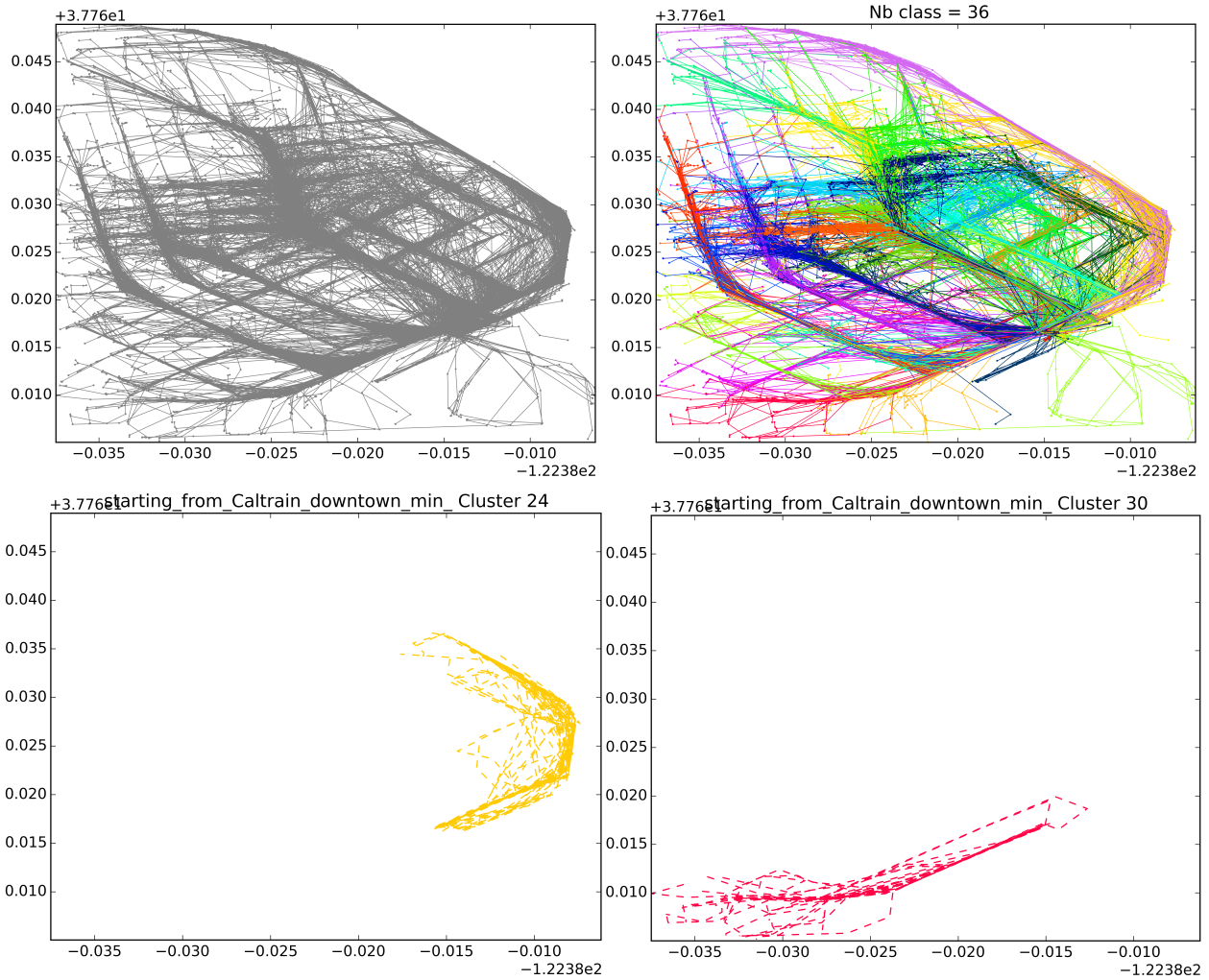
Apprentissage de Données Massives
Trois cas d'usage avec R, Python et Spark

Philippe Besse, Brendan Guillouet et Jean-Michel Loubes

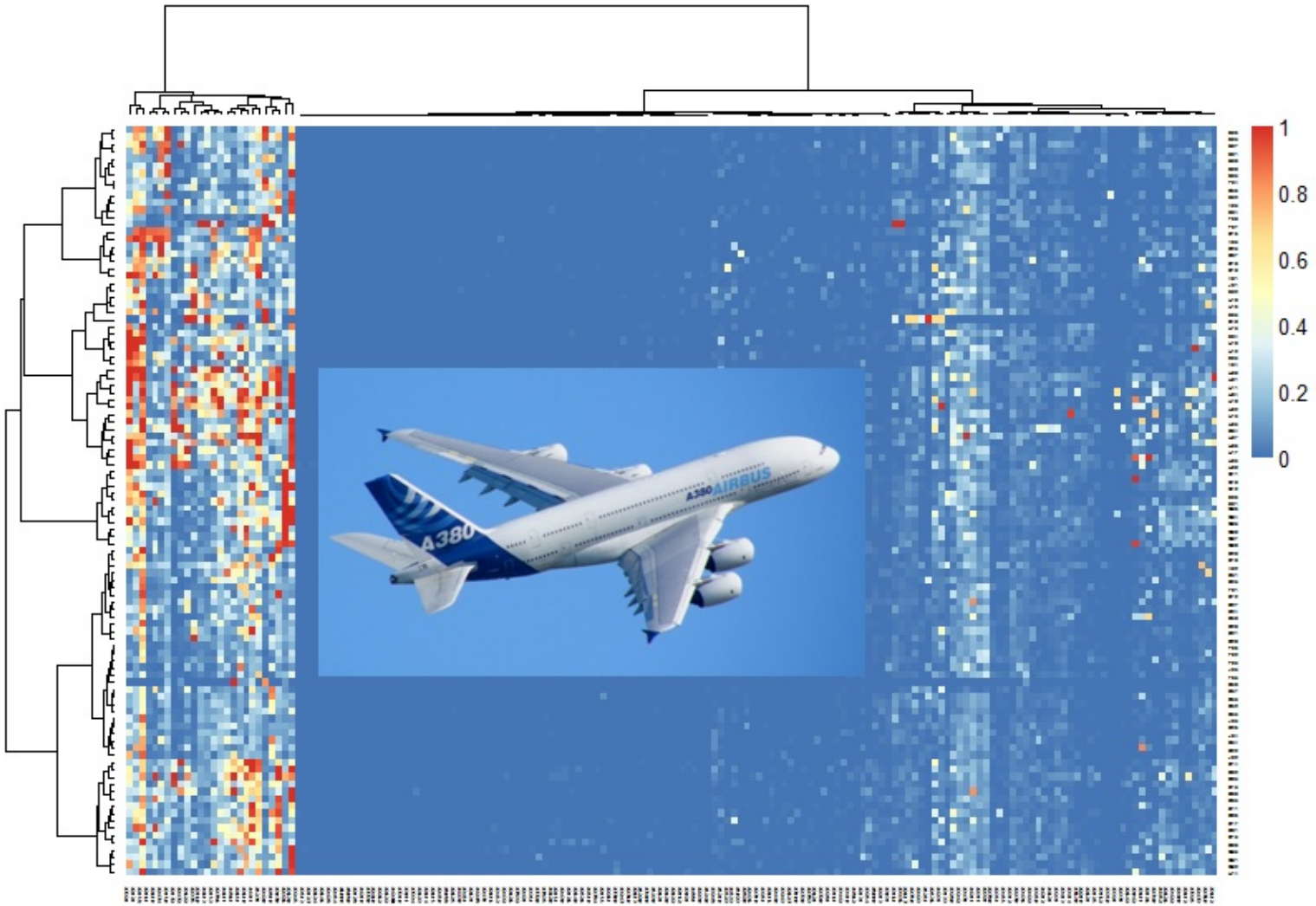
Université de Toulouse
INSA – Dpt GMM
Institut de Mathématiques – ESP
UMR CNRS 5219



Taxis : Classification de trajectoires GPS

Objectif

- Question : Quelle (meilleure) technologie utiliser pour l'apprentissage sur données massives ?
- Grosses data ? RAM, DD ?
- Matériel : Poste personnel, serveur, *Cloud* ?
- Données distribuées (*Hadoop*) ?
- Comparaisons de difficiles à impossibles
- Point de vue du "statisticien"
  - Prototype puis passage à l'échelle du même code
  - Trois cas d'usage : MNIST, MovieLens, Cdiscount
  - Méthodes et algorithmes : préparation, RF, NMF, Logit
  - Trois environnements : R à Python, Python à Spark ?
- Scripts dans `http://github.com/wikistat`



AIrbus : Analyse des messages d'incidents en vol (700 000 en 6 mois)

Nouvelle (?) *Science des Données*

- 1995 *Data Mining* : Gestion de la Relation Client, suites logicielles (Friedman, 1997)
- 2010 *Data Science* : Publicité en ligne, *Hadoop, cloud computing* (Donoho, 2015 ; beamergotobuttonBesse et Laurent, 2016)
  - Données préalables (fouille), dimensions (omiques) $p >> n$,
  - Pas de nouvelles méthodes, seules celles échelonnables
  - Data pas toujours *Grosses* mais *datification* du quotidien
  - Éthique et virtualisation / transparence de la décision
- Science des données : nouveau terme d'erreur
  - Erreur d'optimisation + Compromis biais / variance
  - Optimisation stochastique (Robbins Monro) ; non différentiable
  - Parallélisation des librairies de calcul
- Nouveau modèle économique

Contenu de l'exposé

- Introduction aux technologies
  - R et Python *Scikit-learn*
  - Hadoop, *MapReduce*, exemple de k-*means*
  - Spark
  - SparkML, MLlib
- Performances dans 3 cas : MNIST, MovieLens, Cdiscount
- Matériels
  - Lenovo X240 (Windows 7), proc. 4 cœurs 1.7 GHz, 8Go
  - MacBook Pro (OS X), proc. 4 cœurs 2,2 GHz, 16Go
  - Plateforme (`Hupi.fr`, Linux Spark 1.6), 1 maître (*driver*), 8 exécuteurs (*workers*) de chacun 7Go de RAM

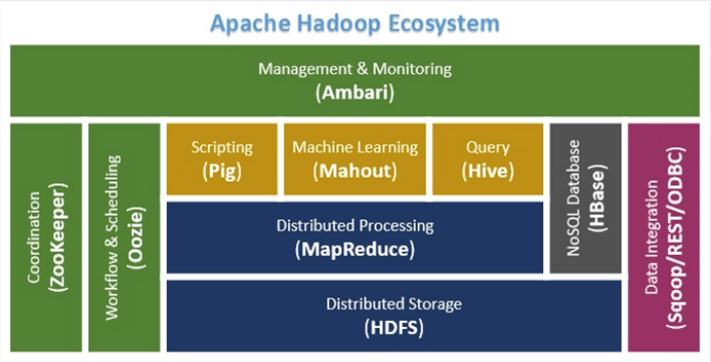
Nouveau modèle économique

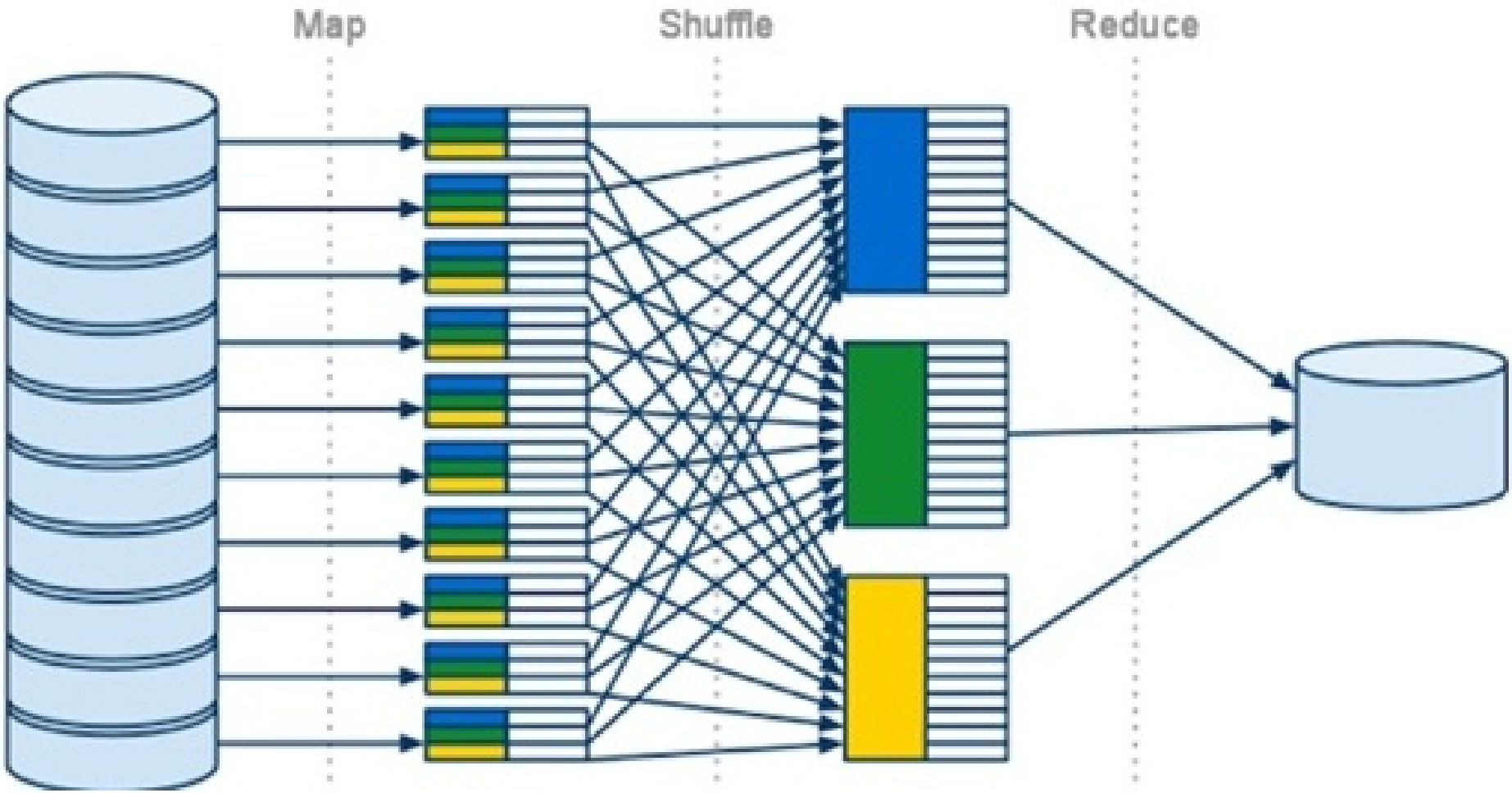
- Marges réduites sur matériels
- Logiciels sous licence GNU, MIT, Apache...
- Vendre du service :
  - Enthought (Canopy), Continuum analytics (Anaconda),
  - Horthon Works, Cloudera (Hadoop, Spark...)
  - Databricks (Spark), Oxdata (H20)
  - Revolution Analytics (RHadoop) – Microsoft
- Nouveau marché du *cloud computing*
  - Platform aaS, Software aaS, Service aaS...
  - Amazon Web Service
  - Microsoft Azure, Google Cloud Computing
  - IBM Analytics, SAS Advanced Analytics...
- Développement industriel *vs.* Recherche académique

Hadoop, MapReduce

- Environnement : *Google* puis Apache (2009)
- *Hadoop Distributed File System* (HDFS)
- Données hétérogènes distribuées
- Tolérance aux pannes matérielles
- *Scalabilité* (milliers de nœuds)
- Parallélisation : *Map Reduce*

- Communication par (clef, valeur)
- Déplacer les algorithmes, pas les données
- Données *immuables*
- Lecture unique ou *streaming*

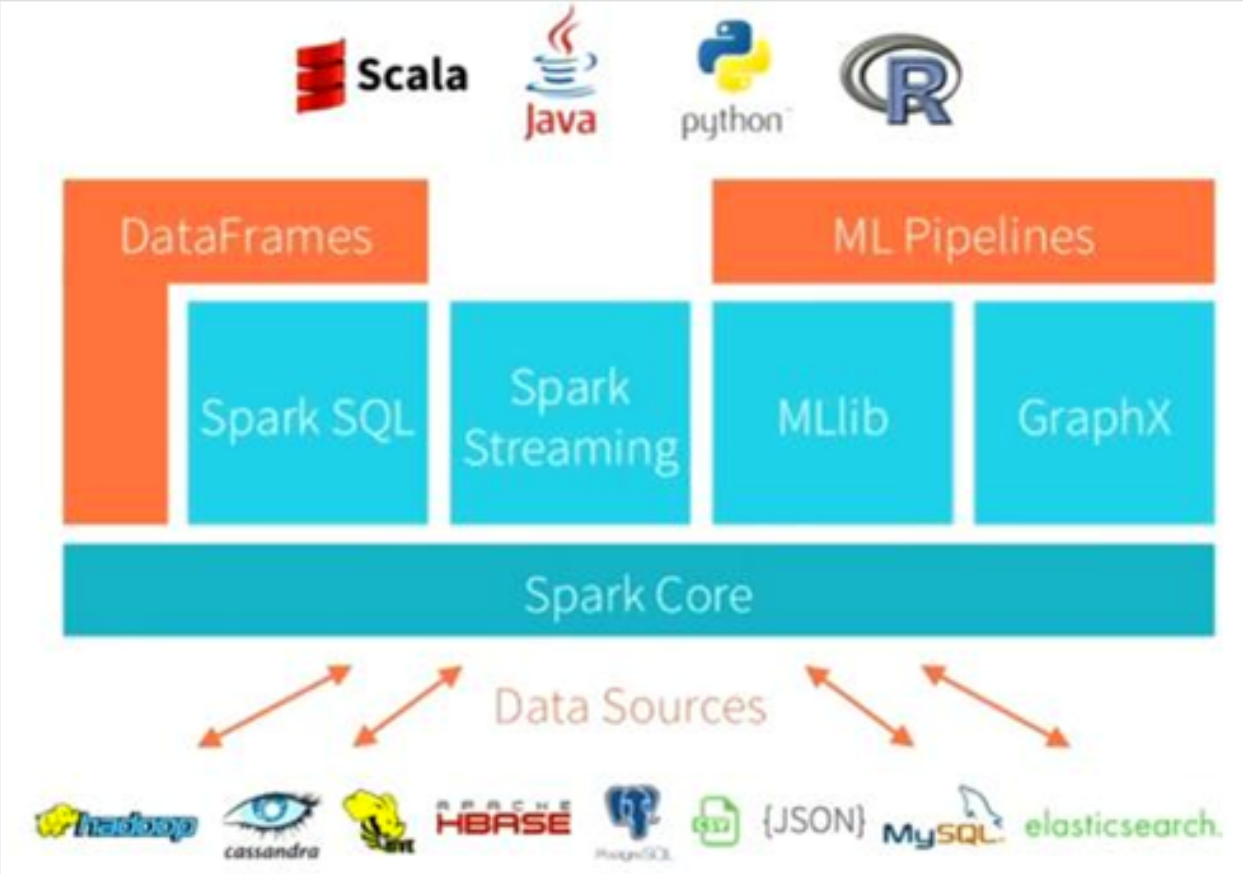




Hadoop Distributed File System (HDFS) & MapReduce

Classification par centres mobiles ($\approx$ *k-means*)

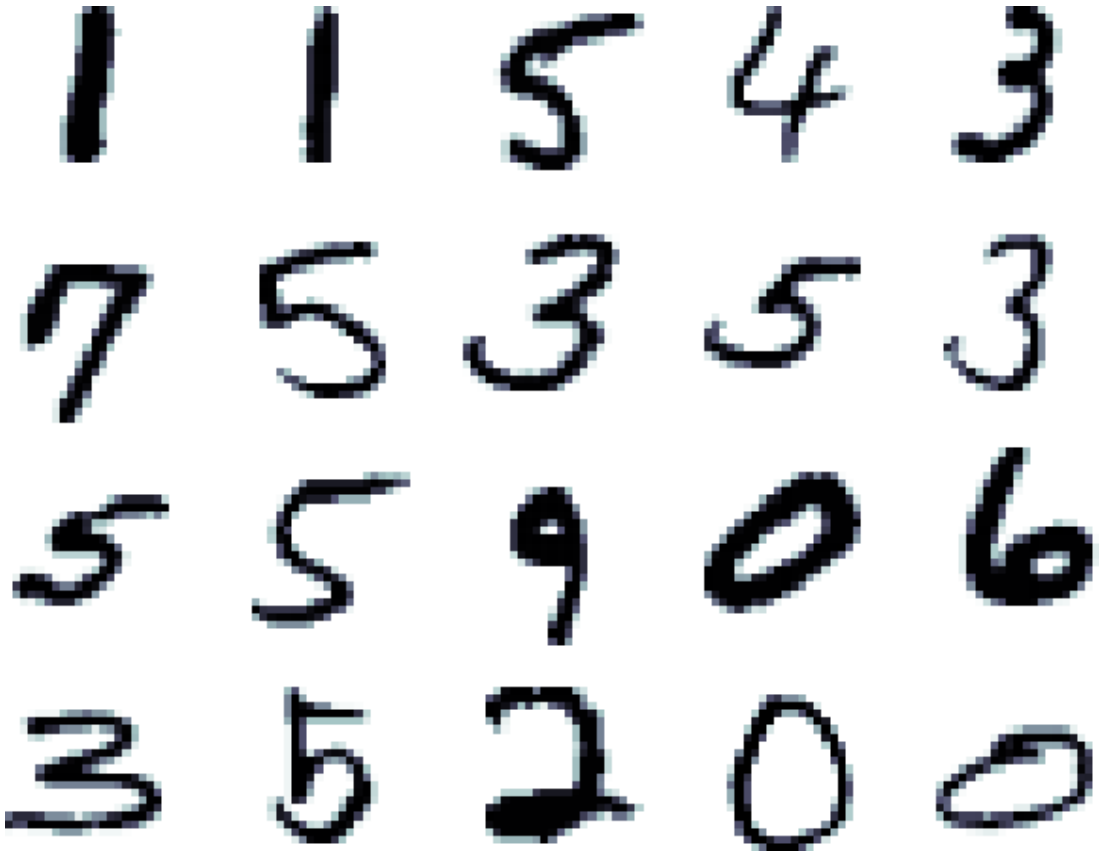
- Définition d'une distance euclidienne
- Algorithme de Forgy (1965)
  - Initialisation des k centres
  - Itération des étapes *MapReduce*
    - Map : Affectation de chaque individu `(valeur)` au centre `(clef)` le plus proche
    - Reduce : Calcul des centres des individus de même clef
    - Mise à jour des centres
- Problème : accès disques à chaque itération
- Solution actuelle de Spark : *Resilient Distributed Dataset*, (Zaharia *et al.*, 2012)



La technologie Spark et son écosystème

Librairies MLlib et SparkML

- Spark 1.6 à 2.0
- MLlib (Resilient Distributed Dataset) : k-*means*, SVD, NMF (ALS), Régression linéaire et logistique (l_1 et l_2), SVM linéaires, Classifieur Bayésien Naïf, Arbre, Forêt Aléatoire, Boosting
- Migration vers SparkML : *DataFrame*, *pipeline*
- Peu de méthodes mais passage à l'échelle "Volume"



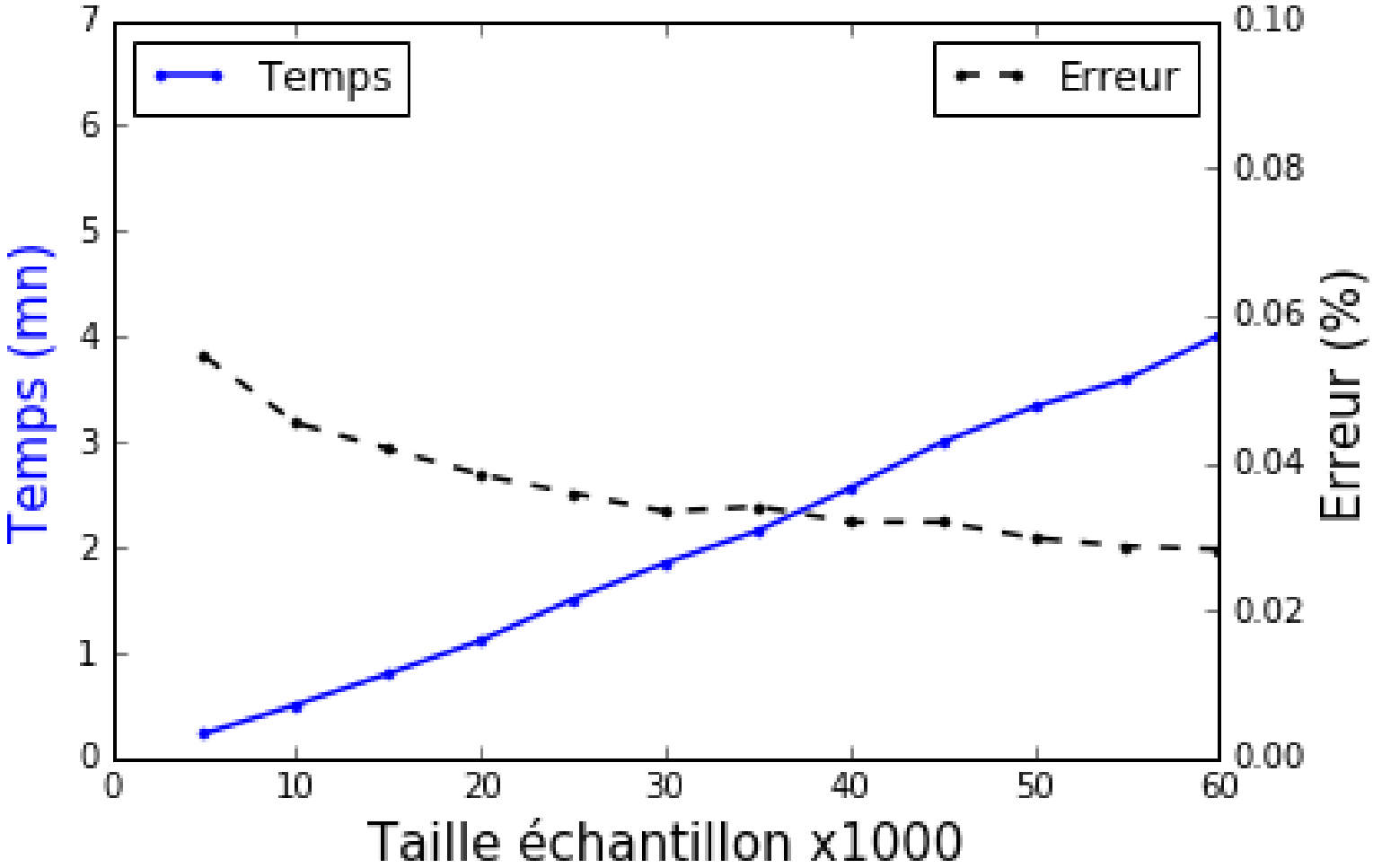
MNIST : quelques exemples d'images de caractères

MNIST : État de l'art

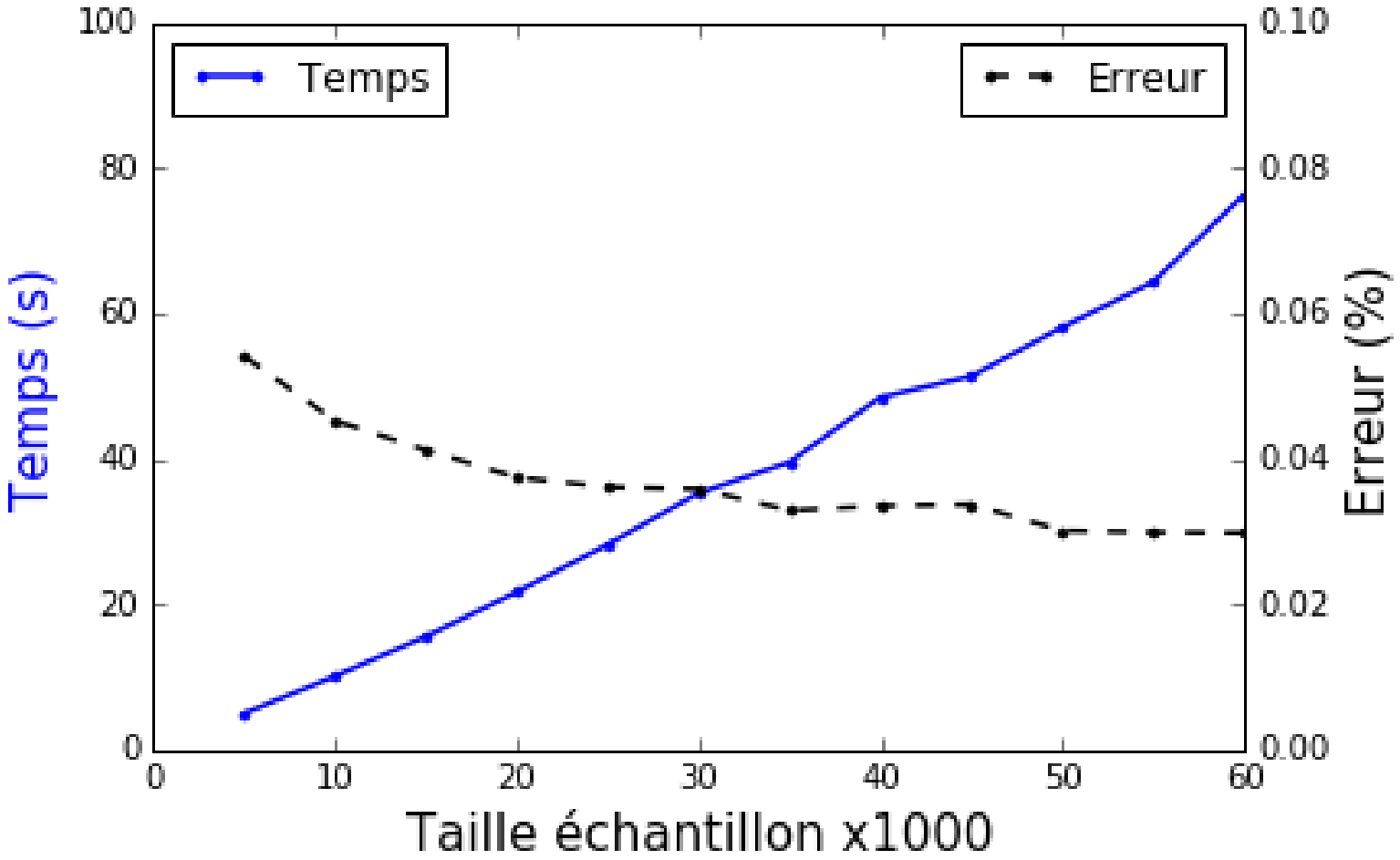
- Site de Yann le Cun
- 60 000 caractères, $28 \times 28 = 784$ pixels
- Test : 10 000 images
- Méthodes classiques (k-nn, RF)
- Prétraitement de normalisation des images
- Distance spécifique (tangeantielle) avec propriétés d'invariance
- Ondelettes et *scattering* (Stéphane Mallat)
- Apprentissage Profond : *TensorFlow, Lasagne, Keras*
- ...
- Comparer l'usage des méthodes classiques

Implémentations de Random Forest

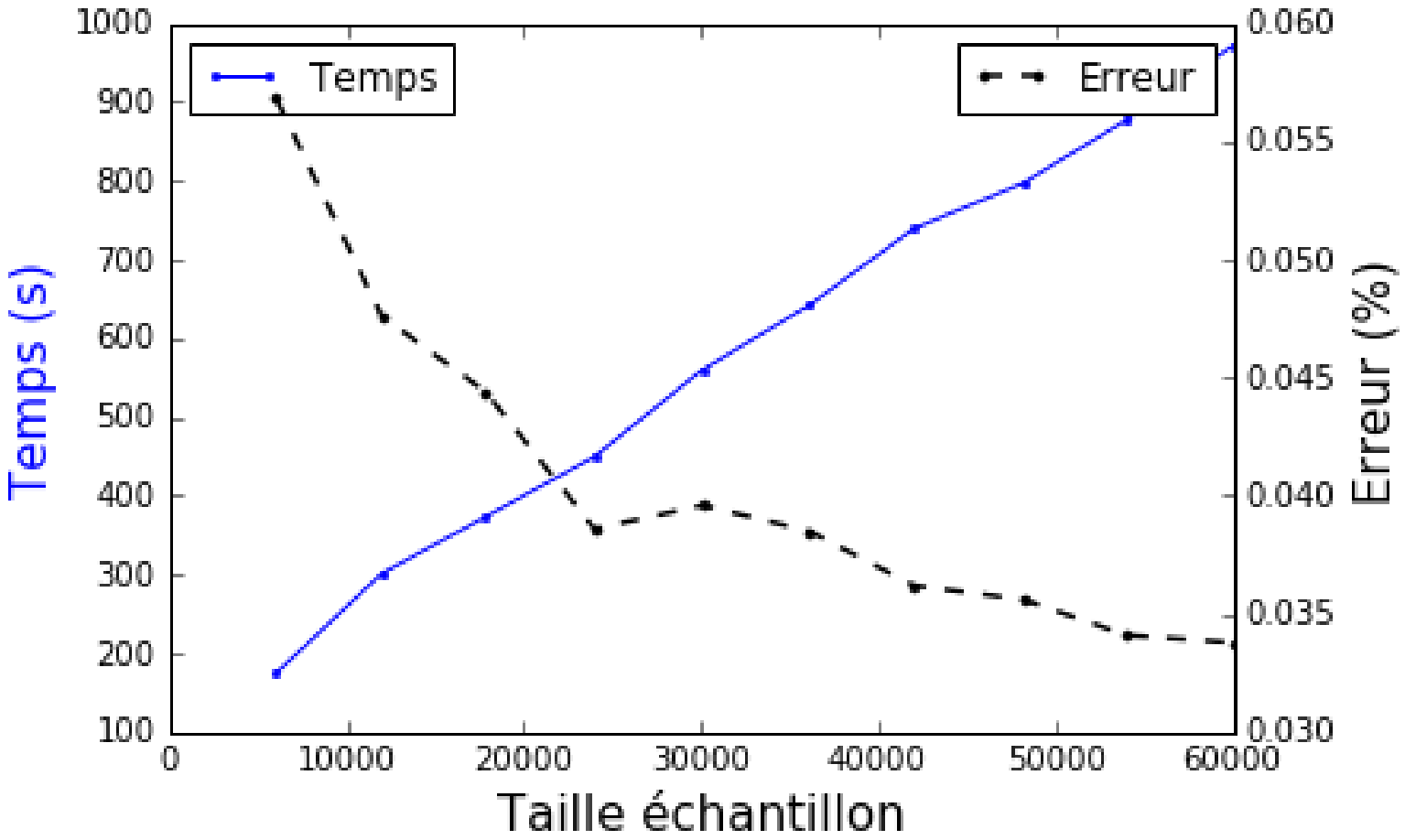
- R : `randomForest, ranger`
- Python : `Scikit-learn`
- MLlib : arbres du projet Google PLANET
  - Élagage d'un arbre
  - `maxBins = 32`
  - `maxDepth` et problèmes de mémoire
- *Bootstrap* sans remise ?



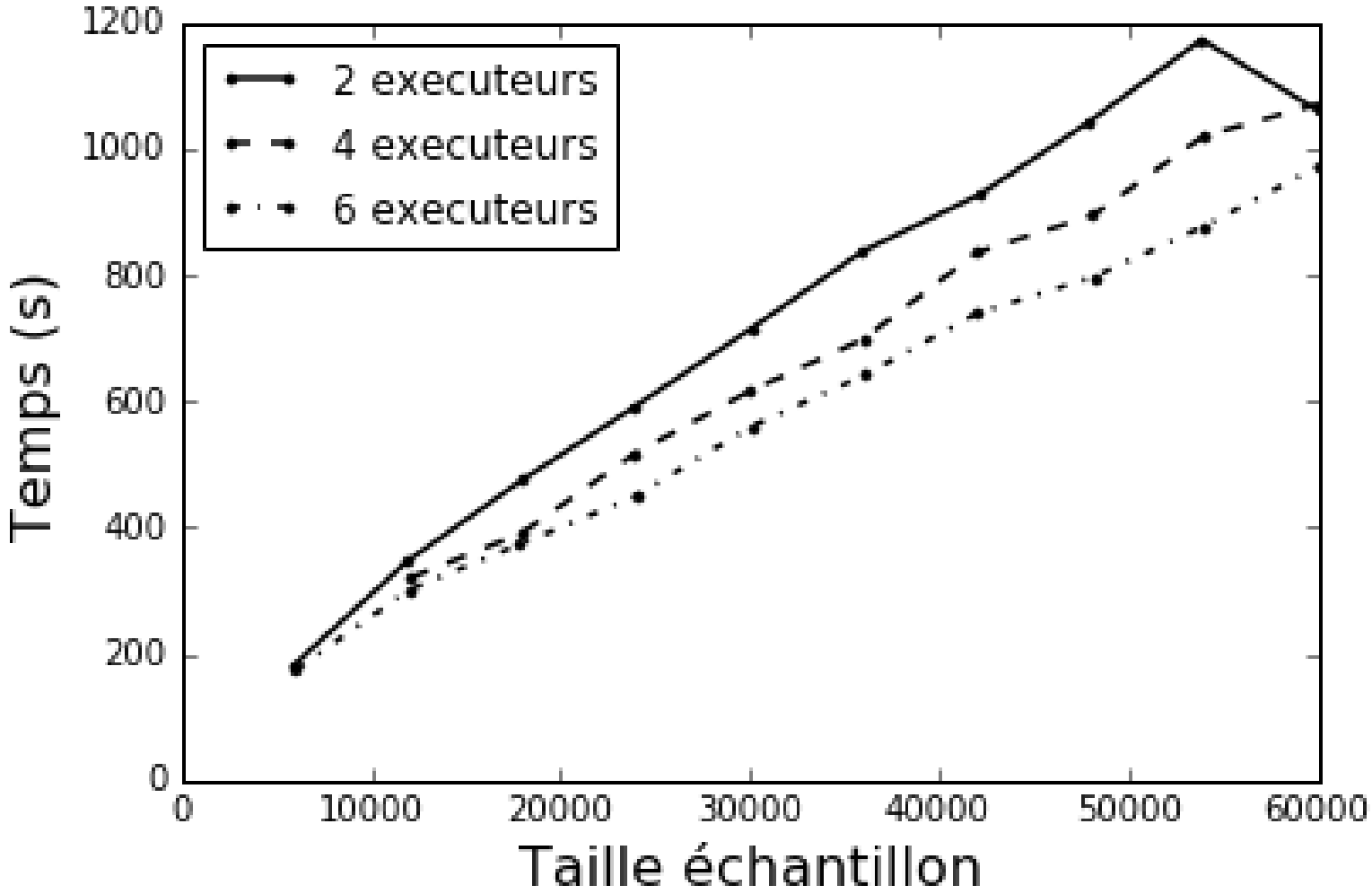
MNIST : forêts aléatoires avec R (`ranger` ***pas*** `randomForest`*)*



MNIST : forêts aléatoires avec Python (Scikit-learn)



MNIST : forêts aléatoires avec Spark (MLlib) ; `maxdepth=15`



MNIST : Forêts aléatoires (Spark) avec 2, 4 ou 6 exécuteurs

MNIST : discussion

- R `ranger` (sauf Windows) ou Python `Scikit-learn`
- Spark : problèmes de mémoire limitent `maxDpeth`, nb arbres
- Spark : *scalability* pas vérifiée
- Plateau de l'erreur fonction de la taille de l'apprentissage
- Architecture intégrée préférable à distribuée

Recommandation par filtrage collaboratif

- Matrices clients $\times$ produits très creuses
- Nombre d'achats ou de clics *vs.* appréciation ou note
- Valeur nulle *vs.* valeur manquante
- Méthodes de voisinages orientées clients *vs.* produits
- Modèle à facteurs latents par factorisation
- Factorisation *vs.* Complétion (Candes et Tao, 2010)
  $\min_{\mathbf{M}} \left(||P_\Omega(\mathbf{X} - \mathbf{M})||_2^2 + \lambda ||\mathbf{M}||_* \right)$
  où $P_\Omega(\mathbf{X})$: "projection" de la matrice $\mathbf{X}$
- Évaluation difficile d'une recommandation : RMSE
- Initialisation : *cold start*

Complétion de matrice

- *Netflix* ou *MovieLens* : problème de complétion
- Sujet à la bibliographie explosive
- Deux méthodes facilement accessibles (R et Spark)
  - Librairie R `softImpute` : Algorithme hybride associant SVDs seuillées et moindres carrés alternés (Hastie et al. 2015)

$$\min_{\mathbf{A_{n\times r}},\mathbf{B_{p\times r}}} ||P_\Omega(\mathbf{X}-\mathbf{AB}^T)||_2^2+\lambda\left(||\mathbf{A}||_2^2+||\mathbf{B}||_2^2\right),$$

  - Librairie Spark MLlib : Complétion par *Non negative Matrix Factorisation* (NMF)

Non negative Matrix Factorisation

$$\min_{\mathbf{W},\mathbf{H}\geq 0} \left[c(\mathbf{X}, \mathbf{WH}) + P(\mathbf{W}, \mathbf{H})\right]$$

- Similaire à la SVD mais avec contrainte de rang sur $\mathbf{W}, \mathbf{H}$
- c : moindres carrés ou Kulback Leibler
- P : régularisation
- Nombreux algorithmes dont ALS
- Convergence locale
- Nombreuses initialisations disponibles
- Optimiser le rang de $\mathbf{W}$ et $\mathbf{H}$
- Optimiser le coefficient de régularisation
- MLlib : deux options factorisation ou complétion : $P_\Omega(\mathbf{X})$

Données MovieLens

`100k` 100 000 évaluations de 1000 utilisateurs de 1700 films.

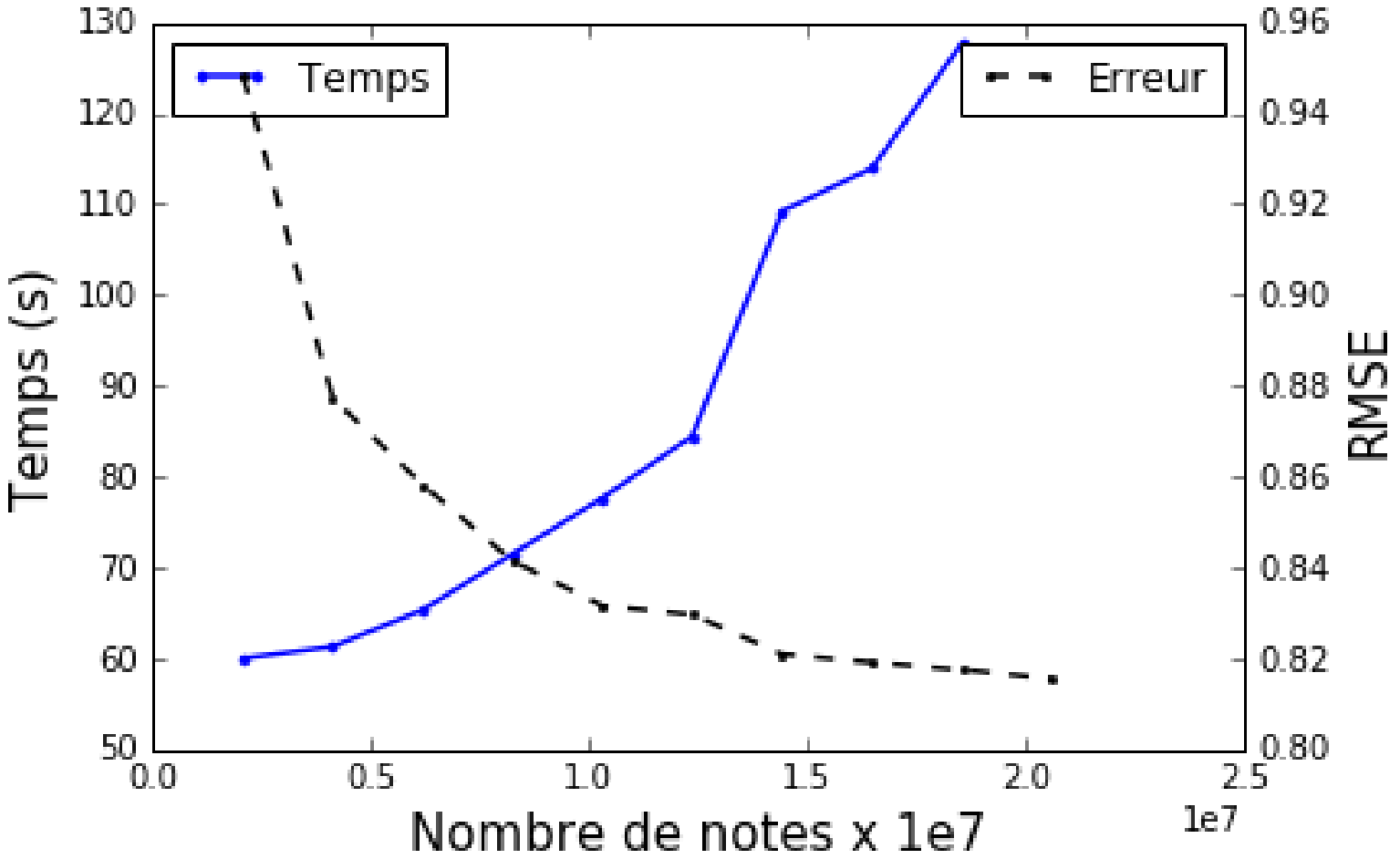
`1M` Un million d'évaluations par 6000 utilisateurs sur 4000 films.

`10M` Dix millions d'évaluations par 72 000 utilisateurs sur 10 000 films.

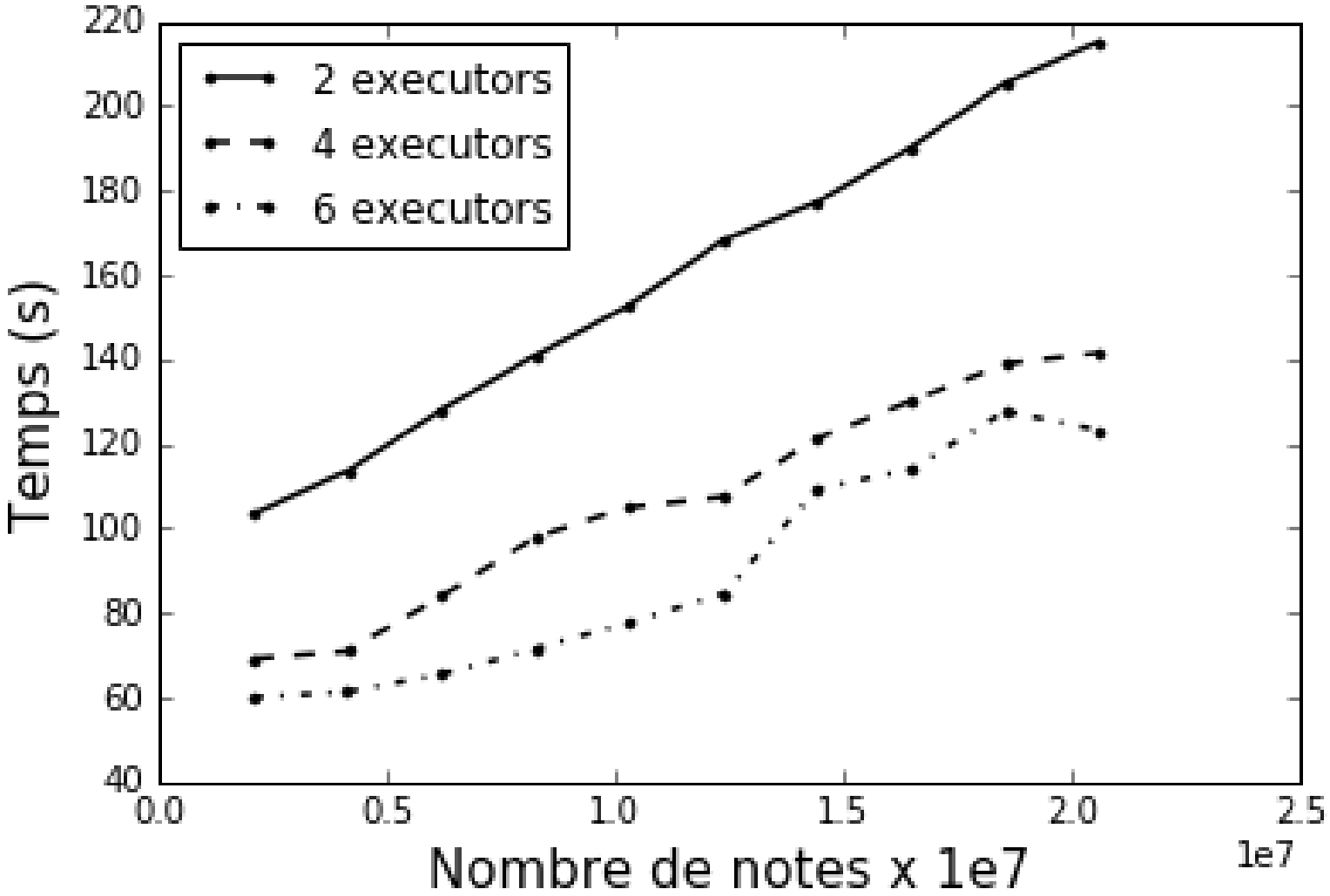
`20M` Vingt deux millions d'évaluations par 138 000 utilisateurs sur 27 000 films.

Rang Max	λ	Temps	RMSE
4	1	5.6	1.07
10	10	12.6	1.020
10	20	12.2	1.033
15	10	19.4	1.016
20	1	26.9	1.020
20	10	26.1	1.016
20	15	24.4	1.018
20	20	27.0	1.016
30	20	40.1	1.020

MovieLens : Optimisation du rang et de la régularisation de softImpute (ALS)



MovieLens : complétion par NMF (MLlib)



MovieLens : complétion par NMF (MLlib)

MovieLens : discussion

- NMF mais pas de complétion dans `Scikit-learn`
- Moins bon RMSE de *softImpute* : pas de contrainte ?
- MLlib : pas tout à fait *scalable*
- Architecture distribuée adaptée aux matrices creuses

Catégorisation de produits (Cdiscount)

- Préparation ou *data munging*
  - Nettoyages (ponctuation, erreurs de code, casse...)
  - Suppression des mots "vides" (*stopwords*)
  - Racinisation (*stemming*) : card(Dictionnaire)$= N$
- Vectorisation de (grosses ?) données
  - Hashage (*hashing trick*)) : $\texttt{n\_hash} < N$

$$i = h(j) \quad h : \{1, \ldots, N\} \longmapsto \{1, \ldots, \texttt{n\_hash}\}$$

  - Xgram : h appliquée à un mot ou couple (bigram) ou...
  - TF-IDF
- Apprentissage

TF-IDF

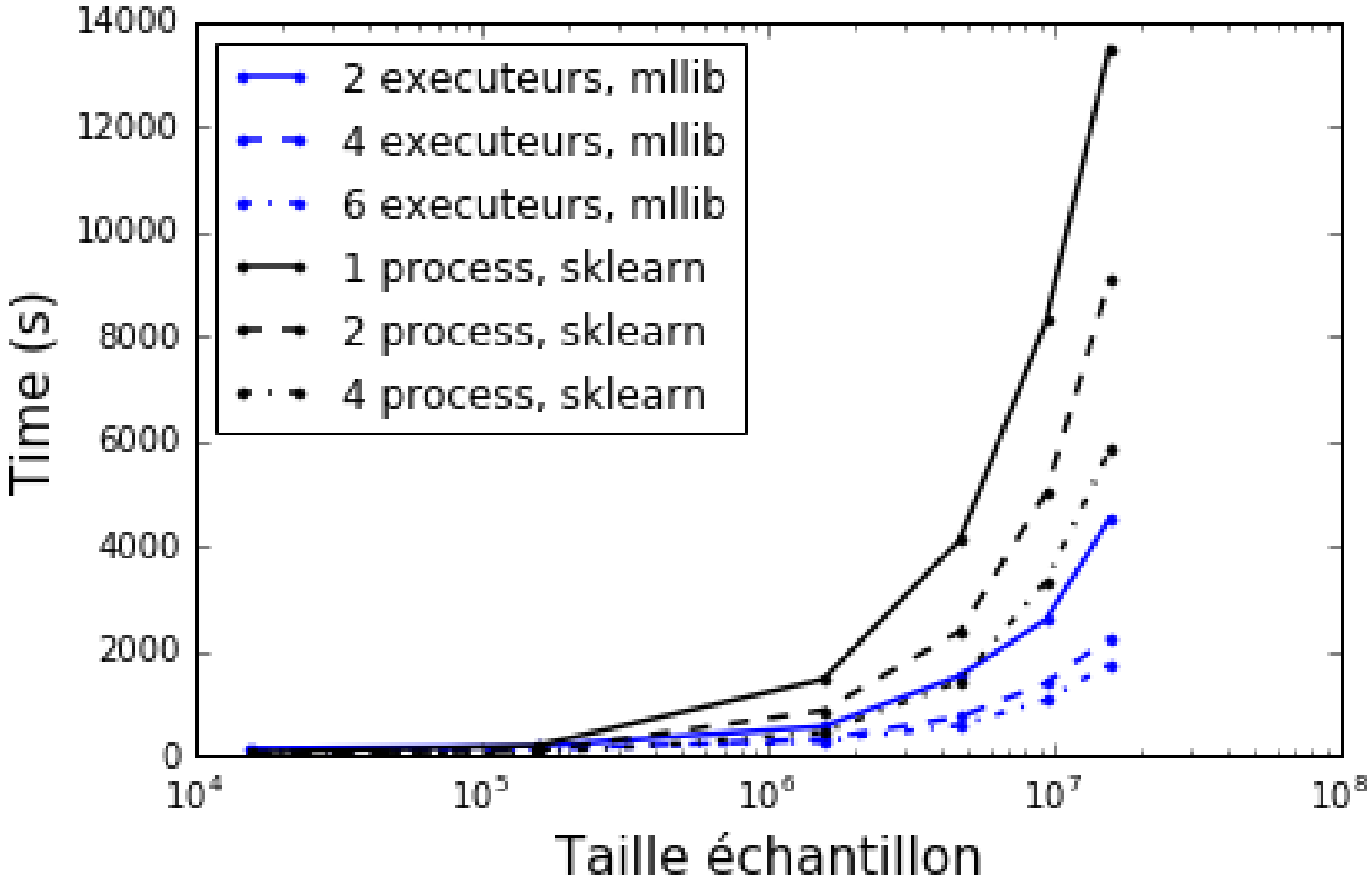
- Importance relative de chaque mot (ou xgram) dans un document par rapport à l'ensemble des documents.
- D : nombre de documents
- $TF(m,d)$: nombre d'occurrences du mot m dans le document d
- $f(m)$: nombre de documents contenant le mot m
- $IDF(m) = \log \frac{D+1}{f(m)+1}$ (version *smooth*)
- TF-IDF : nouvelles variables ou *features* par pondération des effectifs conjoints :

$$V_m(d) = TF(m,d) \times IDF(m)$$

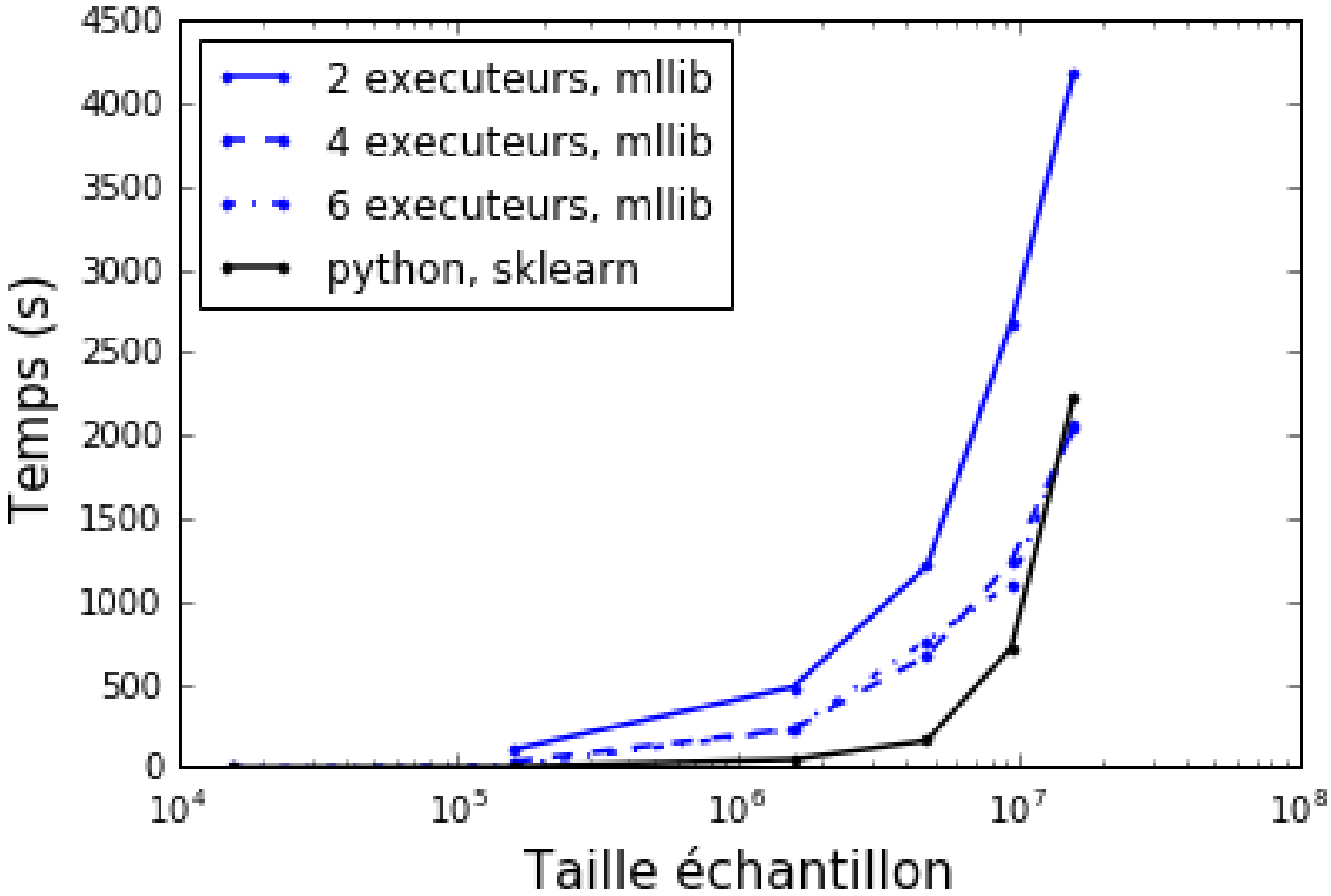
- Même vectorisation (hashage, TF-IDF) sur le test

Cdiscount

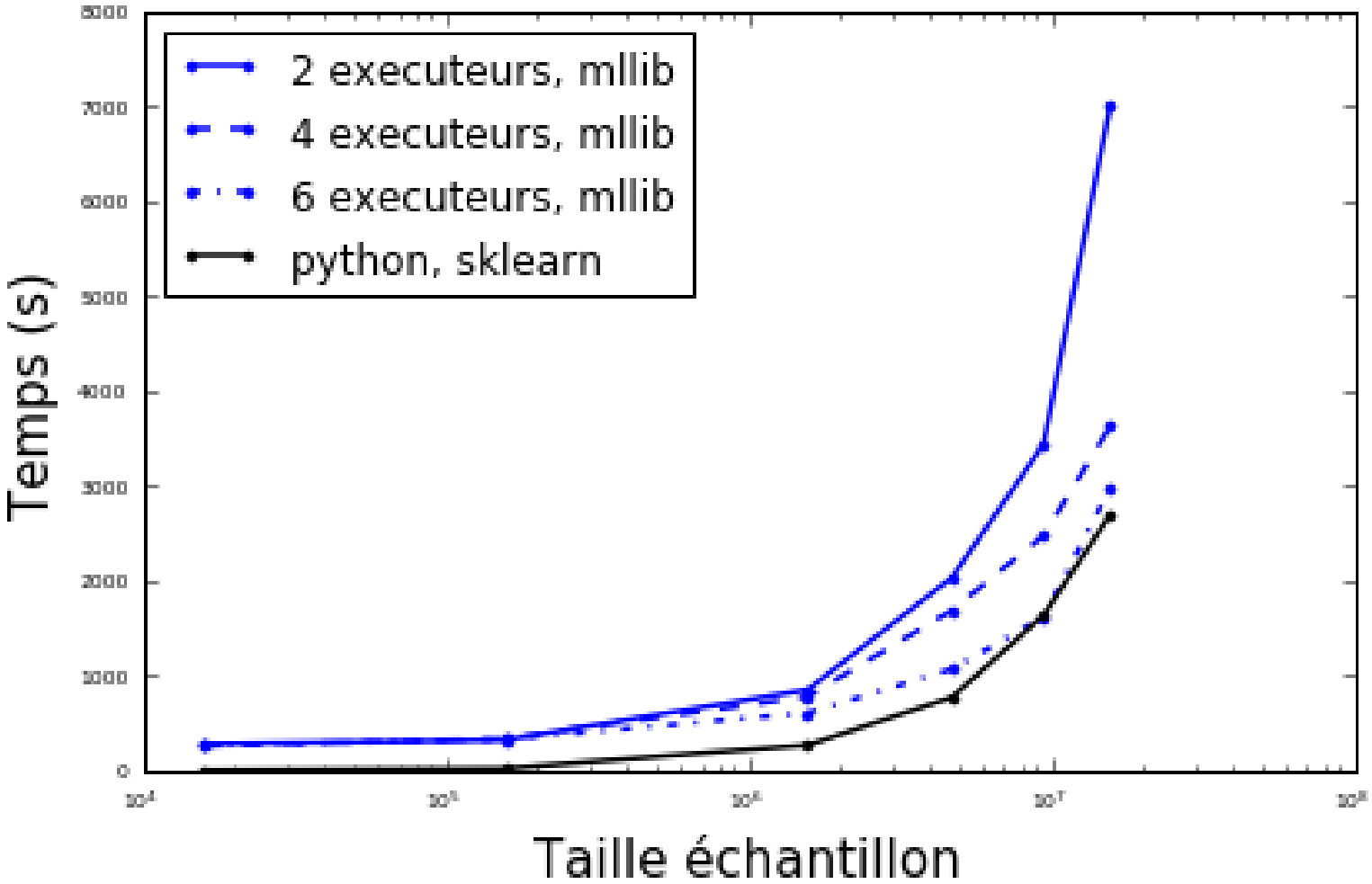
- Données publiques du concours `datascience.net`
- 15M de produits (3.5 Go), 3 niveaux : 5789 classes
- Classes déséquilibrées
- Solution gagnante (Goutorbe et al. 2016)
- Pyramide (Python) de régressions logistiques
- Simplifier : prévoir le 1er niveau : 47 classes
- Comparaison de Python Scikit-learn *vs.* SparkML
- Trois phases : Nettoyage, Vectorisation, Apprentissage



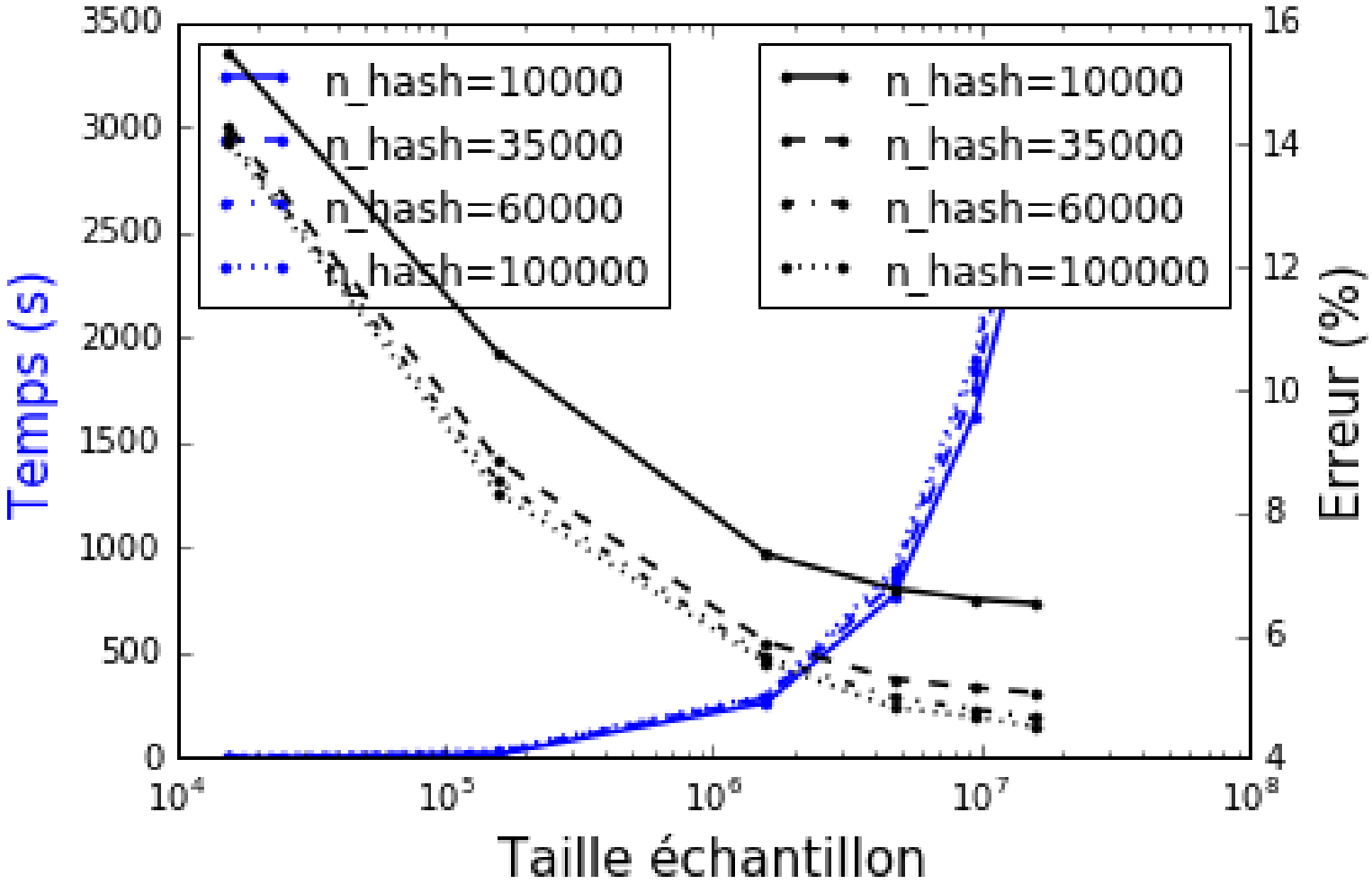
Cdiscount : nettoyage des données



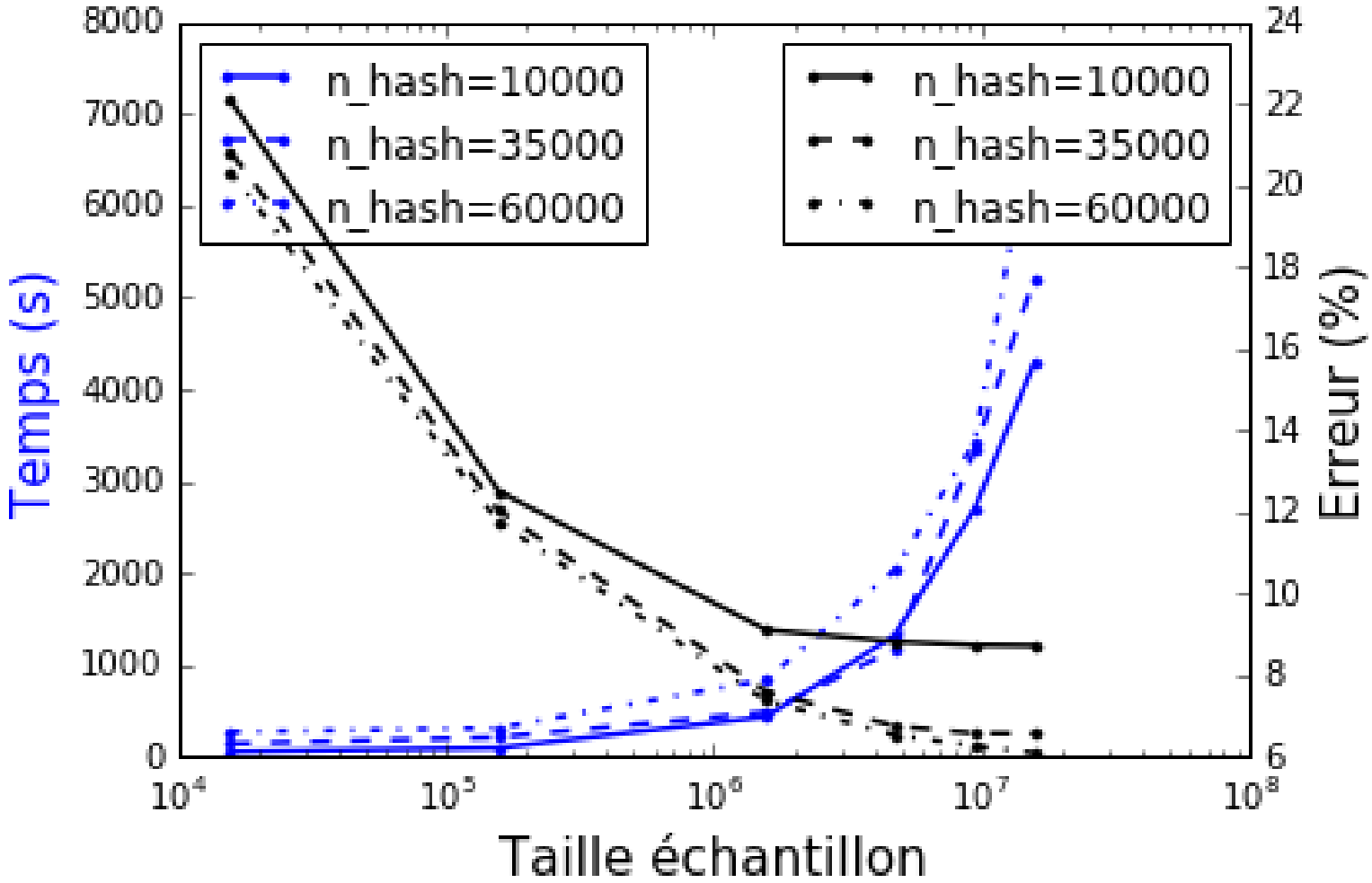
Cdiscount : vectorisation avec `n_hash` *= 60 000*



Cdiscount : apprentissage avec Spark, Python (Scikit-learn) et `n_hash` *= 60 000*



Cdiscount : Apprentissage avec Python Scikit-learn



Cdiscount : Apprentissage avec Spark MLlib

Conclusion 1

- Comparaison entre architectures distribuée *vs.* intégrée
- Problème de maturation des technologies
- Trois étapes :
  - *Data munging*, *streaming* : Spark, SparkSQL
  - Vectorisation : Python, Scikit-learn, Lucene...
  - Apprentissage : grosses data et gros modèles
- R *vs.* Python Scikit-learn *vs.* SparkSQL, SparkML
- Nettoyage et Apprentissage en ligne ?
- Cdiscount, Critéo, Deepky, Tinyclues, Hupi (ppml)...
- Ne pas oublier : fiabilité, représentativité des données

Conclusion 2

- ▸ Les Echos.fr 09-15 :Pour une intelligence artificielle sans danger pour l'humanité
- ▸ PredPol 09-14 : type, place, and time of crime
- ▸ PNAS 01-15 : Computer based personality judgments are more accurate than those made by humans
- ▸ NorthPointe Advancing Justice : Modèle de Cox
- ▸ MIT NewsRoom 04-15 : MIT Sloan professor uses machine learning to design crime (recidivism) prediction models
- ▸ Justice Prédictive 07-16 Croiser l ?ensemble des données jurisprudentielles
- ▸ ArsTechnikaUK 02-16 : The NSA's SKYNET program may be killing thousands of innocent people *"Ridiculous optimistic" machine lernaning algorithm is "completely bullshit" says expert*