

Application de K-means à la définition du nombre de VM optimal dans un cloud

The logo for EGC (Ecole Globale de Cloud Computing) consists of the letters 'EGC' in a bold, blue, sans-serif font.

**EGC 2012 : Atelier Fouille de données complexes :
complexité liée aux données multiples et massives
(31 janvier - 3 février 2012, Bordeaux, France)**

Khaled TANNIR (doctorant 2^{ème} année à l'EISTI)





Présentation

- Khaled TANNIR
 - Doctorant en 2^{ème} année à l'ESTI / laboratoire L@rys
 - (Laboratoire ETIS – Université de Cergy-Pontoise)
 - Directeur de thèse (UCP) : Dan Vodislav
 - Directeur / Encadrant EISTI : Hubert Kadima
 - Encadrant EISTI : Maria Malek

- Activité professionnelle en parallèle en tant que Consultant / Architecte Technique (MS .NET)



Sommaire

- Présentation du contexte / problématique
- L'algorithme d'optimisation
- Quelques résultats
- Conclusion et perspectives



Sommaire

- Présentation du contexte / problématique
- L'algorithme d'optimisation
- Quelques résultats
- Conclusion et perspectives



- Génération de règles d'associations avec l'algorithme « Apriori »
 - « Apriori » est un algorithme connu dans le monde de l'exploration des données pour la génération de règles d'associations à partir du panier de la ménagère. (*très consommateur en ressources*) [1]
- Prendre en charge un gros volume de données
- Exécuter dans un environnement cloud computing



- Des travaux précédents ont été proposés dans ce domaine avec une approche de partitionnement des données [2]:
- On peut constater que :
 - Ne s'exécutent pas (ou pas de façon optimale) dans un environnement cloud
 - Ne sont pas (ou difficilement) applicables à tous les cas
 - L'approche de partitionnement proposée ne s'est pas révélée toujours optimale.

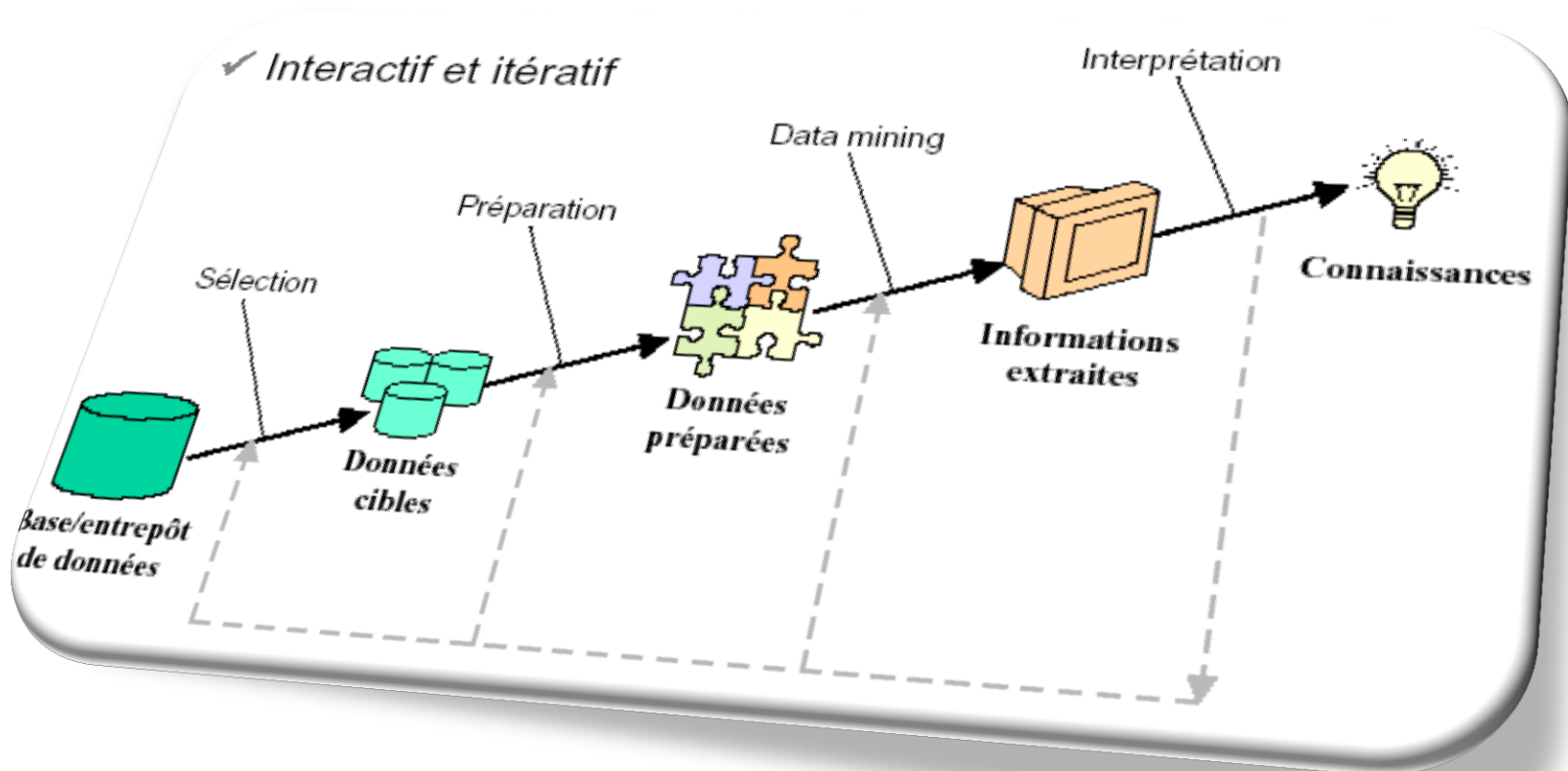


- Répondre à la problématique en proposant des algorithmes qui:
 - Partitionnent les données d'une manière plus « intelligente »
 - S'exécutent de façon optimale dans un environnement cloud
 - Etre (dans la mesure du possible) applicables à tous les cas

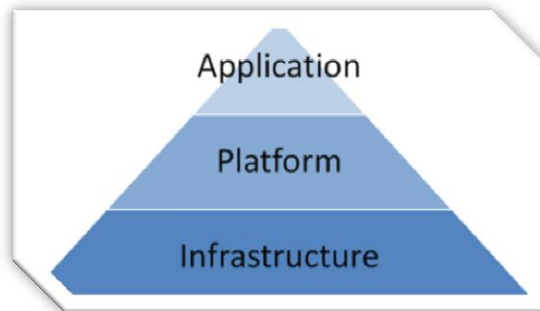


- Partitionner les données en plusieurs partitions homogènes avec un algorithme tel que k-means [3]
 - K-means / h-means pour les données qualitatives
 - K-medoids pour les données quantitatives
- Optimiser la distribution des partitions de données dans un cloud (de sorte à optimiser le nombre d'instances de VM nécessaires)

- Ensemble de techniques d'exploration de données afin d'en tirer des connaissances qui seront présentées à l'utilisateur sous forme de modèles



- Littéralement « ordinateur dans les nuages. »
- Permet d'utiliser des ressources (Mémoire, CPU, HDD...) des serveurs répartis dans le monde entier et liés par un réseau tel internet.



- Matériels:

- Environnement Serveurs Cloud privé hybride basé sur *OpenNebula* [13]
- Amazon EC2 est utilisé comme fournisseur de cloud public. [14]

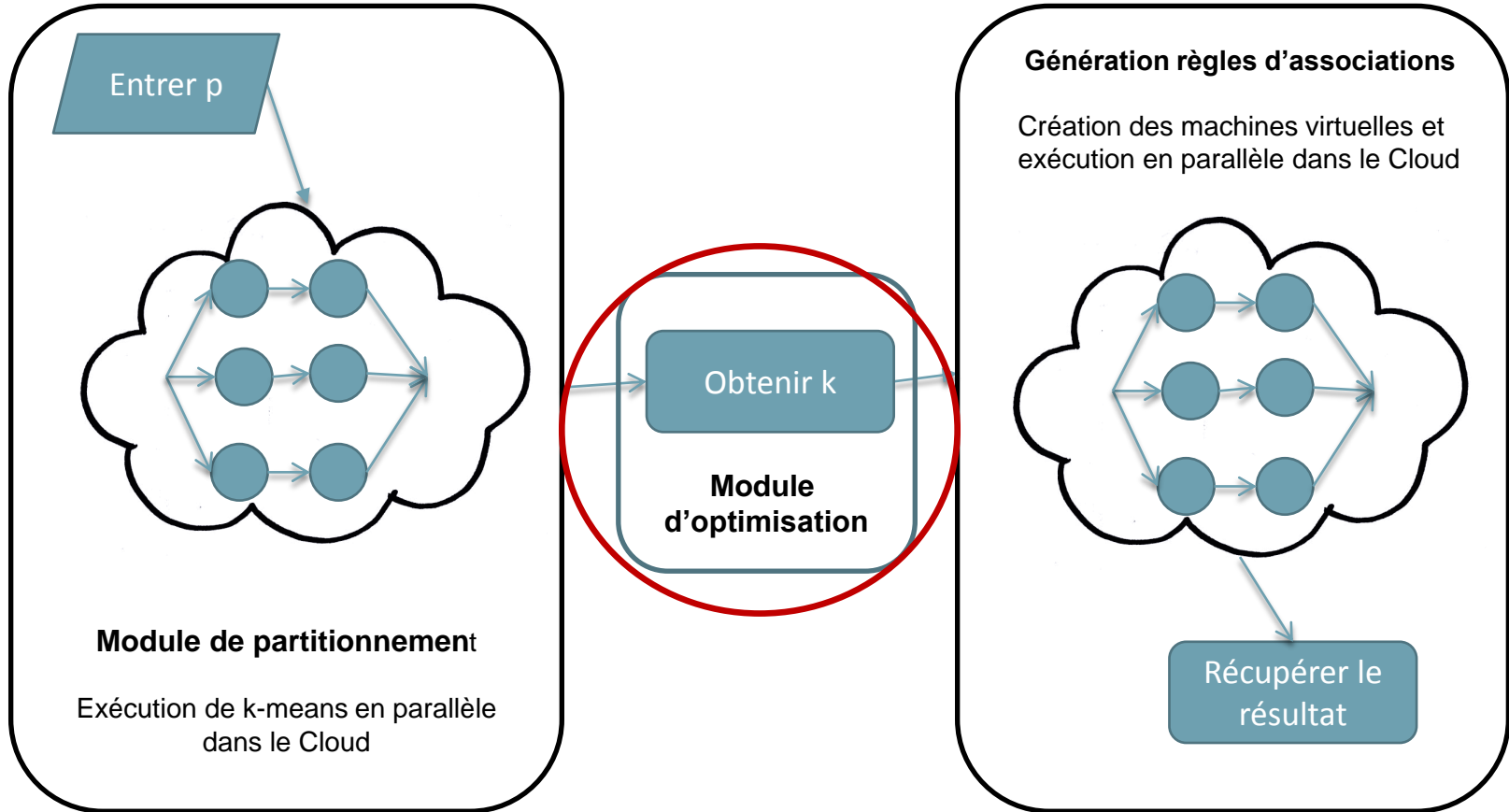
OpenNebula.org
The Open Source Toolkit for Cloud Computing





Sommaire

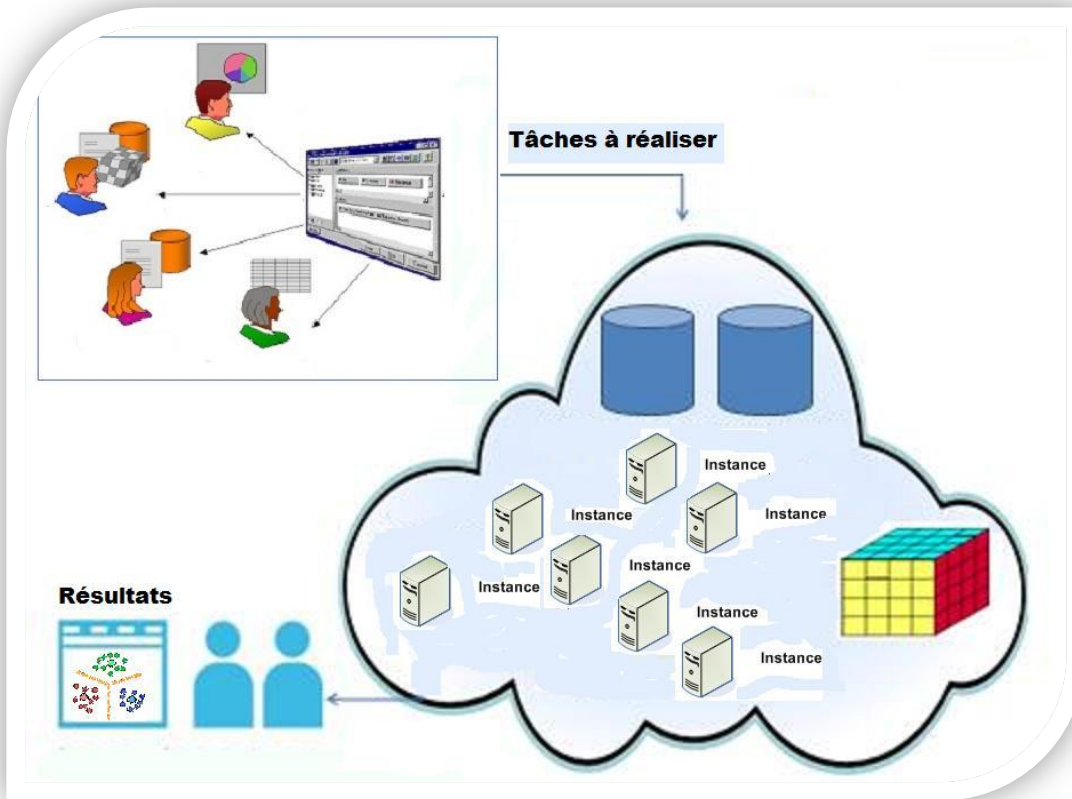
- Présentation du contexte / problématique
- **L'algorithme d'optimisation**
- Quelques résultats
- Conclusion et perspectives



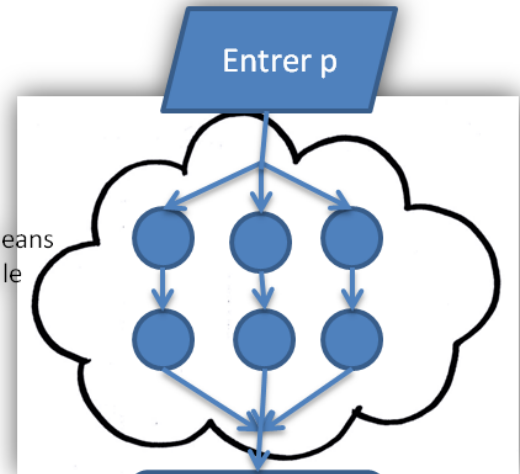
- Seul l'algorithme du module d'optimisation sera présenté



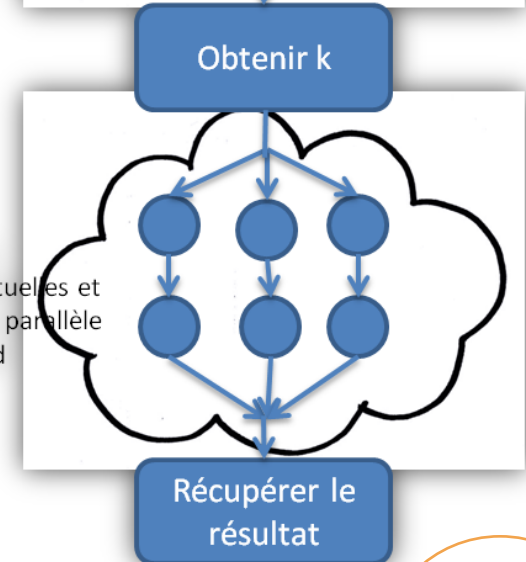
- L'algorithme d'optimisation constitue un sous-ensemble d'un algorithme plus général
- Une seule fonction principale :
 - Fournir une cartographie de la distribution des partitions dans les VM (une map)



Exécution de h-means
En parallèle dans le Cloud



Création des machines virtuelles et exécution en parallèle
Dans le Cloud





- L'algorithme d'optimisation étend l'algorithme k-means avec la méthode du « Simplexe » [10]
 - **Données en entrées :**
 - Une table indiquant la taille et l'identifiant de chaque partition.
 - Nombre de VM maximal et la capacité de chaque VM
 - **Données retournées :**
 - Une table indiquant pour chaque identifiant de partition, la VM correspondante
- **Exigences :**
 - Le nombre de VM maximal fourni doit être suffisant pour contenir les partitions à distribuer



- Le module de partitionnement nous produit:
 - n nombre de partitions (clusters)
 - a_i taille de chaque partition
(= nb de lignes * taille d'une ligne)
- L'utilisateur (ou le système) nous fournit:
 - m nombre de VM maximal
 - b la capacité d'une VM (espace disque max)
- On considère:
 - $x_i = 1$ si une partition a été distribuée, 0 sinon
 - $y_j = 1$ si une VM contient une partition, 0 sinon



- Nous cherchons à :

- minimiser le nombre des VM
- distribuer toutes les partitions sur les VM

- Il s'agit donc de :

- Minimiser $\sum y_i$ (somme des VM) sous les contraintes :

- **1** $\forall_i = 1 \dots n : \sum_{j=1}^m x_j = 1$ *(une partition doit être placée une seule fois)*

- **2** $\forall_j = 1 \dots m : \sum_{i=1}^n a_i x_{ij} \leq b \cdot y_j$ *(la capacité disuqe max de chaque VM doit être respectée)*



- **Initialisation**

- Générer les tables des capacités des VM
- Générer les tables des tailles des partitions
- Appel au module externe de la méthode simplexe
 - Définitions des variables
 - Création des tables binaires / paramètres (représentants les VM et les partitions)
 - S'assurer qu'une partition est distribuée une seule fois
 - S'assurer que les capacités des VM n'est pas dépassée
 - S'assurer que toutes les partitions ont été distribuées
 - Retourner le tableau des affectations Partitions/VM
- Retourner la liste à l'appelant

- **FIN**



Sommaire

- Présentation du contexte / problématique
- L'algorithme d'optimisation
- **Quelques résultats**
- Conclusion et perspectives



- Nous avons partitionné une base de données en 16 partitions. (les tailles sont exprimées en Mo).
- Les partitions ont été distribuées sur 3 VM disponibles ayant 1.4 Go de taille disque chacune

K	26	35	52	77	88	94	137	164	253	364	372	388	406	432	461	851
VM1					X										X	X
VM2						X	X	X	X	X		X				
VM3	X	X	X	X							X		X	X		

$$\text{VM 1 : } 88 + 461 + 851 = 1400 \text{ Mo}$$

$$\text{VM 2 : } 94 + 137 + 164 + 253 + 364 + 388 = 1400 \text{ Mo}$$

$$\text{VM 3 : } 26 + 35 + 52 + 77 + 372 + 406 + 432 = 1400 \text{ Mo}$$



Exemple de résultats retournés par l'algorithme d'optimisation (2)

Distribution de **20** partitions sur **11** VM ayant chacune une capacité maximale

K	MAX Mo	36	26	30	25	30	26	39	37	35	37	36	22	25	39	39	22	39	30	23	29	Σ
V1	60					X	X															56
V2	55			X	X																	55
V3	81								X						X							76
V4	43																					0
V5	65	X																			X	65
V6	83										X	X								X		81
V7	73																X	X				69
V8	78							X							X							78
V9	42																					0
V10	89		X							X			X									88
V11	57								X							X						57

Sur les **11** VM disponibles, **9** ont été uniquement utilisées



Sommaire

- Présentation du contexte / problématique
- L'algorithme d'optimisation
- Quelques résultats
- **Conclusion et perspectives**



Conclusion

- **Points forts:**

- Distribution efficace des partions sur les VM (optimisation de nombre de VM)
- Exécution extrêmement rapide
- Se base sur des moteurs existants pour le calcul du simplexe

- **Limites:**

- Ne prend pas en charge le dépassement des capacités lorsque le nombre de VM max est insuffisant



Perspectives

- Introduire :
 - une notion de « capacité disque minimale » d'une VM.
 - une notion de « coût » pour les instances du cloud public.
- Suggérer un nombre de VM déduit à partir de la table des partitions fournies par le module de partitionnement.



- [1] R. Agrawal and R. Srikant. Fast algorithms for mining associations rules in large databases. In Proc. of the 20th In10 Conf. on Very Large Data Bases (VLDB'94), pages 478-499, September 1994.
- [2] Valerie Fiolet. ALGORITHMES DISTRIBUES D'EXTRACTION DE CONNAISSANCES. 2006.
- [3] R. Howard. Classifying a population into homogeneous groups. J.R. Lawrence (Ed.), Operational Research in the Social Sciences, 1966.
- [4] Frank, Eibe, et al. The WEKA Data Mining Software: An Update. SIGKDD Explorations. July 2009, Vol. 11, 1.
- [5] Ian H., Witten and Eibe, Frank. Data Mining - Practical Machine Learning Tools and Techniques. [ed.] ELSEVIER. s.l. : Morgan Kaufmann Publishers, 2005.
- [6] M.J. Zaki. Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering, 12:372–390, 2000.
- [7] C. Lucchese, S. Orlando, R. Perego. Fast and Memory Efficient Mining of Frequent Closed Itemsets In IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 1, pages 21-36, January 2006.
- [8] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databses. In Proc. of the 21st VLDB Int. Conf. (VLDB'95), pages 432-444, September 1995.

URLs:

- [10] Méthode du Simplexe (http://fr.wikipedia.org/wiki/Algorithme_du_simplexe/)
- [11] IBM ILOG CPLEX Optimizer (<http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>)
- [12] <http://labs.google.com/papers/mapreduce-osdi04-slides/index-auto-0002.html>
- [13] <http://opennebula.org/>
- [14] <http://hadoop.apache.org>
- [15] <http://mahout.apache.org/>



Khaled TANNIR
contact@khaledtannir.net

MERCI de votre attention