BigBio: Utiliser les techniques de gestion du Big data pour les données de la Biodiversité

Ndiouma Bame[1 2] — Hubert Naacke[2] — Idrissa Sarr [1] — Samba Ndiaye[1]

[1] Département de mathématique-informatique, Université Cheikh Anta Diop, Dakar, SENEGAL
prénom.nom@ucad.edu.sn

[2] Sorbonne Universités, UPMC Univ Paris 06, LIP6, Paris, France
prenom.nom@lip6.fr

RÉSUMÉ. Le phénomène de Big data est de plus en plus perçu comme l'un des grands défis informatiques de la décennie en cours. Les perspectives du traitement des Big data sont énormes et concernent des domaines telles que la génomique, les changements climatiques, la gestion des réseaux énergétiques complexes et l'écologie. C'est dans ce contexte que la gestion des données du GBIF, qui vise à fédérer et partager les données de biodiversité à l'échelle mondiale, devient un problème crucial à résoudre avec efficacité. Dans nos précédents travaux, nous avons proposé des solutions permettant de décentraliser l'architecture du GBIF et daccroître l'expressivité des requêtes sur les données du GBIF. Cependant, le problème du passage à l'échelle n'ayant pas été pris en compte, nous l'abordons ici. Dans cette perspective, nous proposons une solution centrée autour de deux aspects : 1) une infrastructure de type cloud permettant de fédérer une quantité croissante de ressources, 2) un modèle de fragmentation et de réplication dynamique tirant profit des spécificités et du mode d'utilisation des données de biodiversité. Nous avons implémenté et simulé notre approche d'optimisation de requêtes et avons mesuré les performances et les résultats obtenus sont prometteurs.

ABSTRACT. The phenomenon of Big data is increasingly seen as one of the major challenges of this decade in both computer science and information system. The Big data issues are multiple and cover areas such as genomics, climate change, management of complex energy systems and ecology. Therefore, the management of the data of the biodiversity stored and shared through the GBIF portal is becoming a paramount issue. In our previous works, we proposed a non-invasive and decentralized architecture for processing GBIF queries over a cloud infrastructure. However, such solutions did not take into account the scalability issue that is crucial in the realm of Big data and which we tackle below. In this respect, we rely on a cloud-based infrastructure in order to partition and replicate data while leveraging on the user access patterns. We assess our solution via a prototype implementation and results obtained through experiments show the effectiveness of our solution.

MOTS-CLÉS : masses de données, répartition dynamique de données, cloud computing, modèle de coût, traitement de requête,GBIF.

KEYWORDS : big data, dynamic data distribution, cloud computing, cost model, query processing, GBIF.

1. Introduction

Le Global Biodiversity Information Facility (GBIF) est un projet international visant à fédérer et partager les données de biodiversité à l'échelle mondiale. Sa base de données est alimentée continuellement par ses correspondants nationaux. La quantité de données est passée de 163 millions d'enregistrements à 430 millions en février 2014 [10]. Le volume important, la variété (du gêne à l'écosystème) et la croissance rapide (des données mobilisées) des données de biodiversité les classent dans le domaine du big data. Le but du GBIF est de permettre le partage des données de la biodiversité à l'échelle planétaire tout en facilitant leur évolution en termes de taille et de type. Malheureusement, l'architecture actuelle du GBIF ne permet pas d'atteindre cet objectif car, en dehors d'être centralisée, elle ne supporte pas une grande variété de requêtes utiles à l'analyse des données de biodiversité. Dans nos précédents travaux [4, 5], nous avons proposé des solutions permettant de décentraliser l'accès aux données du GBIF et d'accroître l'expressivité des requêtes. Cependant, le problème du passage à l'échelle qui est au cœur du big data n'a pas été pris en compte. Dans cette perspective, nous proposons une solution centrée autour de deux aspects :
(i) une infrastructure de type cloud permettant de fédérer une quantité croissante de ressources aussi bien pour le stockage que pour le calcul [1, 6, 3, 12] ;
(ii) un modèle de fragmentation et de réplication dynamique, tirant profit de la structure des données de biodiversité et de leur mode d'utilisation [8]. En effet, la plupart des analyses spécifient des critères délimitant une zone géographique, une plage de temps et un ensemble de nœuds de la hiérarchie taxinomique. Par conséquent, il devient intéressant de concevoir un mécanisme de fragmentation/réplication qui s'adapte aux requêtes d'analyse de l'utilisateur afin, d'une part, de répliquer davantage les données fréquemment sollicitées et, d'autre part, d'ajuster le grain de fragmentation à celui de l'analyse. En outre, puisqu'une réplication totale de la base du GBIF est quasi impossible à cause de sa taille, la stratégie de la fragmentation semble être une solution viable.

A notre connaissance, les solutions existantes les plus avancées pour l'analyse de données à large échelle (Shark [11], Tenzing [7] et AsterixDB [2]), n e permettent ni d'adapter le degré de réplication des données ni de gérer une fragmentation hiérarchique. De plus, ces solutions supposent que toutes les ressources existantes sont disponibles pour une requête, ce qui n'est pas souvent le cas lorsque de nombreux utilisateurs sollicitent conjointement les mêmes ressources. Les solutions basées sur MapReduce et Hadoop n'offrent pas de bonnes performances lorsque la quantité de données traitées par une requête est petite. En outre, elles utilisent une fonction de hachage globale qui décide de l'emplacement futur de chaque fragment. De ce fait, elles ne s'adaptent pas pour un modèle de répartition de données à la demande que nous comptons proposer pour l'analyse des données de la biodiversité. Plus précisément, nous envisageons une solution avec une fonction de coût dynamique permettant de savoir les meilleurs fragments à utiliser et/ou à créer pour résoudre une requête.

La section 2 présente notre proposition d'architecture décentralisée. La section 3 montre notre stratégie de répartition et réplication des données à la demande. La section 4 présente notre modèle de coût dynamique. La section 5 montre la validation expérimentale, la section 6 conclue.

2. Architecture modulaire

Nous détaillons les composants du service de requête présenté en [5]. L'intérêt de cette nouvelle

architecture est d'assurer le passage à l'échelle des traitements. La figure 1 montre les principaux composants et leurs interactions.

– **Le gestionnaire des accès au GBIF (GBIF Access Manager)** coordonne les accès au GBIF pour lire les nouveaux fragments et gérer leur mise à jour.

– L'**index local (Local index)** maintient les informations de localisation de chaque fragment qu'un site local héberge et traite.

– L'**index global** maintient les informations de localisation de tous les fragments.

– Le **gestionnaire des requêtes utilisateurs (User Manager)** sert d'interface aux utilisateurs. Il analyse, optimise et coordonne les requêtes.

– Le **gestionnaire locale (Local Manager)** gère l'exécution des requêtes pour une base de données locale.

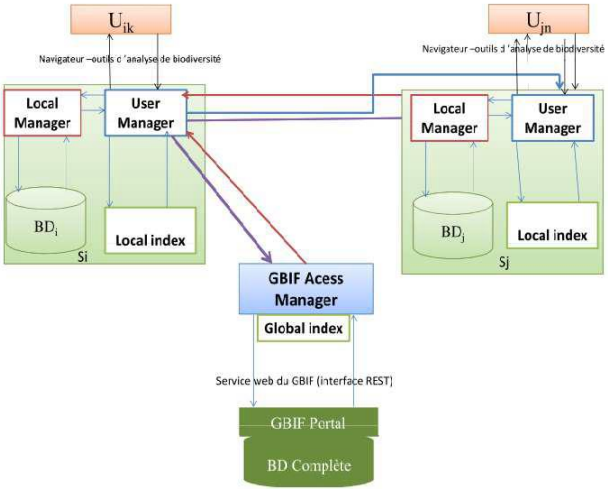

Figure 1. *Architecture modulaire pour traiter les requêtes*

3. Fragmentation et répartition dynamique des données

L'objectif de notre stratégie de distribution des données est de garantir leur disponibilité même si les capacités de stockage des sites seraient limitées. Pour cela, nous fragmentons les données en fonctions de la nature des demandes, les plaçons sur les sites où les demandes ont été soumises et répliquons celles qui sont le plus sollicitées dans les requêtes. L'analyses de données de biodiversité a la particularité de combiner des éléments des dimensions taxinomique, géographique et temporelle. En outre, les données de biodiversité ont une organisation hiérarchique selon ces dimensions. Nous exploitons cette particularité des analyses et la structure hiérarchique des données pour partitionner les données du GBIF.

3.1. Fragmentation hiérarchique des données de biodiversité

Les données de biodiversité ont une structure hiérarchique selon les dimensions taxinomique, géographique et temporelle. Les dimensions taxinomique et géographique sont les plus fréquemment utilisées dans les analyses de données de biodiversité (extinction, distribution, migration, co-occurrence, modélisation de niche écologique d'une espèce, d'un genre, d'une famille dans une zone géographique (région, pays, continents). La classification taxinomique présente une structure hiérarchique dans laquelle un ensemble d'occurrence compose une espèce (ou sous-espèce) qui avec d'autres espèces forment un genre appartenant à une famille. Le processus se poursuit jusqu'à l'écosystème. De la même façon, les données géographiques présentent une organisation hiérarchique où chaque point d'observation fait parti d'une zone qui appartient à une région d'un pays. Un ensemble de pays forme un continent. L'ensemble des continents forme le globe. En termes de maillage, un ensemble de petits rectangles (ou mailles) délimité chacun par des longitudes minimale et maximale et des latitudes maximale et minimale forme une maille plus large et ainsi de suite.
Ainsi, nous définissons un fragment traité par une requête comme étant le résultat de la conjonction des prédicats portant sur ces deux dimensions. Les autres prédicats ignorés dans le processus de fragmentation pourront être appliqués au fragment en question lors du traitement de la requête. Un fragment est défini par la conjonction de nœuds dans la hiérarchie des dimensions taxinomique et géographique. Cette structure hiérarchique des fragments permet de constater une relation de spécialisation/généralisation. Ce qui fait que bien que deux fragments n'aient pas les mêmes prédicats de définition, ils peuvent avoir une relation d'inclusion où le fragment le plus général inclut le plus spécifique. De ce fait, le premier peut être utilisé pour des traitements sur le fragment spécifique. Ce qui favorise la réutilisation des fragments.

3.2. Réplication des fragments à la demande

Pour déterminer les fragments à répliquer, nous nous basons sur les prédicats des requêtes qui sont soumises à partir du site concerné. Une telle réplication à la demande, permet de minimiser les coûts de communication avec le GBIF tout en favorisant le parallélisme des accès et des traitements. Un fragment qui n'est pas encore disponible dans les sites locaux, sera placé au premier site où une requête l'invoquant a été soumise. Des réplications de fragments d'un site local vers un autre peuvent se produire au cours de l'exécution d'une requête. En effet, pour les plans d'exécution qui nécessitent que toutes les données soient en un seul endroit, les fragments nécessaires qui ne se trouvent pas dans le site coordinateur seront transférés vers celui-ci où ils seront insérés (répliqués). Ces fragments pourront être utilisés lors des traitements des prochaines requêtes. Nous verrons en détail ce mécanisme de réplication de fragment au cours du traitement dans la section 4.

4. Traitement des requêtes : modèle de coût dynamique

Le coût d'une requête dépend de plusieurs paramètres : localisation des données, le choix de la réplique à traiter (parmi plusieurs autres répliques du même fragment), le coût de transfert, les charges et performances des nœuds. La réduction du coût d'une requête répartie résulte d'un mécanisme efficace d'optimisation qui prend en compte tous les paramètres susceptibles de l'affecter. Dans cette section, notre objectif consiste à traiter une requête rapidement, quelles que soient la capacité et la charge du site où elle a été soumise. Ce qui revient à minimiser le temps de réponse de

chaque requête. Le contexte de notre solution présente la particularité que toutes les données sont initialement sur un seul site, le portail GBIF. De ce fait, notre stratégie de traitement de requêtes contrôle aussi la distribution des données à travers les sites locaux. Intuitivement, nous devons atteindre un état tel que tous les sites participent autant au traitement des requêtes. Cela peut nécessiter de répliquer certaines données accédées fréquemment. Remarquons qu'il n'est pas souhaitable de répliquer toutes les données sur chaque site, car un site ne dispose pas de capacités de stockage suffisantes pour stocker l'ensemble de la base, et car le surcoût d'ajouter des données dans un site est très élevé ce qui pénalisera le traitement des requêtes pendant la période d'ajout des données. La stratégie que nous proposons tend à répliquer partiellement les données selon les principes suivants :
- Si une donnée n'existe pas parmi les sites, alors importer la donnée depuis le portail GBIF.
- Le site désigné pour traiter une requête est celui dont le coût estimé est minimal, sauf exception décrite ci-dessous.
- Si un site demeure faiblement utilisé par rapport aux autres sites, alors il sera désigné en priorité pour traiter les requêtes dont les données se situent sur le site le plus utilisé. Cela implique un surcoût temporaire pour ajouter des données qui sera amorti rapidement grâce au bénéfice apporté sur le temps d'exécution des requêtes.
Par la suite, nous montrons la manière dont les sites locaux collaborent lors du traitement des requêtes et détaillons notre stratégie d'optimisation qui a pour but de réduire le temps de réponse.

4.1. Mécanisme d'exécution de requêtes

La figure 2 illustre les étapes d'exécution d'une requête. A la réception d'une requête utilisateur, le User Manager (noté UM) analyse la requête pour déterminer les fragments à accéder, en considérant aussi les fragments pouvant inclure d'autres fragments. Il détermine les localisations des fragments impliqués, en consultant l'index local (ou global si nécessaire). Puis il détermine un plan optimal décrivant les opérations et les fragments à accéder sur chaque site. Le UM (noté QUM) qui a reçu, analysé et optimisé la requête peut transmettre le plan à un autre UM désigné pour coordonner le traitement (et noté CUM). Celui-ci envoie les sous-requêtes correspondantes au Local Manager (LM) de chaque site impliqué et/ou invoque les services web pour télécharger les fragments qui existent uniquement sur le portail du GBIF. A la réception de tous les résultats partiels, le CUM les insère dans la base locale (réplication) et soumet la requête complète au LM du même site. A la fin de l'exécution, il reçoit le résultat final du LM et le retourne directement à l'utilisateur.

Nous avons complété l'exécution des requêtes avec un mécanisme de ré-ordonnancement des requêtes visant à éviter qu'un LM devant traiter une requête ne soit inactif pendant l'attente des données provenant des sites distants. Le LM estime sa durée d'attente (D) et choisi de traiter une prochaine requête (parmi celles qu'il a reçues) dont le temps de traitement estimé est inférieur à D. De la même façon, le CUM peut recevoir d'autres requêtes issues des utilisateurs ou des voisins pour analyser, optimiser et/ou coordonner leurs traitements. Ce qui permet de « paralléliser localement » et d'éviter les éventuels blocages entre les sites afin d'exploiter au mieux les ressources disponibles. En outre, toute requête qui nécessite un accès à la base du GBIF est traitée (coordonnée) par le site où elle a été soumise. Ceci permet de placer initialement les données les fragments du GBIF dans les sites où ils sont demandés.

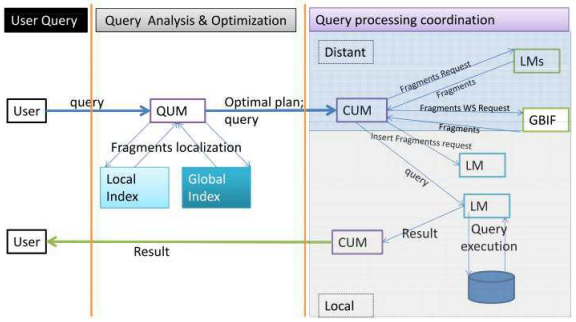

Figure 2. *Principe de traitement de requête*

4.2. Optimisation basée sur le coût

Le processus d'optimisation prend en entrées la requête utilisateur (opérations et données), les localisations (et leurs caractéristiques) de chaque fragment (et réplique). Il explore plusieurs plans possibles pour l'exécution de la requête, envisageant que chaque site soit le coordinateur de la requête. Parmi les plans explorés, il détermine le plan optimal (i.e., de coût minimal) pour le traitement de la requête. L'exécution du plan de moindre coût doit permettre de réduire le temps de réponse des requêtes, ce que nous vérifierons expérimentalement.

Algorithme de génération des plans candidats

Pour chaque site S du système, considérer que S sera le coordinateur et construire un nouveau plan candidat P déterminé comme suit. Pour chaque fragment F de la requête, tel que F n'existe pas sur S, déterminer le site le plus rapide pour transférer (i.e., répliquer) F vers S. Pour cela, recenser d'abord les sites stockant au moins un fragment. Pour chaque paire <fragment ,site> recensée, estimer le coût de transfert (attente + envoi) vers S afin d'ordonner les paires par coût croissant. Finalement, sélectionner en priorité les paires de moindre coût. Tenir compte du fait que les transferts de plusieurs fragments depuis un même site s'effectuent en séquence et ajuster le coût en conséquence. Nous précisons que l'exploration de toutes les combinaisons possibles de paires <fragments, site> n'est pas faisable dans notre contexte à cause de l'explosion du nombre de combinaisons. Puis, estimer le coût total de P, et retenir le candidat de moindre coût parmi les P générés. Afin de répartir équitablement la charge de traitement des requêtes entre tous les sites, bien que certains fragments soient davantage sollicités que d'autres par les requêtes. Nous contrôlons quels sites peuvent se porter candidats pour coordonner une requête. Pour cela, nous imbriquons la procédure de génération du plan d'exécution dans l'alternative suivante : s'il existe au moins un site beaucoup moins chargé que les autres alors choisir le coordinateur parmi les sites beaucoup moins chargés que les autres ; sinon choisir le coordinateur parmi tous les sites.

5. Validation expérimentale

Les principaux objectifs de la validation sont : (i) de montrer la faisabilité de notre approche pour la réplication à la demande et le modèle de coût dynamique en implémentant les fonctions

de transfert, de traitement et d'optimisation de requête, (ii) de montrer que notre approche offre de meilleures performances que les deux approches suivantes. La première approche, que nous appelons MiniTrans, minimise les transferts en désignant comme coordinateur le site disposant de la plus grande quantité de donnée. La deuxième approche, que nous appelons SelfCoord, désigne comme coordinateur le site ayant reçu la requête. Notre stratégie devrait offrir de meilleurs temps de réponse que MiniTrans et SelfCoord, puisque le temps de réponse d'une requête dépend des paramètres de performance et de charge des sites impliqués et de la localisation. Or aucun des deux autres approches n'intègres ces paramètres ensemble. En effet, SelfCoord ignore la capacité de calcul et la charge du coordinateur. MiniTrans, ne permet pas répliquer suffisamment les fragments très fréquemment accédés.

5.1. Méthodes et outils

Nos expériences consistent à mesurer les temps de réponse pour notre stratégie de traitement de requêtes. Ces temps sont ensuite comparés avec les temps de réponse obtenus avec les stratégies MiniTrans et SelfCoord. Nous avons implémenté avec l'outil de simulation Simjava [9] le comportement des utilisateurs, des sites locaux, du portail GBIF ainsi que toutes les interactions entre les participants. Notre système est composé de 13 machines : une machine représente le GBIF et les 12 autres correspondent aux sites locaux. Les nombres (k) et capacités (c) des machines par rapport au site de référence sont : (k, c)= (2 , 0.25), (2 , 0.5), (4 , 1), (2 , 2) , (2 , 4). Les capacités des liens entre les sites locaux varient entre 5 et 15 unités de temps pour le transfert d'un fragment. Sur chaque site local, nous avons instancié un User Manager, un Local Manager. Nous avons implémenté l'algorithme qui calcule l'ensemble des plans candidats pour le traitement d'une requête et l'algorithme basé sur la fonction d'estimation du coût d'un plan, qui détermine le plan optimal pour son traitement. Nous avons aussi implémenté la coordination entre les participants pour le traitement réparti d'une requête. Chaque site local dispose d'un SGBD pour le traitement effectif d'une requête ou d'une sous-requête.
Nous avons généré un jeu de données de 340 fragments qui est placé initialement sur le portail du GBIF. Autrement dit, les sites locaux ne contiennent initialement aucun fragment. Puis les fragments sont répliqués dynamiquement à la demande vers les sites locaux, en fonction des requêtes posées par les utilisateurs. Un utilisateur pose une séquence de requêtes tant que dure l'expérience. Il attend d'obtenir le résultat d'une requête avant de poser la suivante. Une requête accède six fragments en moyenne. Le nombre d'utilisateur pour une expérience est 36. Nous exécutons une expérience pour chaque approche (Minitrans, SelfCoord et notre approche).

5.2. Résultats obtenus

Nous avons récapitulé dans le tableau 3 ci-dessous les résultats de nos expériences. Ces résultats indiquent aussi le temps de réponse moyen et le temps de réponse plafond pour 90 % des requêtes de chaque expérience. Le premier constat est que la stratégie MiniTrans présente les plus mauvais temps de réponse (temps de réponse moyen = 1337 unités) qui sont largement élevés par rapport à notre approche et l'approche SelfCoord. Ceci s'explique par le fait que les données sont répliquées à la demande vers un petit nombre de sites. Ces derniers accumulent de plus en plus de données, sans partage avec les autres. De ce fait, comme le traitement de la requête est envoyé vers le site qui contient la plus grande portion de données impliquées, alors presque tous les traitements sont transférés vers ces sites. Ce qui augmente davantage leurs charges ainsi que le temps d'attente d'une

requête au moment où d'autres sites demeurent libres. Ces résultats montrent également que notre approche offre de meilleurs temps de réponse que l'approche SelfCoord à la fois pour les temps de réponse moyens (134 unités contre 213 unités) que pour les temps de réponse plafond pour 90 % des requêtes (307 unités contre 432 unités). L'efficacité de notre approche relève du mécanisme d'optimisation basée à la fois sur les charges et les capacités des différents nœuds et les coûts de transferts. En fait, SelfCoord ne considère ni la charge et/ou les capacités des sites, même si elle converge comme notre approche vers une situation où les données nécessaires au traitement d'une requête sont toutes disponibles dans le site coordinateur (pas de transfert). En conclusion, le

Approche	Temps réponse moyen	Temps de réponse pour un plafond de 90 % des requêtes de l'expérience
SelfCoord	213	432
Notre Approche	**134**	**307**
Minitrans	1337	2580

Figure 3. *Récapitulatif des temps de réponse*

plan optimal pour l'exécution efficace d'une requête dépend à la fois des charges et capacités des différents qui disposent de données impliquées et le choix du coordinateur où les données doivent être transférées. En effet le choix du coordinateur est capital pour à la réduire les temps d'attente et de transfert et bénéficier de capacités qui accélère le calcul.

6. Conclusions et perspectives

Cet article aborde, d'une part, la répartition et la réplication à la demande de données de biodiversité d'un entrepôt vers un ensemble de machines et d'autre part, le mécanisme de traitement et d'optimisation de requête à travers ce système. Nous avons proposé une architecture modulaire et décentralisée pour augmenter les fonctionnalités et la réactivité du système. Par la suite, nous avons présenté une stratégie de répartition de données du portail vers les sites locaux. Pour cela, nous avons fragmenté les données en exploitant la structure hiérarchique des données de biodiversité et la nature des requêtes de biodiversité qui s'intéressent principalement aux dimensions taxinomique et géographique. La réplication de ces fragments vers les sites locaux se fait à la demande en fonction des prédicats des requêtes. Nous avons également présenté un mécanisme d'exécution de requête qui s'adapte au contexte de répartition des données à la demande et avons proposé une approche d'optimisation des traitements de requêtes basée sur le coût. Enfin, nous avons implémenté et simulé notre approche d'optimisation de requête et avons mesuré les performances et les résultats sont prometteurs.

Les perspectives sont d'étudier les possibilités d'intégrer des données importantes pour les analyses d'autres sources telles les données environnementales, géographiques, satellitaires, imageries, etc. Cela permettrait d'augmenter les possibilités d'exploitation des données du GBIF.

Références

[1] D. J. Abadi. Data Management in the Cloud : Limitations and Opportunities. *IEEE Data Eng. Bull.*, 32(1), 2009.

[2] S. Alsubaiee et al. ASTERIX : An Open Source System for Big Data Management and Analysis. *PVLDB*, 5(12) :1898–1901, 2012.

[3] K. Bajda-Pawlikowski et al. Efficient processing of data warehousing queries in a split execution environment. In *ACM SIGMOD*, pages 1165–1176, 2011.

[4] N. Bame, H. Naacke, I. Sarr, and S Ndiaye. Architecture répartie à large échelle pour le traitement parallèle de requête de biodiversité. In *African Conf. on Research in Computer Science and Applied Mathematics (CARI)*, pages 143–150, 2012.

[5] N. Bame, H. Naacke, I. Sarr, and S Ndiaye. Algorithmes de traitement de requêtes de biodiversité dans un environnement distribué. *ARIMA*, page to appear, 2014.

[6] M. Brantner et al. Building a database on S3. In *ACM Intl Conf. on Management of Data (SIGMOD)*, pages 251–264, 2008.

[7] B. Chattopadhyay et al. Tenzing A SQL Implementation On The MapReduce Framework. In *PVLDB*, pages 1318–1327, 2011.

[8] GBIF. Gbif data use case.

[9] Fred Howell and Ross Mcnab. simjava : A discrete event simulation library for java. In *Intl Conf. on Web-Based Modeling and Simulation*, pages 51–56, 1998.

[10] GBIF Secretary. Gbif data portal, gbif web site, 2014.

[11] Reynold S. Xin et al. Shark : SQL and rich analytics at scale. In *ACM Intl Conf. on Management of Data (SIGMOD)*, pages 13–24, 2013.

[12] Jing Zhao, Xiangmei Hu, and Xiaofeng Meng. ESQP : an efficient SQL query processing for cloud data management. In *CIKM Intl Workshop on Cloud Data Management (CloudDB)*, pages 1–8, 2010.

A. Estimation du coût d'une requête

Nous estimons le temps de réponse d'un plan en fonction des temps d'attente aux sites impliqués, de transfert et de traitement. En d'autres termes, on tient compte à la fois des charges (nombre de traitement en attente) et des capacités (CPU, mémoire) de chaque site et des capacités des liens entre les différents sites. Pour un traitement local, le temps d'une requête dépend de son type (opérations et quantité de données), des performances du site et de la charge. Pour ce qui concerne le temps de transfert, nous considérons le temps d'attente qu'un site peut faire avant de traiter sa sous-requête et transférer son résultat et le temps de transfert de ce résultat. Le premier dépend du nombre et des types des requêtes en attente au site qui doit transférer ses données et des capacités de celui-ci. Le second dépend des capacités du lien de communication et la quantité de données à transférer. Dans notre estimation du temps total des transferts, nous tenons compte des attentes parallèles au niveau des sites (distants et local) et considérons que la réception des données est parallèle et que l'insertion des données reçues correspond à des traitements de requête de *I* (pour Insert). On suppose que tous les fragments transférés ont la même taille. Le tableau ci-dessous montre les notations utilisées dans notre estimation et leurs descriptions.

Symbole	Description
N_i^k	nombre de requêtes de type k en attente au site S_i
S_l	Site de référence pour estimer les capacités de calcul des autres sites
$T_{ml}^{\ k}$	temps moyen de traitement local d'une requête de type k au site S_l
P_i	Rapport des capacités du site de référence sur celles du site S_i: $P_i = T_{ml}^{\ k} / T_{mi}^{\ k}$
T_{ai}	temps d'attente au site S_i : $T_{ai} = \sum_k (N_i^k * T_{ml}^{\ k}) / P_i$
$T_{mi}^{\ k}$	temps moyen de traitement local d'une requête de type k au site S_i
$T_{ti}^{\ k}$	Temps de traitement local d'une requête de type k au site i, $T_{ti}^k = Tm_l^{\ k} / P_i$
$T_{trj}^{\ \ i}$	temps de transfert d'un fragment du site S_j vers le site S_i
T_{sj}	Temps de sélection d'un fragment au site S_j . $T_{sj} = T_{mj}^{\ s} = Tm_l^{\ s} / P_j$
NF	Nombre de fragments distants nécessaire au traitement de la requête

Figure 4. *Description des notations utilisées*

En supposant que S_i est le site coordinateur du traitement de la requête et S_j un site devant transférer des fragments vers S_i, le temps de réponse d'une requête est :

$T_{ri} = MAX(T_{ai}, MAX(T_{aj} + T_{sj} + T_{trj}^i)) + NF * T_{mi}^I + T_{ti}^k$

Ce modèle de coût montre que le temps de réponse d'une requête distribué, dépend à la fois, des charges de tous les sites impliqués (T_{ai}, T_{aj}) , des différents débits entre les sites impliqués et le site coordinateur (T_{trj}^i) et des performances de chaque site (P_j, P_i). Pour le cas particulier d'un traitement local (absence de transferts), elle montre que le coût dépend de la charge locale (T_{ai}) et de la capacité (P_i) du site. En particulier, ce coût est respectivement proportionnel et inversement proportionnel à la charge et aux capacités du site. Il tient également compte du cas où un accès GBIF est nécessaire. Dans ce cas, le portail GBIF correspond à un site S_j de la formule. La prise en compte de tous les paramètres pouvant influencer le coût de la requête permet de déterminer le plan optimal. L'objectif étant de minimiser le temps de réponse de la requête, le plan optimal correspond au plan candidat pour lequel la valeur de T_{ri} est minimale. Cette estimation présente l'avantage de tenir compte de tous les paramètres qui doivent être considérés dans le traitement d'une requête

dans un environnement distribué hétérogène où les capacités des nœuds, des liens et charges ne sont pas identiques.

B. Courbes d'évolution des temps de réponse

B.1. Approche SelfCoord

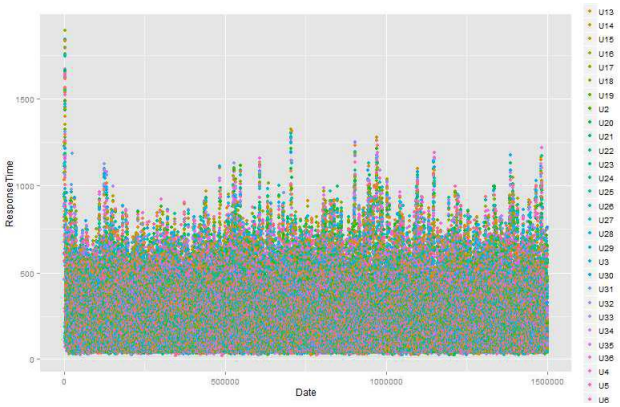

Figure 5. *Temps de réponse de l'approche SelfCoord*

B.2. Notre Approche

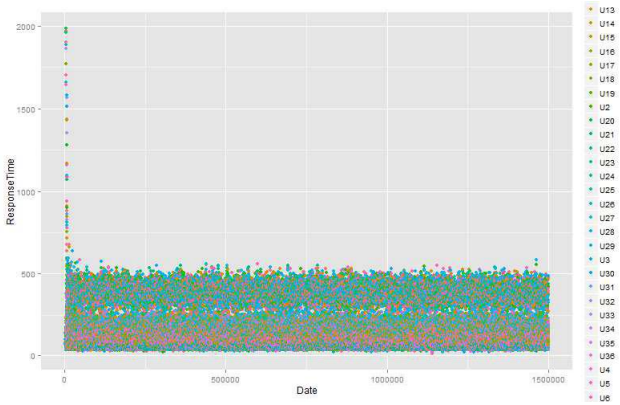

Figure 6. *Temps de réponse de notre approche*

B.3. Approche MiniTrans

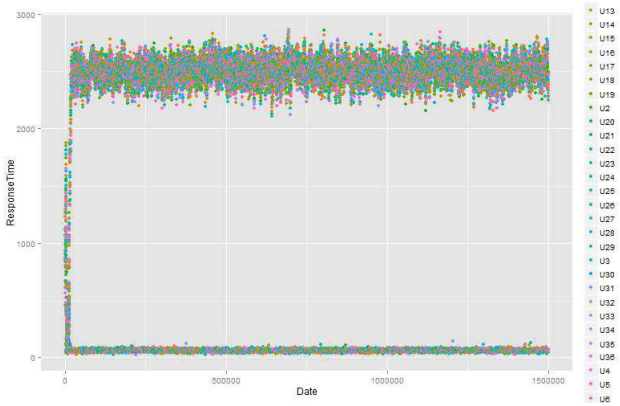

Figure 7. *Temps de réponse de l'approche MiniTrans*