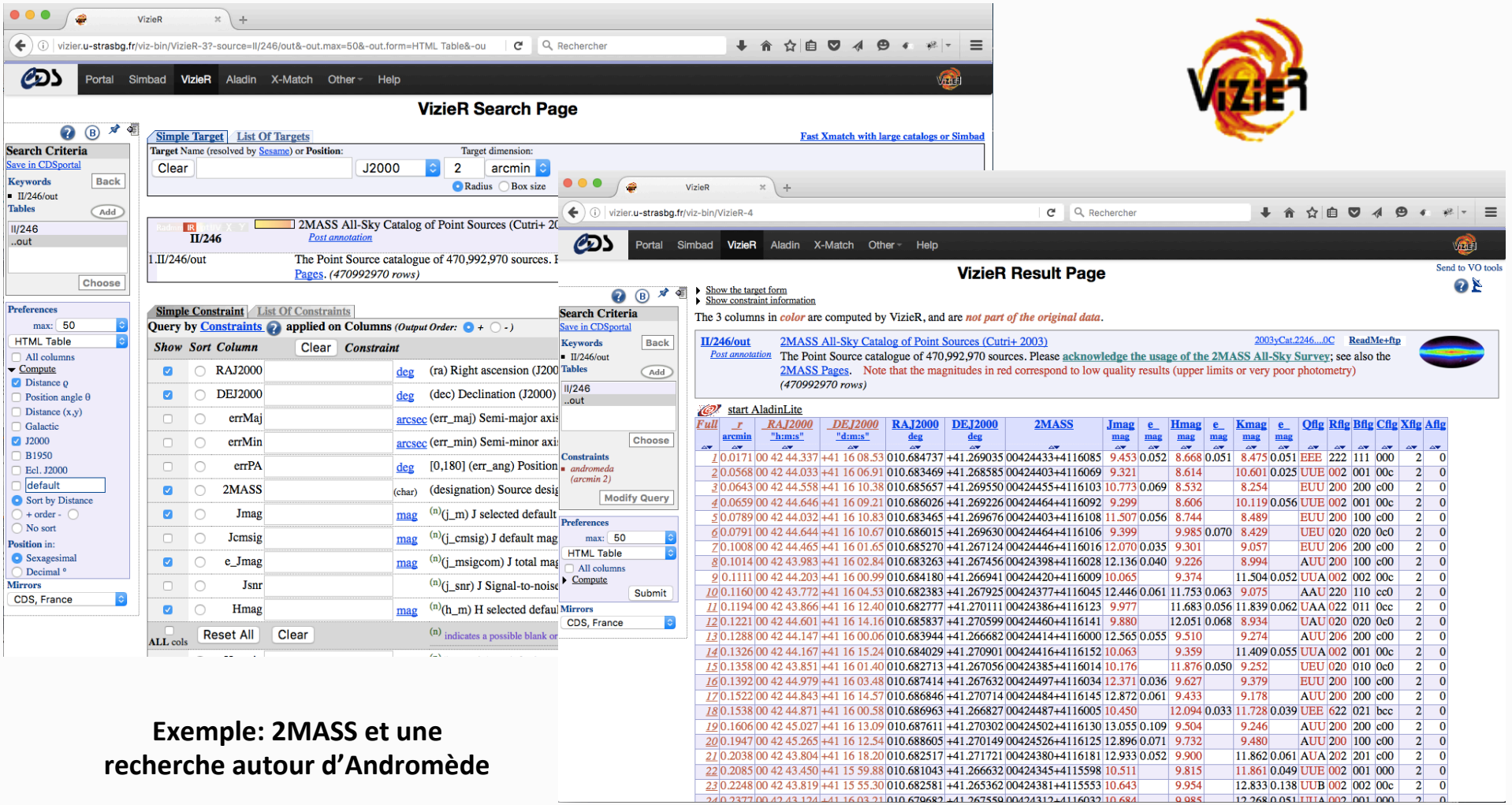
Les données… - focus

SDSS[2] DR9, 469,053,874 sources

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
163-172	F10.6	deg	DEdeg	Declination of the object (ICRS) (d
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	---	---	[:]



Les données… (dans VizieR)



Exemple: 2MASS et une recherche autour d'Andromède

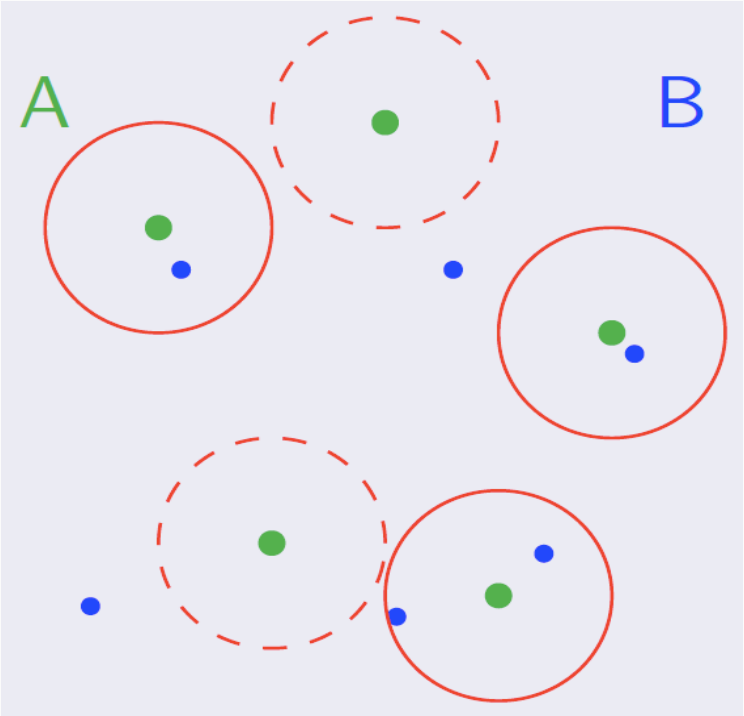
Les données… (dans VizieR) - focus

Full	_r arcmin	_RAJ2000 "h:m:s"	_DEJ2000 "d:m:s"	RAJ2000 deg	DEJ2000 deg	2MASS	Jmag mag	e_ mag	
1	0.0171	00 42 44.337	+41 16 08.53	010.684737	+41.269035	00424433+4116085	9.453	0.052	
2	0.0568	00 42 44.033	+41 16 06.91	010.683469	+41.268585	00424403+4116069	9.321		
3	0.0643	00 42 44.558	+41 16 10.38	010.685657	+41.269550	00424455+4116103	10.773	0.069	
4	0.0659	00 42 44.646	+41 16 09.21	010.686026	+41.269226	00424464+4116092	9.299		
5	0.0789	00 42 44.032	+41 16 10.83	010.683465	+41.269676	00424403+4116108	11.507	0.056	
6	0.0791	00 42 44.644	+41 16 10.67	010.686015	+41.269630	00424464+4116106	9.399		
7	0.1008	00 42 44.465	+41 16 01.65	010.685270	+41.267124	00424446+4116016	12.070	0.035	
8	0.1014	00 42 43.983	+41 16 02.84	010.683263	+41.267456	00424398+4116028	12.136	0.040	
9	0.1111	00 42 44.203	+41 16 00.99	010.684180	+41.266941	00424420+4116009	10.065		
10	0.1160	00 42 43.772	+41 16 04.53	010.682383	+41.267925	00424377+4116045	12.446	0.061	1
11	0.1194	00 42 43.866	+41 16 12.40	010.682777	+41.270111	00424386+4116123	9.977		1
12	0.1221	00 42 44.601	+41 16 14.16	010.685837	+41.270599	00424460+4116141	9.880		1
13	0.1288	00 42 44.147	+41 16 00.06	010.683944	+41.266682	00424414+4116000	12.565	0.055	

…et le service de « cross-match »

- Le service de « cross-match » permet de réaliser une corrélation / identification croisée de sources entre (très) grands catalogues (ordre de grandeur actuelle: 10^9).

Jointure floue entre 2 tables de plusieurs centaines de millions de données

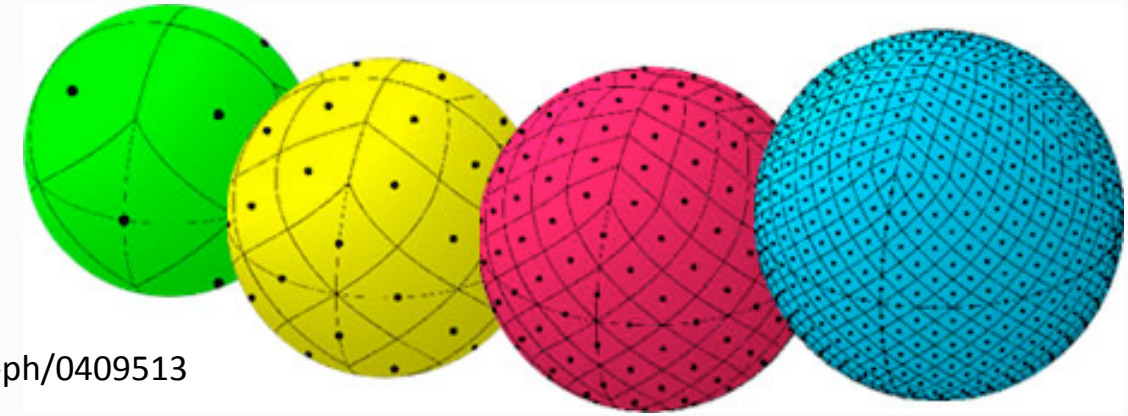


…et le service de « cross-match » (2)

- Il est possible de le faire pour les catalogues proposés par le CDS mais également de télécharger ses propres données (une table avec positions) pour les croiser avec l'un de ces catalogues.
- C'est un service basé sur des développements optimisés et une implémentation sur un serveur bien dimensionné (pour une utilisation en ligne).

…et le service de « cross-match » (3)

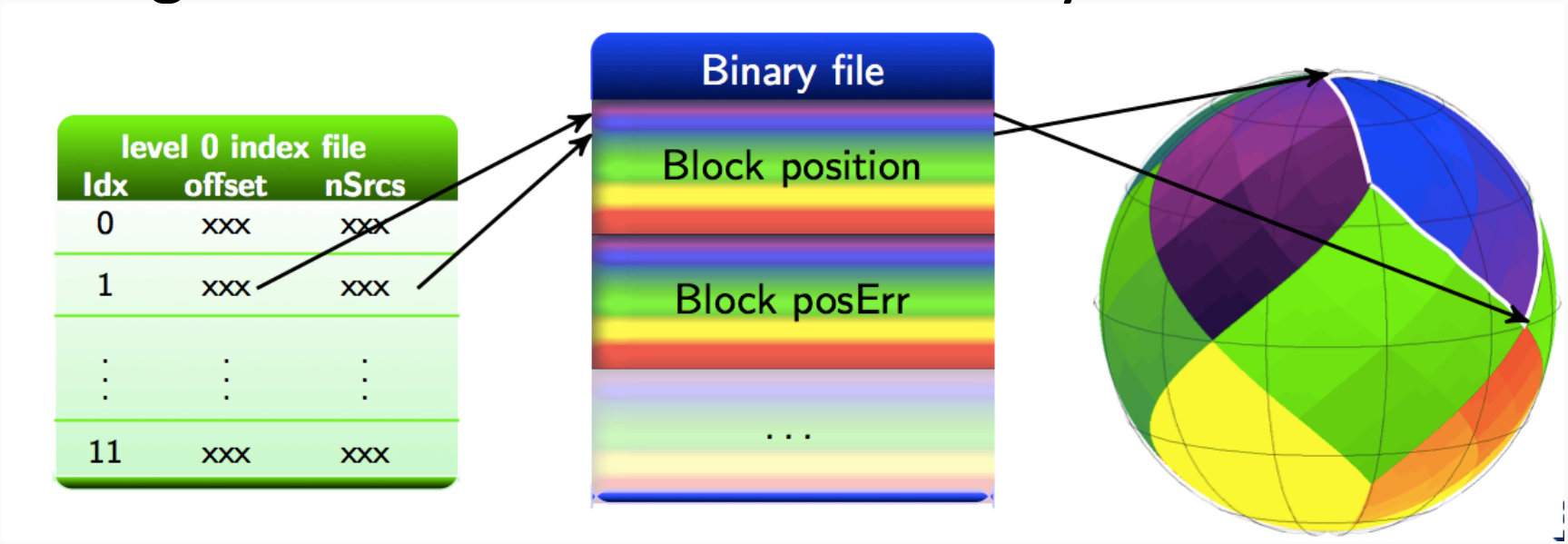
- Zone concernée
  - Tout le ciel: toutes les sources
  - Un cône: uniquement les sources à une certaine distance angulaire d'une position donnée
  - Une cellule HEALPix (pixellisation du ciel)



Crédits: HEALPix – arXiv:astro-ph/0409513

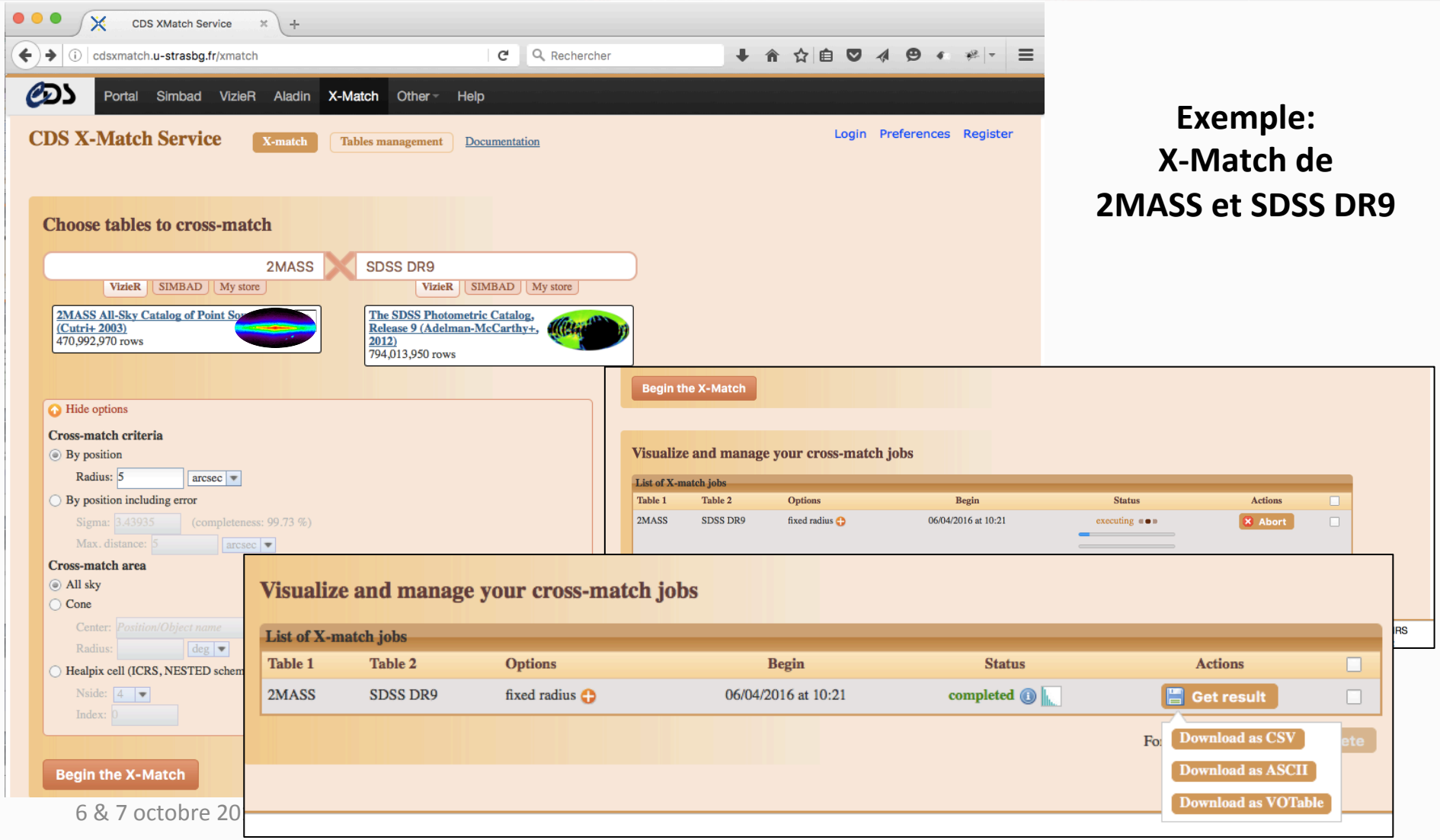
…et le service de « cross-match » (4)

- Les données ne sont pas distribuées mais organisées et stockées sur un système RAID



Le ciel est découpé en losanges de tailles identiques, appelés pixels, chaque source ou objet du ciel est positionné dans un pixel numéroté.

Illustrations



Exemple:
X-Match de
2MASS et SDSS DR9

Illustrations (2)

Exemple:
Un « extrait » du résultat en CSV

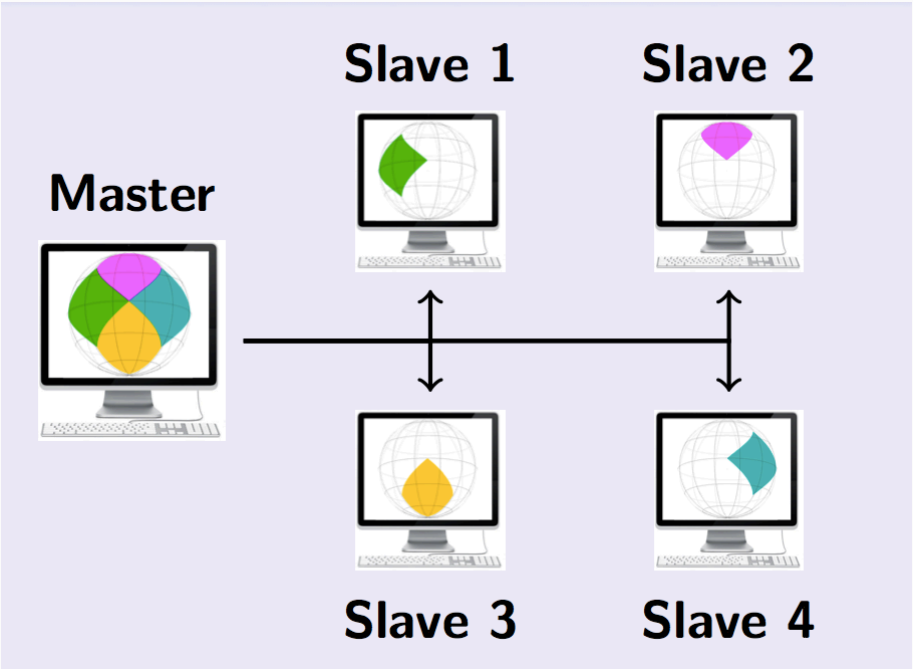
Téléchargements — vi 1459930879752A.csv — 108×36

```
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,
```


```
angDist,2MASS,RAJ2000,DEJ2000,errHalfMaj,errHalfMin,errPosAng,Jmag,Hmag,Kmag,e
MeasureJD,SDSS9,RAdeg,DEdeg,errHalfMaj,errHalfMin,errPosAng,umag,gmag,rmag,ima
imag,e_zmag,objID,cl,q_mode,flags,Q,ObsDate,pmRA,e_pmRA,pmDE,e_pmDE,SpObjID,zs
lass
0.305453,02595905+0000200,44.996055,+0.005565,0.170,0.160,76,16.376,15.770,15.
0,2451084.8062,J025959.06+000020.2,44.996116,+0.005624,0.002,0.002,90,19.548,1
28,0.006,0.007,0.007,0.013,1237663784217084122,6,1,0000201090020010,3,2003.885
```

Et s'il était distribué?

- Dans le cas de Hadoop / Spark, les données sont distribuées sur plusieurs serveurs
- Comment les données sont-elles distribuées ?, comment optimiser cette distribution ?



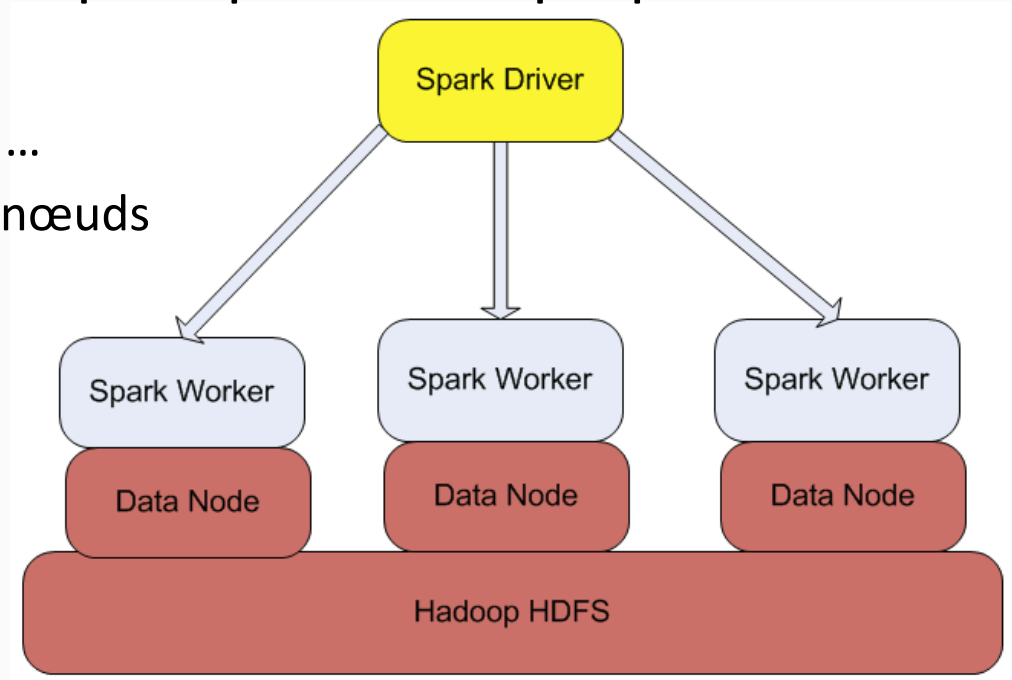
Les « bancs de test »

- Données: nombreux catalogues (SDSS, 2MASS, etc.)
  - Ordre de grandeur jusqu'à ~ 60 Go et plusieurs dizaines de millions d'éléments en sortie (exemples: 2MASS 58Go, SDSS DR9 54Go, ~49 10^6 éléments en sortie)
- Ressources internes
  - Jusqu'à 6 nœuds (4 cœurs, 16Go, 1 To), des machines de bureau sous Ubuntu 14.04
- Ressources externes louées ponctuellement
  - 12 nœuds OVH (serveurs dédiés), 4 cœurs, 32Go, Raid 2*2To, sous Ubuntu 14.04

Serveur de X-Match (2*6 coeurs, 32Go, 12To (15k tours))

Les « bancs de test » (2)

- Architecture classique en utilisant directement les distributions d'Apache (Spark 1.5* pour Hadoop 2.6) + Java
- Mode standalone dans lequel Spark a son propre gestionnaire de cluster
  - Sans Apache Yarn, Mesos, …
  - Ajout rapide de nouveaux nœuds



Crédits : BigHadoop

* …et maintenant Spark 2.0

Phase d'étude – Préparation des données

- Avant l'exécution **les fichiers d'entrée sont stockés dans HDFS**.
- Ces fichiers sont dans un premier temps **chargés dans deux RDDs** ((Resilient Distributed Dataset, une collection distribuée de données) simples où chaque ligne du RDD est un élément contenant des informations sur un objet dans le ciel.
- **Chaque RDD est ensuite transformé en PairRDD** (RDD contenant une paire de clé/valeur): à chaque élément du RDD est attribuée une clé représentant le numéro de pixel de la source grâce au découpage HEALPix du ciel.
- Les **éléments des PairRDDs** sont alors des **couples (clé, valeur)** où la **valeur contient toutes les informations** dont les coordonnées (ra, dec) de la source dans le système de coordonnées équatoriales.

Phase d'étude – Préparation des données (2)

- Les **PairRDDs** sont ensuite **distribués** sur les **nœuds du cluster**.
- Cette **distribution** est faite sur la base **d'un partitionnement par hachage** où les PairRDDs sont découpés en partitions qui vont être stockés sur les nœuds.
- Le partitionnement par hachage consiste **à regrouper tous les éléments ayant la même clé (même numéro de pixel) dans une même partition**.
- Les partitions sont donc stockées sur des nœuds différents
  - **Les éléments de même clé se retrouvent sur les mêmes nœuds**
  - Cette distribution des données est essentielle pour la deuxième partie du programme.
- Enfin les **PairRDDs** sont **enregistrés dans HDFS sous forme de fichiers binaires grâce à une méthode permettant de garder la structure (clé, valeur)**.

Phase d'étude - Jointure

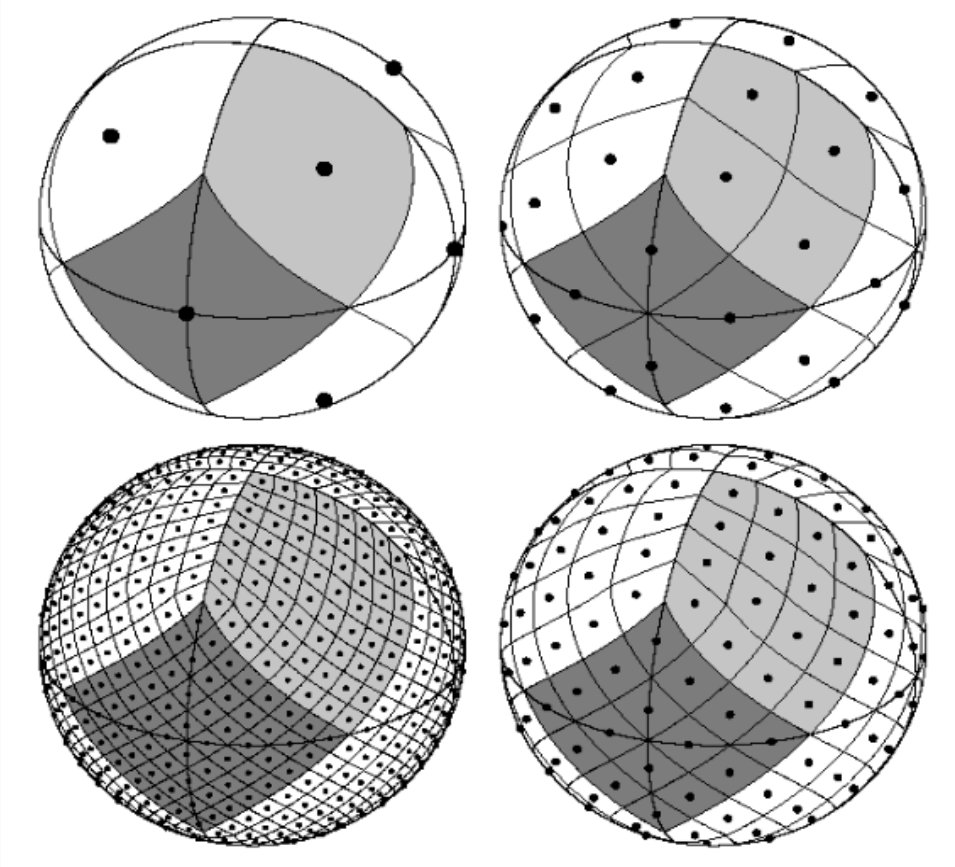
- Les fichiers binaires enregistrés précédemment sont directement chargés dans deux PairRDDs.
- Sur le deuxième PairRDD est appliquée une méthode qui **duplique** certaines sources dans les **pixels voisins**.
- Les **deux PairRDDs** sont ensuite **joints au niveau de la clé**. La jointure donne lieu à un nouveau PairRDD où les éléments sont de type (clé, valeur1, valeur2).
- La **jointure** étant faite sur la **clé** (numéro de cellule), deux sources proches peuvent être dans des cellules différentes et ne sont donc pas jointes (d'où la **duplication des sources dans les cellules voisines pour limiter les effets de bord**).

Phase d'étude – Jointure (2)

- **La duplication est effectuée de la manière suivante**
  - Un cercle de rayon fixé est tracé autour de la source
  - Si des pixels voisins se trouvent en partie à l'intérieur de ce cercle, la source est alors dupliquée dans ces cellules voisines.
- **Les éléments joints sont ensuite filtrés**
  - Seuls les éléments joints dont la distance entre les deux sources est inférieure à un certain seuil sont gardés.
- **Le résultat final est enregistré dans HDFS sous format texte pour une visualisation et une utilisation ultérieures**.

Illustration

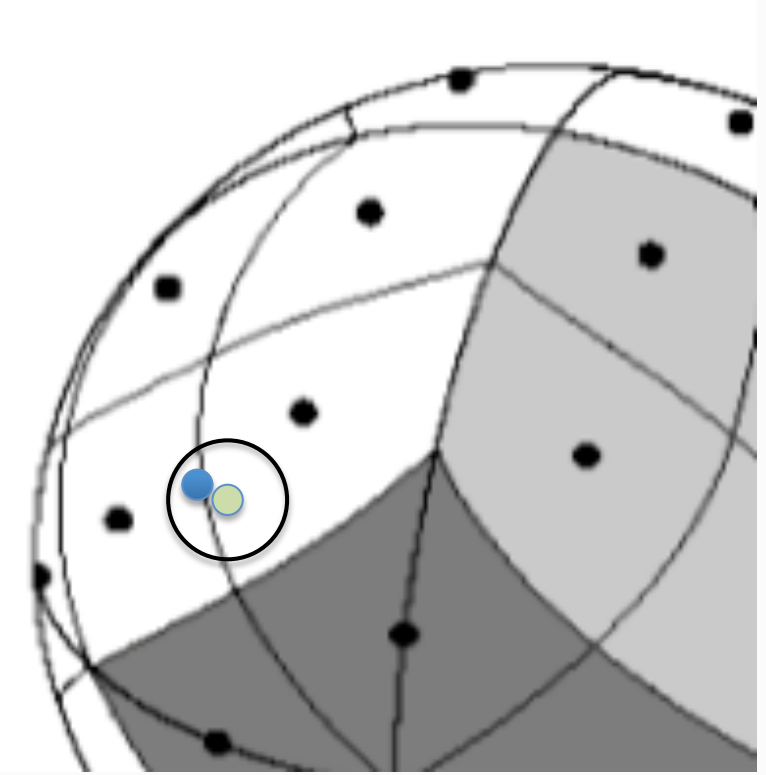
- Une implémentation du X-Match en MapReduce, Couples (clé = n°de pixel, valeur)



Découpage HEALPix du ciel

Illustration (2)

- Effets de bord
  - Jointure floue
  - Duplication des sources dans les cellules voisines si besoin



Crédits : HEALPix – arXiv:astro-ph/0409513

Phase d'étude – Co-location

- Lors du partitionnement par hachage des RDDs, les **éléments de même clé** sont placés sur les **mêmes nœuds pour un RDD donné**.

- Ceci n'implique pas que des clés communes à deux RDDs soient également sur les mêmes nœuds. Dans ce cas, cela engendre **un temps de transfert des données entre les nœuds lors de la jointure**, ce qui **affecte les performances**.

Résultats

- Données en entrée (SDSS DR7 (sources primaires) et 2MASS): fichiers de 54GB et 58GB ; 357 175 411 et 470 992 970 d'objets
- Données en sortie: 49 208 820

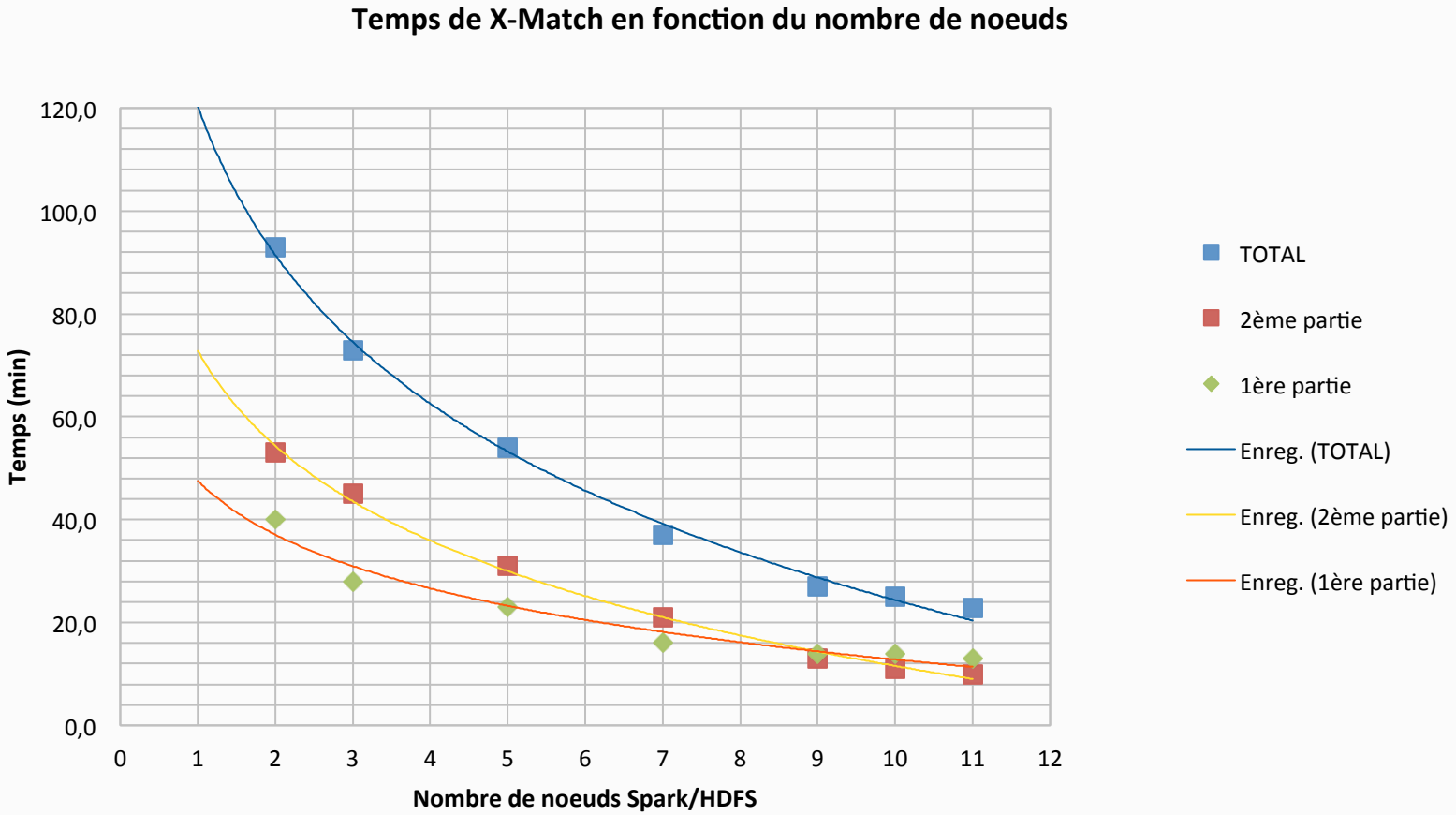
Cross-Match (duplication des sources faite dans la 2e partie ; avec toutes les données en sortie)											
Taille des blocs HDFS = 128MB pour les fichiers en entrée ; sdss7.csv et 2mass.csv répliqués 2x											
HashPartitioner	**60 partitions**										
Taille des blocs HDFS en sortie	**32MB**										
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**

Le service de X-Match actuel nécessite **15 minutes** de traitement pour les mêmes données.

Résultats (2)- focus

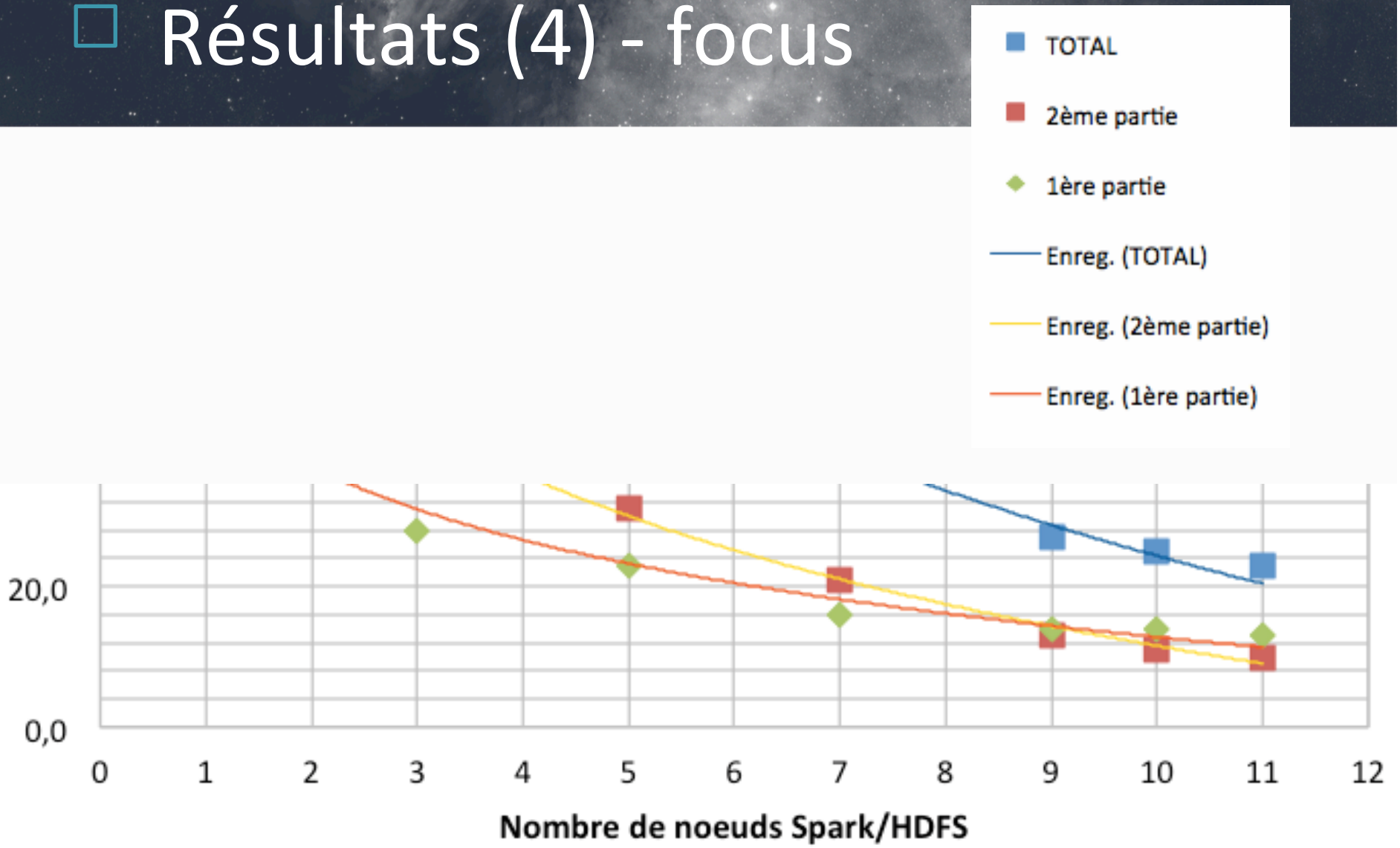
7	8	9	10	11
16,0		**14,0**	**14,0**	**13,0**
4,9		4,9	4,8	4,7
2,7		2,0	2,3	1,5
5,2		5,2	5,1	5,0
3,6		1,9	1,6	1,4
21,0		**13,0**	**11,0**	**9,9**
4,7		3,5	3,0	2,9
8,3		5,5	4,9	4,3
7,6		3,4	2,4	2,3
37,0		**27,0**	**25,0**	**22,9**

Résultats (3)



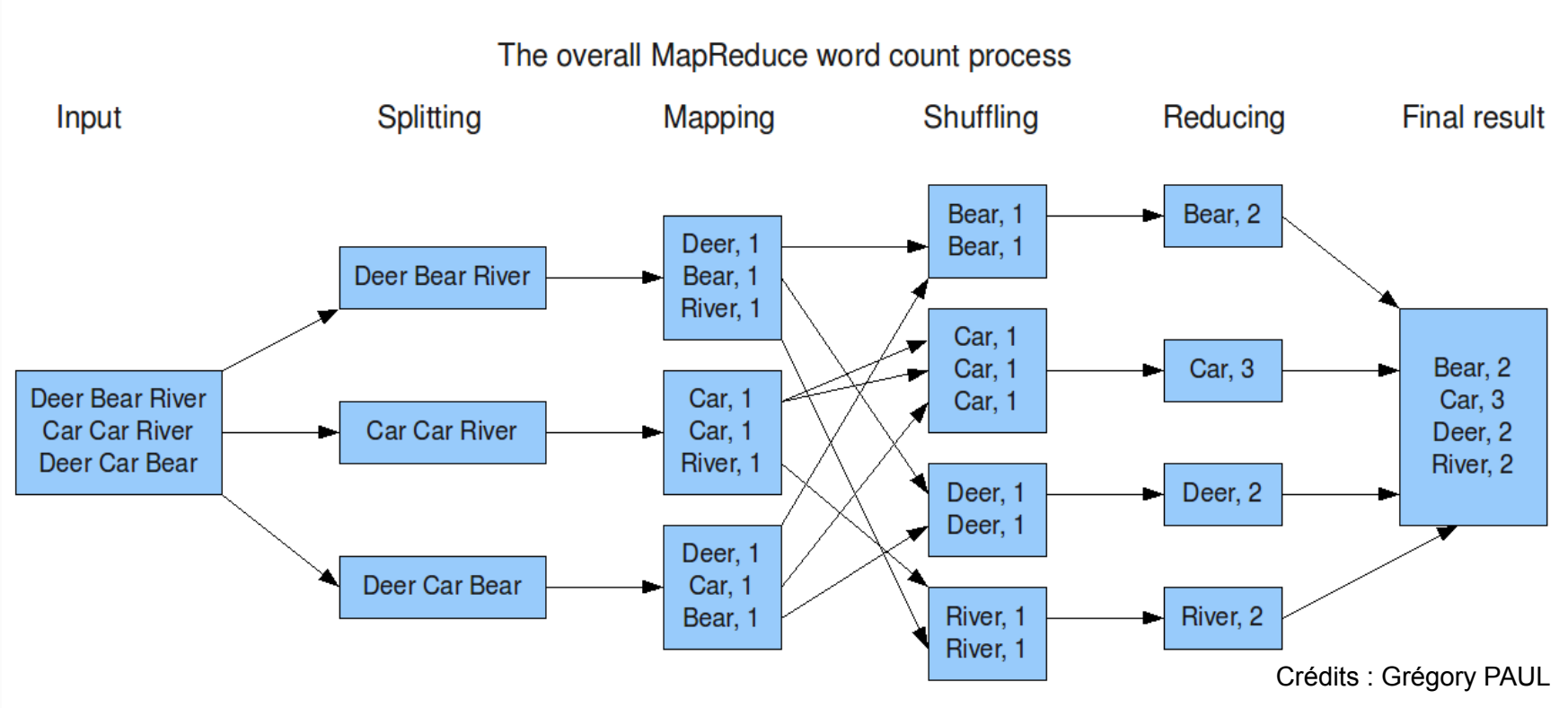
Le service de X-Match actuel nécessite 15 minutes de traitement pour les mêmes données (découpées en de multiples fichiers dans HDFS), ce qui correspond à la seconde partie (les données sont déjà préparées)

Résultats (4) - focus



Résultats (5) - Phase de « shuffle »

- Redistribution sur les nœuds



Crédits : Grégory PAUL

Résultats (6)

- Les résultats obtenus:
  - On obtient un temps inférieur à celui du service de X-Match
  - A partir de 8/9 nœuds cela peut devenir une alternative à l'architecture existante
  - En terme de coût l'ensemble de serveurs dédiés « en location » est intéressant (exemple: 8*60*12, env. 6000 euros / an)
- N.B.: le banc de test n'était pas « optimisé »

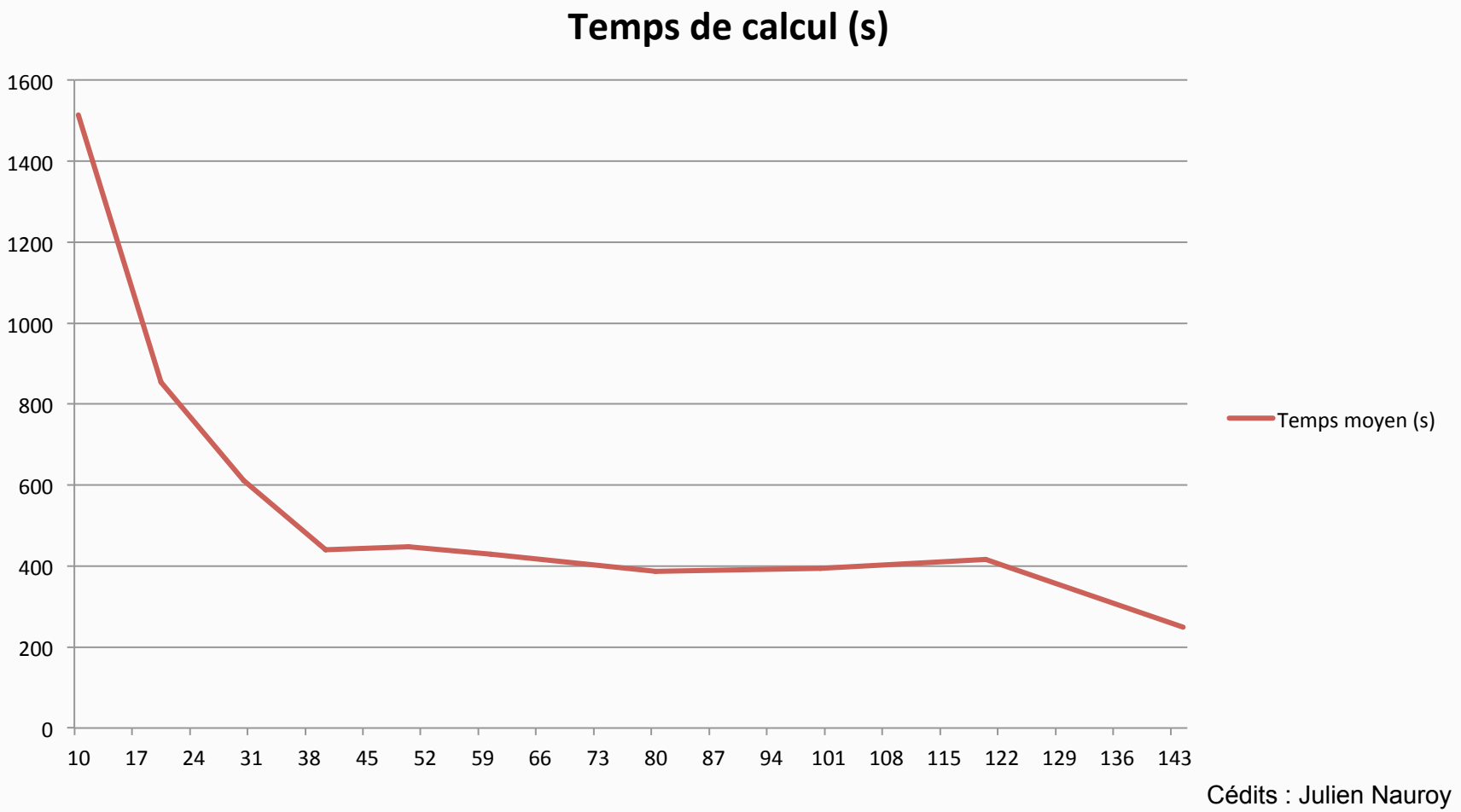
Résultats (7)

- Goulot d'étranglement: « shuffle »
  - Optimisation possible (?) du code par la « co-location des données », pas de « block affinity groups »

Prolongations de la phase 1

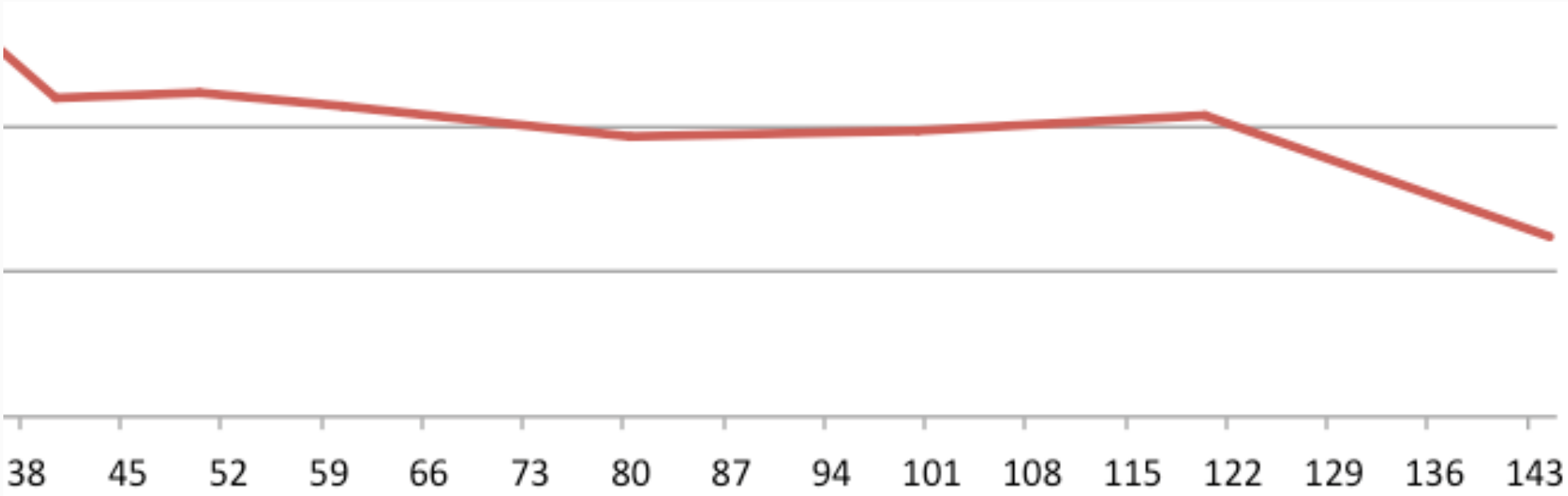
- Suite à la réunion LoOPS consacrée à Spark en avril 2016 au LAL, Julien Nauroy (DI Paris Sud) a réalisé des tests du X-Match avec son cluster Spark

Résultats – benchmark J. Nauroy



Cédits : Julien Nauroy

Résultats – benchmark - focus



Résultats – benchmark J. Nauroy (2)

- Jusqu'à 40 cœurs, comportement linéaire
- Après 40 cœurs: plus d'amélioration
  - Goulot d'étranglement = débits disque/réseau
- Vers 144 cœurs…
  - Probablement dû à l'absence de « spill »

Résultats – benchmark J. Nauroy (3)

- Pistes d'optimisation
  - Lecture avec session.read.csv
  - Faire varier les partitions
    - des jeux en entrée
    - des structures intermédiaires
    - en particulier spark.sql.shuffle.partitions
  - Optimiser la taille mémoire

Résultats – benchmark J. Nauroy (3)

- Jointure de grands jeux pas optimale
  - 200 partitions par défaut
- Lecture CSV tout à fait viable
  - Mieux si on définit le schéma
- Beaucoup à gagner par essais successifs
  - Partitionnement
  - Types de données (Datasets, RDD)

Deuxième phase

- Débutée en septembre pour 6 mois
- Capitaliser sur le travail précédent, utiliser Spark 2.0
- Passage à Scala (l'API Java est « expérimentale »)
- Se rapprocher d'une mise en production par de nouveaux tests hardware
- Explorer les possibilités d'autres technologies en tant que complément, par exemple Docker

Deuxième phase (2)

- « Dockerisation » de Spark
  - Dans un contexte où l'on veut amener le traitement vers les données
  - Passer d'une simple exécution, par exemple d'un jar, à une exécution environnée
  - Intégration continue avec Drone / GitLab
  - Jupyter en cours d'évaluation
- Différents problèmes à traiter / résoudre notamment en terme de sécurité, de persistance des données, etc.
- Cette partie est plutôt en bonne voie

Perspectives

- Beaucoup d'avancées en seulement un mois, quelques ambitions pour les 5 mois restants
  - Améliorer de façon notable les performances au niveau du X-Match tout en restant dans le domaine du raisonnable (notion à définir !) en terme d'architecture et de coût
  - Amener la « dockerisation » dans un état stable et « en production » pour Spark mais également dans le cadre plus général du CDS

Crédits : Grégory PAUL

Et surtout…

- Echanges et collaborations !!!
- Nos différents travaux sont documentés et peuvent servir à d'autres

Crédits : Grégory PAUL

Liens

- 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/

Et Journée Spark au LoOPS, http://reseau-loops.github.io/journee_2016_04.html