Optimisation d'architecture de lacs de données basée sur les chaînes d'approvisionnement

Marzieh Derakhshannia, Anne Laurent, Dickson Owuor

LIRMM, Université de Montpellier, CNRS, Montpellier, France
{prenom.nom}@umontpellier.fr
http://www.lirmm.fr/

Résumé. Les lacs de données constituent une nouvelle génération de dépôts de données. Dans cet article, nous nous appuyons sur une modélisation mathématique de problèmes joints de "location-allocation" utilisés dans la conception de réseau de chaîne d'approvisionnement afin d'améliorer l'architecture des lacs de données et leur performance. Un lac de données est alors considéré comme étant une chaîne d'approvisionnement et les données du lac sont considérées comme des produits avec une durée de vie déterminée. Nous faisons l'hypothèse d'un lac géré avec la paradigme MapReduce et nous résolvons le modèle mathématique à l'aide d'algorithmes gloutons pour déterminer les optimaux de tâches à exécuter pour optimiser les performances tout en minimisant les coûts totaux.

1 Introduction

Au cours de la dernière décennie, de nombreuses organisations ont décidé d'améliorer leurs plates-formes de stockage de données afin de traiter le volume important de données (organisationnelles, de capteurs, ...) produites de manière continue Llave (2018). Les défis principaux sont liés à la préparation d'environnements efficaces pour recueillir et maintenir les données pertinentes à un haut niveau de qualité. La problématique du stockage de données n'est pas nouvelle, et les entrepôts avaient émergé dans les années 1990 pour répondre en partie. Cependant, ils supposent la définition a priori des informations et connaissances à extraire des données, ce qui n'est pas toujours le cas. On parle de *lacs de données* qui sont vus comme une nouvelle génération de systèmes de stockage de données évolutifs pour répondre à l'exigence des plates-formes de stockage de données flexibles et agiles pour les organisations Giudice et al. (2019); Fang (2015). Ils constituent une nouvelle manière de charger, stocker, traiter et visualiser les données Pasupuleti et Purra (2015).

Le lac de données contient d'énormes données structurées et non-structurées dans leurs formats natifs, non pas pour une utilisation immédiate, mais pour de futures interrogations Gorelik (2019). Les architectures de lacs de données sont créées de manière différente de celles des entrepôts de données traditionnels, et tirent parti des outils et des structures pour réduire ou éliminer les processus de préparation de données inefficaces et ingérables Loshin (2013); LaPlante et Sharma (2016); Giebler et al. (2019). Par exemple, Inmon (2016) a défini un lac de données sous la forme d'une structure constituée de différentes sous-structures (appelés

bassins). LaPlante et Sharma (2016) ont décrit l'architecture de référence de Zaloni Sharma et Safari (2018) qui est basée sur l'architecture de zone.

Dans notre travail, nous faisons l'hypothÃ¨se qu'un lac de données peut être vu comme une chaîne d'approvisionnement dans laquelle de nombreux composants sont mobilisés pour fournir des données de haute qualité à l'utilisateur final de manière systématique et optimale Derakhshannia et al. (2020).Une chaîne d'approvisionnement est un réseau intégré de divers acteurs tels que les fournisseurs, les producteurs, les distributeurs et les détaillants pour fournir les produits ou services spécifiques aux clients finaux de manière systématique Beamon (1998); Min et Zhou (2002); Daskin (2011).

Afin d'atteindre une chaîne d'approvisionnement intégrée, les gestionnaires doivent prendre des décisions en fonction des trois principaux niveaux : le niveau stratégique, le niveau tactique et le niveau pratique Chopra et Meindl (2007); Munoz et al. (2012); Beamon (1998); Zhang et al. (2014); Min et Zhou (2002). Pour cette raison, des problèmes hybrides de prise de décision multi-critères sont considérés afin de concevoir un réseau de chaîne d'approvisionnement optimisé Ahmadi Javid et Azad (2010); Zhang et al. (2014) abordant tous les niveaux en même temps Contreras et al. (2012).

Dans cet article, nous nous appuyons sur les similitudes entre lac de données et chaîne d'approvisionnement et utilisons des problèmes hybrides comme les problèmes de "location-allocation" qui sont généralement utilisés pour gérer le réseau de chaîne d'approvisionnement, afin de prendre les décisions stratégiques, tactiques et pratiques pour un lac de données.

Nous fournissons un modèle mathématique afin de minimiser les coûts totaux de conception de l'architecture de lac de données. Par conséquent, des algorithmes méta-heuristiques sont utilisés pour résoudre un problème NP-difficile comme l'algorithme glouton.

Différentes technologies et plateformes sont proposées pour implanter les architectures de lacs de données. Apache Hadoop et Spark sont les principales White (2009); Priya (2017) avec l'utilisation de MapReduce comme paradigme afin de répondre aux requêtes White (2009); Husain et al. (2011). Nous nous appuyons sur cet environnement.

2 Description et formulation du problème

Dans cette étude, nous proposons une architecture de lacs de données basée sur la structure de chaîne d'approvisionnement. La donnée est alors vue comme le produit, la zone d'ingestion de données comme le fournisseur, la zone de stockage de données comme les fabricants ou les entrepôts, la zone de traitement des données comme le distributeur et les *data scientists* comme les clients Derakhshannia et al. (2020). Nous considérons un problème de "location-allocation" afin de proposer le modèle mathématique pour l'optimisation du lac de données.

Nous considérons une architecture Apache Hadoop/Spark avc le paradigme MapReduce tel que proposé par Husain et al. (2011) afin de satisfaire les requêtes SPARQL.

Il s'agit alors de définir le nombre minimal de jobs Mapreduce et les coûts liés aux exécutions de jobs pour répondre aux requêtes (SPARQL dans leur cas).

Le job MapReduce est créé dès que le client soumet la requête White (2009).

Un flux de jobs doit alors être exécuté avec les deux fonctions Map et Reduce.

Selon les principes de problémes de "location-allocation", il faut déterminer le nombre optimal et la meilleure location d'établissements (par exemple établissements de distributeurs) pour couvrir toutes les demandes de clients dans la chaîne d'approvisionnement, la plus rapide

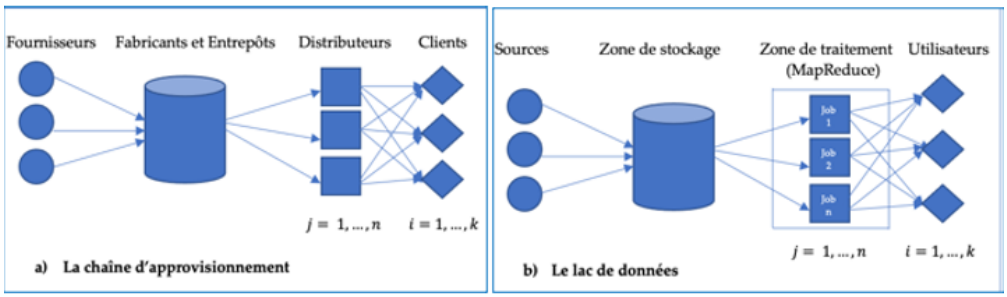

FIG. 1: Analogie entre lac de données et chaîne d'approvisionnement

et en moins de temps possible tout en minimisant les coûts totaux Melo et al. (2008); Tancrez et al. (2012).

Ce sont les jobs attribués aux requêtes des utilisateurs qui préparent les données et les mettent à disposition. La Figure 1 montre notre architecture et l'analogie entre lac de données et chaîne d'approvisionnement.

2.1 Modélisation mathématique

Dans cette section, la formulation de la programmation mathématique du problème de "location-allocation" pour le lac de données est présentée. Puisque l'un des objectifs les plus importants de l'émergence de lac de données à époque des données massives est de réduire le coût total de stockage et de traitement de données, le rôle du modèle est de minimiser le coût global de déploiement de lac de données qui est entraîné par le réseau. Les hypothèses et les variables de décisions du modèle sont expliquées ci-dessous.

Nous formulons les hypothèses suivantes.

— Les coûts de lancement d'un lac de données sont fixes.
— Le lac de données contient des jobs de MapReduce qui sont exécutés pour répondre aux requêtes.
— Le volume de données est augmenté régulièrement.
— Toutes les demandes des utilisateurs doivent être couvertes.
— La qualité des données est garantie par la gouvernance des données.
— La capacité du lac de données pour répondre aux requêtes des utilisateurs est limitée.

Nous considérons les indices I et J avec :

I L'ensemble des utilisateurs
J L'ensemble des jobs candidats de MapReduce

Les paramètres du système et les notations associées sont listées ci-dessous.

— Ca La capacité d'accès des utilisateurs
— D_i La quantité de la demande de l'utilisateur i $(\forall i \in I)$
— E Le coût du processus de extraction, chargement, et transformation de donées (ELT)
— M Le coût du logiciel de métadonnées
— S Le coût du stockage des données
— MI_j La phase d'entrée de Map pour Job j

— MO_j La phase de sortie de Map pour Job j
— RI_j La phase d'entrée de Reduce pour Job j
— RO_j La phase de sortie de Reduce de Job j
— Q Le coût de l'opération de requête
— L Le coût de latence
— A Le coût par processus analytique

Les variables de décision sont les suivantes.

$$X_{ij} = \begin{cases} 1 & si\ la\ demande\ de\ l'utilisateur\ i\ est\ couverte\ par\ le\ job\ j \\ 0 & sinon\ (\forall j \in J, \forall i \in I) \end{cases}$$

$$Y_j = \begin{cases} 1 & si\ le\ job\ j\ est\ actif \\ 0 & sinon\ (\forall j \in J) \end{cases}$$

Comme nous l'avons indiqué, nous souhaitons optimiser l'architecture de lac de données à travers une fonction objectif qui minimise les coûts du lac de données. En fait, la fonction objectif intègre les coûts des phases d'entrée et de sortie de MapReduce, des coûts de la satisfaction de la demande, et des coûts des jobs actifs. En nous basant sur la fonction objectif, les hypothèses, et les contraintes potentielles pour ce modèle, la modélisation mathématique se formalise de la manière suivante :

$$\begin{aligned} Min : \sum_{i \in I} \sum_{j \in J} X_{i,j} D_i (A + Q + E + M + S) + \\ \sum_{j \in J} Y_j (MI_j + MO_j + RI_j + RO_j) + Y_j L \end{aligned} \tag{1}$$

$Subject\ to$:

$$\sum_{i \in I} X_{i,j} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j \in J} X_{i,j} D_i \leq Ca \qquad \forall i \in I \tag{3}$$

$$X_{i,j} \leq y_j \qquad \forall j \in J\ i \in I \tag{4}$$

$$X_{i,j} \geq 0 \qquad \forall j \in J\ i \in I \tag{5}$$

$$y_j \in 0, 1 \qquad \forall j \in J \tag{6}$$

L'équation (1) minimise les coûts totaux du lac de données. L'équation (2) garantit que chaque demande d'utilisateur est entièrement assignée par le job actif. L'équation (3) représente la limitation de capacité du lac de données. L'équation (4) indique que la demande de l'utilisateur ne peut être attribuée à un job sauf si ce job est actif. Finalement, les équations (5) et (6) énoncent le type des variables de décision.

3 Méthode de résolution

Dans cette section, nous proposons une méthode méta-heuristique pour résoudre le problème d'optimisation pour lac de données. Comme nous avons mentionné, les problèmes de "location-allocation" appartiennent à la classe des problèmes NP-difficiles. Pour cette raison,

nous développons un algorithme glouton pour obtenir le montant optimal des variables de décision définis. Notre modèle détermine *"les jobs actifs j les moins chers et les plus rapides pour satisfaire la demande de l'utilisateur i"*. On suppose que la demande de l'utilisateur (i) se produit de manière aléatoire entre un intervalle de temps $(t, t+1)$ et qu'il est automatiquement associé à une *aptitude de job* $(X_{i,j})$.

Soit $\tau_j(t)$ la vitesse chronologique associée au job j au temps t et, chaque agent au temps t choisit le job qui sera au temps $t+1$. Nous définissons pour tous les choix créés par q l'aptitude des agents dans l'intervalle $(t, t+1)$ par une itération unique de notre algorithme heuristique. Par conséquent, chaque n itération de l'algorithme, chaque agent achève son choix et chaque job démarre instantanément et, à ce stade, la vitesse chronologique de job est mise à jour selon l'équation ((7)).

$$\tau_j(t+n) = \tau_j(t) + \varrho \tag{7}$$

où ϱ est un facteur dérivé du rapport entre le temps nécessaire pour terminer le job et le temps entre l'intervalle $(t, t+n)$:

$$\varrho = \frac{\Delta t_j}{\Delta(t, t+n)} \tag{8}$$

Afin de définir la probabilité de $p^k_{i,j}$ que le $k$ème agent choisisse le job j pour la demande d'utilisateur i, nous définissons *overhead* $\eta_j = 1/c_j$ où c_j est le le coût requis pour exécuter le job j). Par conséquent, $p^k_{i,j}(t)$ est décrit dans l'Equation (9).

$$p^k_{i,j}(t) = \begin{cases} \frac{\tau_j(t)\cdot\eta_j}{\sum_{k\in allowed_k}\tau_k(t)\cdot\eta_k} & \text{if } j \in allowed_k \\ 0 & otherwise \end{cases} \tag{9}$$

où $allowed_k$ est l'ensemble de jobs *'actifs'* dont $Y_j = 1$

La probabilité de $p^k_{i,j}(t)$ est un compromis entre *overhead* η_j (qui indique que le job le moins cher doit être sélectionné avec la probabilité la plus élevée) et la vitesse chronologique au temps t $\tau_j(t)$ (qui indique que le job le plus rapide jusqu'à temps t doit être choisi avec la probabilité la plus élevée).

4 Résultats d'expérimentation

Nous avons implémenté notre méthode heuristique en `Python`[1] avec un échantillon des 4 jobs choisis de manière aléatoire. L'efficacité est définie comme étant la vitesse du job actif pour répondre à la requête. La Figure 2 représente le résultat de ces expérimentations (mesures de coût et d'efficacité) .

Nous comparons l'allocation de demande à un job en utilisant un modèle équitable et en utilisant le modèle heuristique proposé, comme les Figures 3(a) et 3(b) le montrent. Le modèle heuristique alloue plus de demandes aux jobs à haute efficacitè et à des coûts relativement bas, comme le montre la Figure 3(b).

1. `https://github.com/owuordickson/dl_opt`

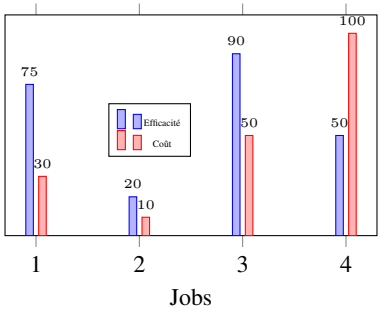

FIG. 2: Valeurs synthétiques de coût et d'efficacité pour 4 echantillons de jobs

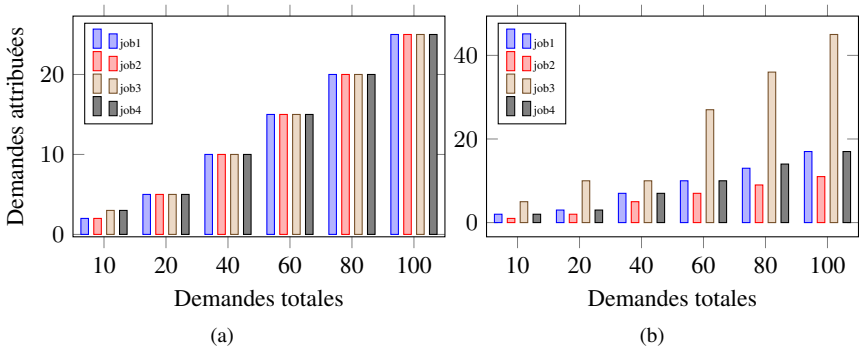

FIG. 3: Diagrammes à barres (a) Les demandes partagées de manière égale par les jobs disponibles et (b) Les demandes partagées sur la base du modèle heuristique proposé.

Nous enregistrons le temps d'exécution d'allocation du modèle heuristique (ligne continue grise) et le comparons avec le temps d'exécution d'allocation de modèle équitable (ligne pointillée bleue) dans la Figure 4 (a). Nous observons que le modèle heuristique a l'efficacité d'allocation la plus élvée. Nous observons également que les coûts calculés cumulés du modèle heuristique sont inférieurs à ceux du modèle équitable comme le montre la Figure 4(b).

5 Conclusion

Dans cet article, nous définissons un problème de "location-allocation" pour la conception d'architecture de lac de données avec le paradigme MapReduce basée sur l'analogie aux chaînes d'approvisionnement. Nous utilisons un algorithme glouton pour résoudre notre modèle. Les résultas montrent l'intérêt de cette nouvelle architecture de lac de données proposée. L'efficacité du modèle mathématique et la prise en compte des problèmes de "location-allocation" permettent d'obtenir des jobs moins coûteux et plus rapides pour répondre aux requêtes des utilisateurs.

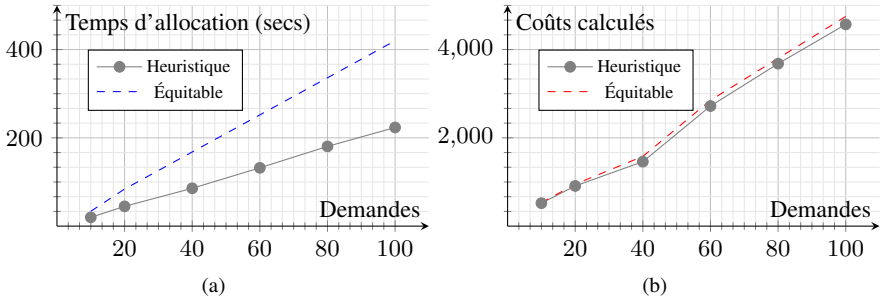

FIG. 4: (a) Diagramme de tracé de temps d'exécution d'allocation par rapport au nombre de demandes (b) diagramme de tracé de coûts calculés cumulés par rapport au nombre de jobs attribués.

Références

Ahmadi Javid, A. et N. Azad (2010). Incorporating location, routing and inventory decisions in supply chain network design. *Transportation Research Part E : Logistics and Transportation Review 46*(5), 582 – 597.

Beamon, B. M. (1998). Supply chain design and analysis : : Models and methods. *International Journal of Production Economics 55*(3), 281 – 294.

Chopra, S. et P. Meindl (2007). Supply chain management. strategy, planning & operation. In *Das summa summarum des management*, pp. 265–275. Springer.

Contreras, I., E. Fernandez, et G. Reinelt (2012). Minimizing the maximum travel time in a combined model of facility location and network design. *Omega 40*(6), 847 – 860. Special Issue on Forecasting in Management Science.

Daskin, M. (2011). *Network and Discrete Location : Models, Algorithms, and Applications*. 9780471018971.

Derakhshannia, M., C. Gervet, H. Hajj-Hassan, A. Laurent, et A. Martin (2020). Data lake governance : Towards a systemic and natural ecosystem analogy. *Future internet 12*(8), 126.

Fang, H. L. (2015). Managing data lakes in big data era : What's a data lake and why has it became popular in data management ecosystem. *IEEE Int. Conf. on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)* ,(15503297), 820–824.

Giebler, C., C. Groger, E. Hoos, H. Schwarz, et B. Mitschang (2019). Leveraging the data lake - current state and challenges. *Springer Nature 11708*, 179–188.

Giudice, P. L., L. Musarella, G. Sofo, et D. Ursino (2019). An approach to extracting complex knowledge patterns among concepts belonging to structured, semi-structured and unstructured sources in a data lake. *Information Sciences 478*, 606 – 626.

Gorelik, A. (2019). *The Enterprise Big Data Lake : Delivering the Promise of Big Data and Data Science*. O'Reilly Media.

Husain, M., J. Mcglothlin, M. Masud, L. Khan, et B. Thuraisingham (2011). Heuristics-based query processing for large rdf graphs using cloud computing. *Knowledge and Data Engineering, IEEE Transactions on 23*, 1312 – 1327.

Inmon, B. (2016). *Data Lake Architecture : Designing the Data Lake and avoiding the garbage dump*. Technics publications.

LaPlante, A. et B. Sharma (2016). *Architecting Data Lakes*. O'Reilly Media.

Llave, M. R. (2018). Data lakes in business intelligence : reporting from the trenches. *Procedia Computer Science 138*, 516 – 524. CENTERIS.

Loshin, D. (2013). Chapter 5 - data governance for big data analytics : Considerations for data policies and processes. In D. Loshin (Ed.), *Big Data Analytics*, pp. 39 – 48. Morgan Kaufmann.

Melo, T., S. Nickel, et F. Saldanha-da Gama (2008). Network design decisions in supply chain planning. *http ://publica.fraunhofer.de/documents/N-82092.html 140*(1434-9973), 29 pp.

Min, H. et G. Zhou (2002). Supply chain modeling : past, present and future. *Computers & Industrial Engineering 43*(1), 231 – 249.

Munoz, E., E. Capon, J. M. Lainez, M. Moreno-Benito, A. Espuna, et L. Puigjaner (2012). Operational, tactical and strategical integration for enterprise decision-making. In *European Symposium on Computer Aided Process Engineering*, Volume 30 of *Computer Aided Chemical Engineering*, pp. 397 – 401. Elsevier.

Pasupuleti, P. et B. S. Purra (2015). *Data Lake development with Big Data : explore architectural approaches to building Data Lakes that ingest, index, manage, and analyze massive amounts of data using Big Data technologies*. Packt Publ.

Priya, C. P. (2017). *Apache Spark for data science cookbook : over insightful 90 recipes to get lightning-fast analytics with Apache Spark*. Birmingham : Packt Publishing.

Sharma, B. et a. O. M. C. Safari (2018). *Architecting Data Lakes, 2nd Edition*. O'Reilly Media, Incorporated.

Tancrez, J.-S., J.-C. Lange, et P. Semal (2012). A location-inventory model for large three-level supply chains. *Transportation Research Part E : Logistics and Transportation Review 48*(2), 485 – 502.

White, T. (2009). *Hadoop : The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

Zhang, Y., M. Qi, L. Miao, et E. Liu (2014). Hybrid metaheuristic solutions to inventory location routing problem. *Transportation Research Part E : Logistics and Transportation Review 70*, 305 – 323.

Summary

Data lakes have recently emerged as a new generation of data repository. Data lake architecture design, which has significant impacts on data lake performance and data quality, is an active topic. In this paper, we study a joint "location-allocation" problem which is used in supply chain network design for improving data lake architecture and performance. we propose a mathematical model applied to a MapReduce environment, based on an analogy between data lakes and supply chain. We solve this model with a greedy algorithm and determine the optimal numbers of MapReduce jobs that should be run in such a data lake to optimize the performance.