Rapport d'évaluation à mi-parcours de thèse

****

Titre : Gestion de données complexes pour la modélisation de niche écologique

Candidat : Ndiouma BAME

Email : ndiouma.bame@lip6.fr

Directeur de thèse : Bernd AMANN

Encadreur : Hubert NAACKE, Samba NDIAYE, Idrissa SARR

Date de début de la thèse : 01/12/2011

Cours de la formation doctorale suivis : Formation PDI-MSC, 2011, 2012

Publications :

- Ndiouma BAME, Hubert Naacke, Idrissa Sarr, Samba Ndiaye. *« Traitement décentralisé de requêtes de biodiversité »*, CNRIA 2013, Ziguinchor, Sénégal, pages 126-136
- Ndiouma BAME, Hubert Naacke, Idrissa Sarr, Samba Ndiaye. *« Architecture répartie à large échelle pour le traitement parallèle de requêtes de biodiversité ». 11th African Conference on Research in Computer Science and Applied Mathematics (CARI'12)*, Algiers, Algeria, pages 143-150, 2012.
- Ndiouma BAME, Hubert Naacke, Idrissa Sarr, Samba Ndiaye. *« Architecture répartie à large échelle pour le traitement parallèle de requêtes de biodiversité »*, ARIMA 2013 (en cours d'évaluation)

Tables des Matières

1. **Contexte**
   1.1. Présentation du système d'information GBIF
   1.2. Portail du GBIF : architecture et fonctionnement
2. **Cas d'utilisation des données de biodiversité : modélisation de niche écologique**
   2.1. Niche écologique d'une espèce
   2.2. Modélisation de niche écologique
   2.3. Données pour modéliser la niche écologique d'une espèce
   2.4. Utilisation des données du GBIF pour la modélisation de niche écologique
3. **Problèmes liés au fonctionnement du portail GBIF**
   3.1. Besoin d'autres types de requête
   3.2. Accès centralisé et croissance des données et des traitements : passage à l'échelle difficile
4. **Proposition d'une architecture décentralisée non-intrusive**
   4.1. Objectifs
   4.2. Contraintes
   4.3. Méthode de résolution
   4.4. Architecture et composants
   4.5. Fragmentation selon la structure hiérarchique des données de biodiversité
   4.6. Réplication des fragments à la demande
   4.7. Localisation des fragments
   4.8. Traitement parallèles des requêtes
5. **Etat de l'art et contributions**
   5.1. Exploitation des données de biodiversité
   5.2. Répartition de données à la demande
   5.3. Localisation de données dans un environnement distribué
   5.4. Optimisation du traitement distribué de requête
6. Travaux à faire dans le futur
7. Conclusions
8. Références

1. Contexte

1.2. Présentation du système d'information GBIF

Un des plus grands défis du 21ème siècle est de réussir à établir un équilibre entre les besoins des différentes populations et les actions nécessaires à une gestion durable des ressources vivantes. Les chercheurs, étudiants, décisionnaires, ont besoin d'accéder aux stocks mondiaux de biodiversité. C'est ainsi que le GBIF a été créé en 2001 pour le partage des données mondiales de biodiversité, du gène à l'écosystème. Le GBIF est un consortium international visant à fédérer les données de biodiversité à l'échelle mondiale. Il est reconnu comme étant la référence, pour les données primaires de biodiversité, sur laquelle s'appuient les autres initiatives (LifeWatch, GEOBON).

L'objectif du GBIF est de permettre aux responsables politiques, aux décisionnaires, aux chercheurs et au grand public, partout dans le monde d'accéder de façon électronique et gratuite aux stocks mondiaux de données primaires sur la biodiversité. Pour atteindre ses objectifs, le GBIF travaille en étroite collaboration avec les programmes établis et les organisations qui compilent, maintiennent et utilisent les ressources d'informations biologiques.

1.2. Portail du GBIF : architecture et fonctionnement

Le portail du GBIF fournit un accès électronique aux collections mondiales de données primaires sur la biodiversité. Le portail rassemble la description de la plupart des collections de données de biodiversité existant à travers le monde. Pour mettre en œuvre l'intégration des collections (recensement, numérisation, standardisation, ...), le GBIF a tissé un réseau de correspondants nationaux dans de nombreux pays. Le rôle d'un correspondant national est de gérer l'informatisation locale des collections d'un pays.

1.2.1. Architecture du portail GBIF

L'architecture du portail du GBIF représenté sur la figure 1, est fondée sur les correspondants nationaux appelés *nœud nationaux*. Un nœud national dispose de ressources informatiques dédiées à l'intégration des données nationales au sein du portail. Le portail réalise deux services distincts : premièrement, l'intégration des données et, deuxièmement, l'interrogation des données intégrées. (i) L'architecture pour l'intégration des données dans la base centrale du portail est composée des nœuds nationaux (data node ou participant node) et du nœud central du portail (portail du GBIF). Un protocole spécifie l'envoi des données vers le nœud central. Dans ce module, le data node correspond à un fournisseur de données primaires de biodiversité alors qu'un participant node est un fournisseur agréé (participant du consortium) qui peut lui-même approuver des data nodes. (ii) L'architecture pour l'interrogation des données est actuellement centralisée ; toutes les requêtes sont traitées par le nœud central.

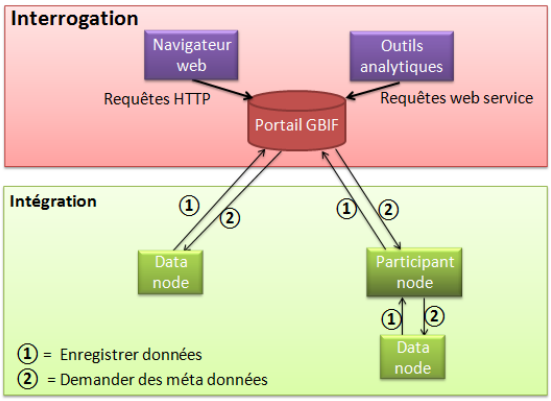

Figure 1 : Architecture du GBIF

1.2.2. Schéma des données

Nous décrivons brièvement le schéma des données du portail du GBIF. La base contient un ensemble de collections. Une collection contient un ensemble d'*occurrences*. Une occurrence décrit l'observation d'un spécimen. Une occurrence contient de nombreux champs taxinomiques ainsi qu'un géo-référencement. Une occurrence constitue l'unité d'information élémentaire de la base de données. Par ailleurs, la provenance des collections est décrite : une collection appartient à un fournisseur, lui-même rattaché à un pays. En termes de taille, la base de données contient plus de 400 millions d'occurrences issues de plus de 10000 collections et 560

fournisseurs. La taille totale de la base est de plusieurs centaines de giga octets. Pour ce qui concerne la structure, base de données dispose d'une table principale (très volumineuse) qui contient toutes les informations sur les occurrences de spécimens partagées à travers le réseau et des tables annexes (de petites tailles).

1.2.3. Requêtes supportées par le portail

Le portail GBIF permet de rechercher des occurrences d'espèces. Le portail propose deux types de requêtes : (i) une requête par navigation selon un pays, une collection ou une structure taxinomique ; (ii) une requête conjonctive multi critères portant sur un nombre fini de champs (nom scientifique, date, localisation géographique, etc..). Les requêtes supportées actuellement par le portail, sont des sélections simples sur les occurrences. La forme générale des ces requêtes est la suivante :

SELECT attribut1, ……, attribut k
FROM table_occurrence
WHERE prédicat1 AND prédicat2, AND …….AND prédicat n

Les prédicats (*prédicat1, …, prédicat n*) sont de la forme *attribut i = valeur* (sauf pour les prédicats sur latitude, longitude et altitude qui supporte des comparaisons d'inégalité) où *attribut i* n'apparait pas dans les autres prédicats. Le résultat d'une requête a une structure fixe : c'est un ensemble d'occurrences avec tous leurs champs. Le portail ne supporte aucun autre type de requête, à l'heure actuelle. La nature simple de ces requêtes limite les possibilités d'interrogation des données disponibles. En effet, pour les cas d'utilisation notés au [3], les analyses portent sur un ensemble fini d'espèces dans des dimensions spatiale et temporelle définies. Ce qui oblige les usagers à effectuer des accès multiples au portail pour télécharger les occurrences de chacune des espèces nécessaires à l'analyse et les garder en local. Ce qui serait coûteux en termes d'accès, de transferts et de stockage surtout lorsque le volume de données à manipuler est important. Une fois les données en local, l'utilisateur effectuent des calculs de regroupements (densités, abondance, distribution, découpage temporel, découpage spatial, …) sur les occurrences d'espèces. Par ailleurs, le portail propose deux types d'interfaces pour l'interrogation : une interface de type IHM (interface homme machine à travers un navigateur) et une interface programmatique permettant d'automatiser l'interrogation (sous la forme de service web). Le service web peut être ainsi utilisé directement par de nombreuses applications tierces qui étudient divers aspects de la biodiversité. La facilité d'accès automatisé accroit fortement le nombre de requêtes que doit traiter le portail, ce qui accentue le besoin pour une solution de traitement de requêtes plus efficace.

2. Cas d'utilisation des données de biodiversité : modélisation de niche écologique

2.1. Niche écologique d'une espèce

La niche écologique d'une espèce est l'ensemble des paramètres et conditions qui permettent à l'espèce d'accomplir son cycle biologique et sa reproduction à l'intérieur d'une plage limitée de variations environnementales d'origine abiotique (facteurs physiques et chimiques) et biotique (interaction entre espèces) [29]. Hutchinson a précisé ce concept en définissant la niche écologique d'une espèce comme un hyper-volume multidimensionnel dans lequel chaque dimension représente un paramètre biotique ou abiotique conditionnant la présence de l'espèce. La niche fondamentale est l'ensemble des ressources potentielles qu'une espèce peut utiliser dans son milieu lorsque les conditions sont idéales (absence de compétition ou prédation). La niche réalisée (ou niche réelle) est l'ensemble des ressources réellement utilisées par une espèce en conditions naturelles (compétition pour les ressources, contrôle pour un consommateur (prédateur ou herbivore), contrôle par un parasite, absence d'une autre espèce (un pollinisateur)).

2.2. Modélisation de niche écologique

La modélisation de la niche écologique d'une espèce est assez complexe et est limité par la méconnaissance de certaines interactions entre l'espèce à modéliser et d'autres espèces dont on ignore et aussi par le manque d'information sur le comportement biologique de l'espèce. La modélisation statistique est utilisée pour modéliser la niche écologique d'une espèce. La modélisation statistique de la niche écologique d'une espèce consiste à construire une fonction de paramètres environnementaux qui prédit la probabilité de présence de l'espèce à partir d'un jeu de données de calibration comprenant des données de présence/absence ou d'abondance de l'espèce et des valeurs de paramètres environnementaux aux sites d'observation [29]. L'objectif de la modélisation de la niche écologique d'une espèce est d'étudier (comprendre, trouver, prédire) les comportements (distribution, migration, menace d'extinction, possibilité d'établissement, …) de l'espèce vis-à-vis des facteurs environnementaux selon les dimensions spatio-temporelles de l'étude. On trouve des applications de la modélisation de niche écologique en agronomie (impact des abeilles sur une fleur, orientation vers une nouvelle espèce d'arachide, …), en écologie (dégager les zones de protection pour une espèce menacée, migration des hirondelles en hiver,…), en élevage (rendement d'une espèce herbivores en fonction des ressources et cohabitation entre plusieurs espèces), ….

2.3. Données pour modéliser la niche écologique d'une espèce

Pour modéliser la niche écologique d'une espèce, il es nécessaire de disposer d'un jeu de données de calibration associant des données d'occurrence de l'espèce à des valeurs de paramètres environnementaux sur un certain nombre de sites d'observation. Les données d'occurrence de l'espèce peuvent être des données de présence/absence ou des données d'abondance. Les paramètres environnementaux peuvent être des données physico-chimiques (températures, salinité, rayonnement solaire) qui caractérisent la zone d'étude et des données d'interaction avec d'autres espèces (co-occurrence entre espèce).

2.4. Utilisation des données du GBIF pour la modélisation de niche écologique

Nous venons de voir dans la sous-section précédente que les données nécessaires à la modélisation de la niche écologique d'une espèce sont les données de présence/absence ou d'abondance et les données environnementales. Le GBIF fournit des données de présence/absence et permettrait de calculer l'abondance de l'espèce dans les zones d'étude grâce au géo-référencement des données. Pour ce qui concerne les paramètres environnementaux, le GBIF pourrait fournir des données biotiques (interaction avec d'autres espèces) grâce à des requêtes de co-occurrences et les données physico-chimiques sont obtenues auprès d'autres sources (bioclim, worldclim, …). Le papier *« Finding pattern in bee-plant relationship »* illustre un cas d'utilisation des données du GBIF complétées avec d'autres données pour l'étude des relations entre les abeilles et les plantes. Dans cette étude, l'utilisateur calcule dans les densités de co-occurrence entre abeilles et plantes en fonction d'un maillage. La cohabitation est considérée pour les zones où les densités sont supérieures à une valeur minimale. Pour comprendre ou modéliser une espèce, il est nécessaire de disposer de connaissances basiques sur la densité ou l'abondance de l'espèce dans chacune des différentes zones de l'étude. Ces informations d'entrées aux analyses nécessitent des traitements d'agrégation sur les espèces et les dimensions spatiale et/ou temporelle, que le GBIF n'effectue pas en amont actuellement.

3. Problèmes liés au fonctionnement du portail GBIF

Les problèmes identifiés dans le fonctionnement actuel du GBIF peuvent être résumés en deux points : (i) une limitation de l'expressivité des requêtes qui réduit les possibilités d'exploitation des données; (ii) une architecture centralisée qui peut être source de congestion pour les accès et traitements multiples et simultanés, ne garantie pas le passage à l'échelle avec le support de nouvelles fonctionnalités par le GBIF. Nous détaillons dans les deux sous-sections suivantes ces deux problèmes avant de proposer dans la section suivante une solution à ceux-ci.

3.1. Besoin d'autres types de requête

En plus des requêtes prédéfinies, les usagers ont exprimés leurs besoins pour d'autres types de requêtes plus expressives, mais qui ne sont pas supportées par le portail actuel du GBIF. Généralement, les usagers ont besoin de combiner les résultats de plusieurs requêtes simples (supportées par le portail) pour effectuer des opérations ensemblistes non supportées par le portail telles que l'union, l'intersection, la différence entre plusieurs ensembles d'occurrences. Les usagers ont aussi besoin d'agréger et de dénombrer des occurrences selon des critères spatiaux et temporels (e.g. superposer les distributions géographiques de deux espèces, calcul de la densité d'une espèce dans une zone). Nous avons vu que dans les cas d'utilisation des données de biodiversité, il est nécessaire de faire des traitements d'agrégation pour calculer des densités (ou abondance) ou des co-occurrences. Pour la modélisation de la niche écologique d'une espèce, le système actuel du GBIF ne supporte pas les calculs de l'abondance de l'espèce et des densités de co-occurrence avec une autre espèce. L'utilisateur est obligé à chaque fois d'accéder au portail GBIF pour lire les occurrences de chaque espèce nécessaire à ses analyses, de faire ces traitements d'agrégation avant de démarrer ses analyses. Le téléchargement des données d'occurrence et leur stockage en local peut être coûteux surtout lorsque la quantité de données à utiliser est très importante. Outre les coûts de transfert et de stockage, ces traitements supplémentaires qui précédent l'analyse ne facilite pas l'utilisation des données du GBIF, alors qu'ils pourraient être effectués en amont au GBIF en fonction de paramètres liés aux dimensions et aux échelles d'agrégation. L'exemple du cas d'utilisation de données du GBIF sur l'étude des relations entre les abeilles et les plantes [] n'est pas supporté directement par le GBIF. L'utilisateur doit combiner les résultats de plusieurs requêtes simples (pour lire les abeilles et lire les plantes) et effectuer les calculs de co-occurrence en local, avant de modéliser les interactions entre abeilles et plantes. En plus de ces traitements d'agrégation, des utilisateurs ont besoins de connaitre les espèces endémiques à une zone, alors que ce type de requête n'est pas aussi supporté par le GBIF.
Le manque d'expressivité des requêtes supportées par le GBIF ne favorisent pas les exploitations qui devraient être faites des données partagées. De telles exploitations nécessitent des traitements impliquant des opérations de jointure, de regroupement suivi d'agrégation plus complexes ou des requêtes imbriquées qui ne sont pas supportées par le portail.

3.2. Accès centralisé et croissance des données et des traitements : passage à l'échelle difficile

Le portail du GBIF fournit un accès centralisé à toutes les données intégrées dans la base de données du GBIF. Plusieurs problèmes se posent pour faire face à la demande croissante des usagers. En outre l'ajout de nouvelles fonctionnalités génère de nouveaux traitements plus intensifs, alors que l'accès centralisé ne garantie pas le passage à l'échelle.

- **La croissance des données** : Le nombre d'enregistrements accessibles dans le portail est passé de 163 millions en décembre 2008 à 416 millions en septembre 2013 [3]. La croissance du nombre d'enregistrement s'accentue à l'heure actuelle pour atteindre prochainement le milliard d'enregistrements. Cette quantité énorme de ressources sur les données primaires de la biodiversité mondiale incite les usagers à affluer davantage vers le portail. Ce qui augmente considérablement les accès simultanés et les traitements.

-La croissance des requêtes : Le nombre de requêtes par jour s'accentue à cause du nombre croissant et élevé d'usagers (plus de 775 000 visites en 2012 [2]) et de l'accès automatisé depuis des applications tierces. De plus, dès lors que le portail complètera ses fonctionnalités de requêtes pour mieux satisfaire les besoins des usagers, le portail devra traiter des requêtes de plus en plus complexes et nombreuses.

-Disponibilité du portail : La disponibilité du portail est sa capacité à répondre à toutes les requêtes en un temps raisonnable. Les accès centralisés, multiples et simultanés au portail posent des problèmes de congestion. Ce qui ralentit la réactivé du système vis-à-vis des usagers et augmente le temps d'attente pour visualiser leurs requêtes. Par exemple la durée moyenne d'une visite en 2012 est 3.43 minute [2]. Avec la croissance des usagers on peut s'attendre à des millions de visites dans peu d'années. Ce qui serait proportionnel à la latence si des politiques d'optimisation ne sont pas menées.

La centralisation de l'accès au GBIF, de sa base de données et des traitements sur un seul serveur pose pas mal de problèmes sur sa réactivité vis-à-vis de ses usagers et rend difficile le passage à l'échelle vu la croissance de ses usagers, des données et des traitements et surtout lorsque de nouvelles fonctionnalités seraient ajoutées.

4. Proposition d'une architecture décentralisée non-intrusive

Nous proposons une solution pour répartir le traitement des requêtes et les données interrogées sur plusieurs machines afin d'allouer plus de ressources de stockage et de calcul. L'objectif étant de satisfaire toutes les demandes croissantes des utilisateurs tout en évitant la concentration de celles-ci, qui est une source de congestion (passage à l'échelle difficile). La méthode de résolution doit tenir compte des contraintes imposées par le fonctionnement actuel du portail GBIF.

4.1. Objectifs :

Les objectifs fixés dans cette thèse sont de :

(i) Compléter les fonctionnalités de traitement de requêtes pour pouvoir traiter les requêtes complexes des utilisateurs afin de favoriser la croissance des exploitations faites des données de biodiversité (de la sélection simple d'occurrences aux analyses de biodiversité).

(ii) Optimiser le traitement des requêtes pour augmenter la réactivité du système vis-à-vis des usagers (réduire la latence). En effet, le support de nouvelles fonctionnalités (objectifs (i)) augmentera le nombre d'usagers qui vont générer des quantités énormes de traitements. La solution doit donc assurer le passage à l'échelle avec un mécanisme de traitement de requêtes adéquat.

4.2. Contraintes

(i) Se coupler avec le portail existant, sans nécessiter de modifier le portail actuel. Cela est impératif pour que la solution soit utilisable en réalité. Cela impose de compléter le service de requêtes existant avec d'autres services de requêtes complexes venant relayer (de manière transparente pour les utilisateurs) le portail centralisé.

(ii) Séparer entièrement l'étape de traitement des requêtes et l'étape d'intégration de nouvelles données. Les requêtes sont par conséquent évaluées sur un cliché (snapshot) des données. La mise à jour des données issues de l'étape d'intégration est effectuée en dehors des périodes de requêtes. Dans le contexte de l'analyse de données de biodiversité portant sur plusieurs années, les utilisateurs tolèrent l'absence temporaire des données les plus récentes dans le cliché qu'ils interrogent.

4.3. Méthode de résolution

Pour atteindre les objectifs fixés tout en tenant compte des contraintes ci-dessus, nous répartissons les données et les traitements selon les principes suivants :

(i) Disposer de plusieurs machines hébergeant chacune un serveur de données capable de traiter la requête d'un utilisateur. Opter pour paralléliser les requêtes afin de garantir la disponibilité du service de requêtes :

parallélisme inter-requête pour supporter le traitement simultané de plusieurs requêtes dans le délai escompté, et parallélisme intra-requête pour accélérer une requête complexe traitée trop lentement sur un seul serveur de données.

(ii) Fragmenter dynamiquement les données, à la demande, en fonction des requêtes et en exploitant la structure hiérarchique des données de biodiversité.

(iii) Réplication des données, à la demande, en fonction des requêtes.

(iv) Optimiser le coût des requêtes. Exploiter au mieux les ressources disponibles pour réduire le temps de réponse des requêtes avec la collaboration entre les sites locaux pour le partage des données et des traitements. On suppose que plusieurs utilisateurs (collaborateurs d'un même projet, chercheur sur un vecteur transmetteur de l'agent pathogène lors d'une épidémie, solution de médicament contre une maladie, etc.) accèdent à des fragments de données assez communs.

4.4. Architecture et composants

Nous proposons une architecture distribuée qui s'interpose entre les utilisateurs et le portail GBIF. L'intérêt de cette architecture est de décharger le portail des tâches d'interrogation et d'offrir aux usagers la possibilité de traiter leurs demandes sans limitation : bref pour optimiser l'exploitation faite des données de biodiversité. La figure 2 montre l'architecture proposée avec les principaux composants et leurs interactions.

- **Le portail :** c'est un entrepôt récapitulant l'ensemble des données primaires de biodiversité accessibles à travers le réseau GBIF [1]. Cette base est utilisée pour traiter directement les requêtes ou comme passerelle pour accéder aux bases des fournisseurs. Dans le cadre de notre thèse, nous considérons le cas où les requêtes de l'utilisateur sont traitées au portail. On accède aux données de la base du portail par navigateur web et/ou par des appels web service.

- **Les utilisateurs** : Ce sont les usagers du GBIF qui utilisent les données de biodiversité pour effectuer leurs analyses. Chaque utilisateur, pour accéder aux données, se connecte au site le plus proche. Via l'interface de le gestionnaire des requêtes d'utilisateurs (User Manager), il peut consulter les données d'occurrences, effectuer des calculs d'abondance et même faire ses analyses en chargeant les données complémentaires et choisissant les paramètres et/ou algorithmes désirés.
- **Le gestionnaire des accès au GBIF (GBIF Access Manager)** : coordonne les accès au GBIF. Pour éviter qu'un site local accède au GBIF pour lire un fragment qui dispose déjà d'une réplique dans les autres sites ou que deux sites accèdent simultanément au portail GBIF pour lire le même fragment, le GBIF Access Manager est chargé de recevoir toutes les demandes d'accès au GBIF des sites locaux, de formuler les appels web service correspondants et de soumettre la requête au GBIF. A la réception des résultats, il les transmet au demandeur et insère une entrée concernant la disponibilité du fragment dans le système, dans son catalogue (Global index). De cette façon, le GBIF Access Manager dispose d'un catalogue global (Global index) qui indexe tous les fragments disponibles dans les sites locaux. Le GBIF Access Manager est également informé des modifications apportées aux fragments locaux afin qu'il mette à jour son catalogue. Il est aussi charger de la gestion de mises à jour des fragments locaux par rapport à ceux du portail GBIF. Pour cela, il remplace de façon périodique chaque fragment disponible dans les sites locaux par une nouvelle version lue du portail GBIF.

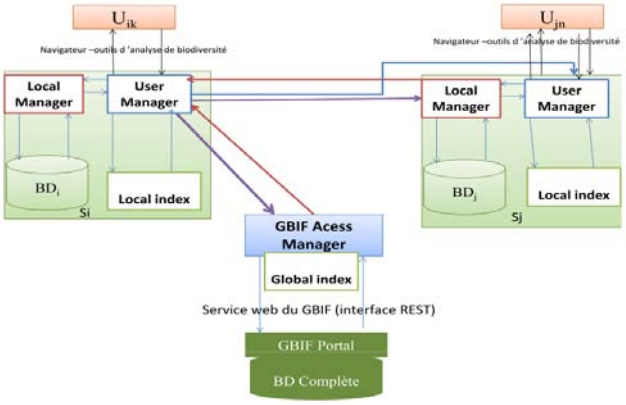

Figure 2: Architecture distribuée pour traiter les requêtes de biodiversité

- **Site locaux ($S_{i, sj}$)**: Les sites locaux disposent chacun d'une base de données locale qui consiste à héberger les fragments de données sollicités par les utilisateurs qui s'y connectent. Un utilisateur accède aux données du GBIF à travers un site local S_i. Les différents sites locaux sont reliés par un réseau et collaborent pour échanger des données et traiter des requêtes. En effet, les fragments traités par un utilisateur sont stockés dans la base de données locale et peuvent être réutilisés par le même utilisateur ou d'autres utilisateurs lors des prochaines demandes soumises au site local ou aux sites voisins. Ce qui permet de réduire davantage les traitements d'interrogation au portail GBIF. Chaque site local, S_i, héberge une base de données locale *BDi*, un gestionnaire des requêtes d'utilisateurs (User Manager), un gestionnaire des requêtes locales (Local Manager) et un catalogue.
  - **La base de données locale BD_i** est vide au démarrage et est alimentée au fur et à mesure que de nouvelles requêtes demandant de nouvelles données sont soumises à partir du site S_i via son interface User Manager et/ou que des données ont été transférées lors de traitements de requêtes. Nous verrons les détails des réplications de fragments dans les prochaines sous sections. La base BDi contient toutes les données nécessaires pour le traitement d'une requête locale.
  - **Catalogue** : Chaque site Si dispose d'un catalogue local qui contient toutes les informations relatives à la localisation de chaque fragment qu'il héberge et aux fragments qu'il traite. Composant essentiel, le catalogue, grâce à son contenu de localisation des fragments, permet au User Manager de déterminer le plan optimal. Il est donc utilisé par le gestionnaire des requêtes d'utilisateurs (User Manager) pour localiser les fragments nécessaires à une requête lors du processus d'analyse et d'optimisation de la requête.
  - **Le gestionnaire des requêtes d'utilisateurs (User Manager) :** fournit une interface conviviale aux usagers où ils pourront soumettre leurs demandes sans limitations. Il dispose d'un composant lui permettant de déterminer toutes les relations d'inclusion possible entre les fragments. Il effectue les travaux d'analyse et d'optimisation des requêtes d'utilisateurs. A la réception d'une requête utilisateurs, il effectue les analyses nécessaires avec le catalogue pour trouver la localisation de chaque fragment nécessaire, fait les traitements d'optimisation avant de rediriger la requête et le plan optimal au User Manager désigné pour coordonner l'exécution.
  - **Le gestionnaire des requêtes locales (Local Manager) :** est chargé de la gestion des traitements de toutes les requêtes au niveau de la base de données locale. Il interagit avec les User Manager local ou distants pour recevoir des requêtes ou retourner des résultats. Il dispose d'une file d'attente qui contient les requêtes en attente lorsque le SGBD local est occupé.

4.5. Fragmentation selon la structure hiérarchique des données de biodiversité

Les données de biodiversité ont la particularité d'avoir une structure hiérarchique selon plusieurs dimensions. En effet les dimensions de la classification taxinomique, géographique, temporelle et fournisseurs présentent chacune une structure hiérarchique. Dans notre travail, nous nous limitons aux dimensions taxinomique et géographique qui sont les plus fréquemment utilisées dans les analyses de données de biodiversité notamment en modélisation de niche écologique.

4.5.1. Structure hiérarchique de la taxinomie

La classification taxinomique des espèces en biodiversité présente une structure hiérarchique dans laquelle un ensemble d'occurrence compose une espèce (ou sous-espèce) qui avec d'autres espèce forment un genre appartenant à une famille. Le processus se poursuit jusqu'au règne. L'ensemble des règnes constitue l'écosystème.

4.5.2. Structure hiérarchique de la géographie

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 (mailles) délimité chacun par une longitude maximale, une longitude minimale, une latitude maximale et une latitude minimale, forme une maille plus large et ainsi de suite.

4.5.3. Définition des fragments

Nous venons de voir que les données de biodiversité présentent une structure hiérarchique selon les dimensions taxinomique et géographique et avons vu dans les cas d'utilisation que les analyses en biodiversité concernent souvent ces deux dimensions (distribution, migration, co-occurrence, modélisation de niche écologique, etc.). De ce fait, on suppose dans la suite qu'une requête utilisateur dispose de prédicats concernant au moins une de ces deux dimensions. Le fragment traité par la requête est donc défini par la conjonction des prédicats sur ces deux dimensions. Les autres prédicats seront ignorés dans le processus de fragmentation et pourront être appliqués dans le fragment en question lors du processus de traitement de la requête. Ainsi un fragment est défini par la

conjonction de nœuds dans la hiérarchie des dimensions taxinomique et géo-spatiale. Cette structure hiérarchique des fragments permet de constater une relation de spécialisation/généralisation entre des fragments. Ce qui fait que bien que deux fragments n'ont pas les mêmes prédicats de définition peuvent avoir une relation d'inclusion où le fragment le plus large inclut le plus petit. De ce fait, le fragment qui inclut l'autre peut être utilisé pour des traitements sur le fragment inclus. Ce qui favorise la réutilisation des fragments pour des requêtes qui ne les invoquent pas directement dans les prédicats. Un fragment est identifié par deux parties conjointes : nœud de la structure taxinomique et nœud de la structure géo-spatiale. Un nœud est défini par la concaténation de l'identifiant de son parent direct (supérieur hiérarchique) et son identifiant dans son groupe (éléments issues directement du même parent).

4.6. 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, faite à la demande, permet de minimiser les coûts de stockage des BD_{is} et de communication avec le GBIF tout en favorisant le parallélisme des accès et des traitements. Ce qui a comme avantage de diminuer le temps de réponse du système. De plus, cette stratégie de réplication permet de placer les données aux endroits où elles sont fortement et fréquemment utilisées. Ainsi les fragments sont placés aux sites où ils sont sollicités. Un fragment qui n'est pas encore disponible dans les sites locaux, sera placé au 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 requêtes qui nécessitent que toutes les données soient en un seul endroit, les fragments nécessaires qui ne se trouvent pas dans le site de traitement seront transférés vers celui-ci et sont donc répliqués pour être utilisés lors des prochaines requêtes les invoquant. Les tailles des bases de données locales étant limitées, lorsque le seuil est atteint, une stratégie de remplacement est appliquée avant de répliqué un nouveau fragment. Cette stratégie vise à supprimer le fragment le moins fréquemment utilisé si celui-ci dispose d'autres répliques dans d'autres sites. Sinon, il est déplacé vers un autre site disposant suffisamment d'espace de stockage où il serait probablement sollicité. Cette probabilité est calculée en fonction des relations sémantiques entre le fragment en question et ceux les plus utilisés dans les sites.

4.7. Localisation des fragments

La localisation des données est un point essentiel lors du processus d'optimisation et de traitement de requête dans un environnement distribué. Dans notre architecture, chaque site dispose d'un catalogue local qui indexe tous les fragments hébergés dans la base de données locale. Nous proposons d'utiliser le catalogue du GBIF Access Manager pour localiser les fragments distants. Le GBIF Access Manager coordonnant tous les accès au portail GBIF, dispose de toutes les informations de localisation de tous les fragments dans les sites locaux. Ce qui lui permet de disposer d'un catalogue global sur les localisations des fragments. Le GBIF Access Manager est informé de toutes les modifications et mouvements apportés aux fragments des sites locaux afin qu'il mette à jour le catalogue global. Dans l'optique de retrouver le plus rapidement possible les localisations des fragments à traiter, sans pour autant surcharger le GBIF Access Manager et de réduire le trafic réseau, chaque site indexe également les fragments sollicités par ses utilisateurs. Le processus d'indexation de fragments distants est géré par le GBIF Access Manager de façon périodique. Il utilise les statistiques de demandes de localisation de fragments soumis par chaque site pour les informer de façon périodique des localisations des fragments et de leurs répliques. Les sites locaux vont ainsi combiner ces informations de localisation des fragments distants sollicités avec leur index local. Lorsque l'index local ne permet pas de localiser un fragment, le User Manager demande la localisation au catalogue global. Le GBIF Access Manager enregistre dans son log cette demande pour informer au User Manager de toutes les localisations et modifications apportées aux fragmentés demandés lors de chaque rafraichissement.

4.8. Traitement parallèles des requêtes

Le traitement de requêtes dans un environnement distribué est assez complexe par rapport aux architectures centralisées. Le coût d'une requête dépend de plusieurs paramètres : localisation des données, le choix de la réplique à traiter, le coût de transfert, les charges et capacités des nœuds. Les environnements distribués notamment à large échelle sont caractérisés par l'hétérogénéité des composants (nœuds et liens) et les traitements simultanés de plusieurs requêtes à travers le système. Dans un tel contexte, la réduction du coût d'une requête résulte d'un mécanisme efficace d'optimisation qui prend en compte tous les paramètres susceptibles d'affecter ce coût. L'objectif de notre approche est de traiter plus de requêtes pendant une période donnée et pour chaque requête, de minimiser son temps de réponse. Dans la suite de cette sous-section, nous montrons la manière dont les sites locaux collaborent lors des traitements de requêtes et présentons notre stratégie d'optimisation qui a pour but de réduire le temps de réponse.

4.8.1. Mécanisme d'exécution de requêtes

A la réception d'une requête utilisateur, le User Manager effectue l'analyse de la requête pour trouver les fragments nécessaires et leurs relations d'inclusion avec d'autres fragments qui seraient suffisant pour traiter un ou plusieurs fragments de la requête. Grâce au catalogue local et au catalogue global du GBIF Access Manager (en cas de nécessité), il détermine les localisations des fragments impliqués. Ensuite, il effectue les optimisations nécessaires pour choisir le plan optimal contenant l'ensemble des opérations et fragments à traiter dans chaque site. Pour trouver ce plan optimal, il génère tous les plans possibles en fonction du nombre de répliques de chaque fragment et de la localisation de chacune d'elles. Pour chaque plan, il évalue le coût total de traitement en fonction des attentes, des transferts et des capacités des sites. Le plan optimal correspond à celui qui minimise ce coût. Lorsque deux ou plusieurs plans ont le même coût, nous choisirons de traiter celui qui minimise le coût de transfert afin de réduire le trafic réseau. Le User Manager envoie un message contenant la requête complète de l'utilisateur et le plan optimal au User Manager du site désigné pour coordonner le traitement. Celui-ci coordonne le traitement de la requête en formulant et envoyant les sous-requêtes correspondantes aux Local Manager de chaque site distant impliqué. En attendant que toutes les données distantes nécessaires à l'exécution de la requête lui soient transmises, le Local Manager du site désigné continue de traiter les requêtes qui lui sont soumises dont les temps de traitement estimés sont inférieurs au temps d'attente et de transfert des requêtes en attente de données distantes. A la réception de tous les résultats partiels, le User Manager les insère dans la base locale et soumet la requête complète au Local Manager. A la fin de l'exécution, il reçoit le résultat final du Local Manager et le retourne directement à l'utilisateur. Ce mécanisme permet d'éviter que le site qui est chargé d'effectuer le traitement ne soit inactif pendant l'attente des transferts qui dépend des sites distants. De la même façon, le User Manager local 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. La figure ci-dessous illustre le processus général de traitement d'une requête utilisateur.

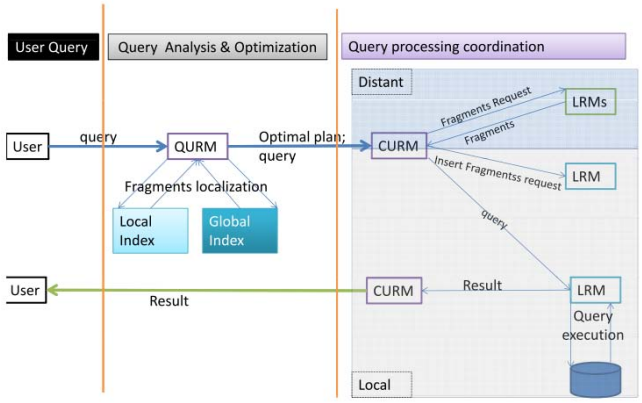

Figure 3 : principes généraux d'exécution de requête

4.8.2. Optimisation basée sur le coût

Notre processus d'optimisation prend en entrées la requête utilisateur (opérations et données), les localisations (et leurs caractéristiques) de chaque fragments (et répliques) et produit le plan optimal pour le traitement de la requête. Ce plan correspond à celui qui offre le meilleur temps de réponse. Pour aboutir à ce plan, il effectue toutes les combinaisons possibles pour la lecture des fragments nécessaires à la requête sachant que pour chaque fragment, une seule réplique est utilisée dans une combinaison. Une combinaison pour la lecture des fragments nécessaires au traitement de la requête et le site coordinateur du traitement final constitue un plan. Le produit cartésien de toutes les combinaisons possibles avec l'ensemble des sites correspond à l'ensemble des plans candidats. Pour chacun d'eux, on estime le coût et le plan optimal correspond à celui qui dispose du coût minimal. Par la suite, nous désignons ce coût par le temps de réponse. Nous évaluons le temps de réponse d'un plan en fonction des temps d'attente, de transferts et de traitement qui dépendent respectivement des charges (nombre de traitement en attente) des capacités des liens (bandes passantes entre les sites) et des capacités (CPU, mémoire) de chaque site. 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. On désigne le temps moyen de traitement local d'une requête de type k au site S_i par T_{mi}^{k}. De la même façon, le temps d'attente dépend du nombre de chaque type de requête en attente et des capacités du site. On note T_{ai} , le temps d'attente au site Si et N_i^k le nombre de requêtes de type k en attente au site Si. 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 des données. 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 de la bande passante du lien de communication et la quantité de données à transférer. Dans notre méthode d'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

séquentielle et que l'insertion des données reçues dans la base locale correspond à des traitements de requête de type *I*. Nous notons $T_{tr\ j}^{\ i}$, le temps de transfert d'un fragment du site S_j vers le site S_i. Le tableau ci-dessous montre les notations utilisées dans notre estimation et leurs descriptions.

Symbole	Définition
T_{mi}^{k}	temps moyen de traitement local d'une requête de type k au site S_i
N_i^k	nombre de requêtes de type k en attente au site S_i
T_{ai}	temps d'attente au site S_i : $T_{ai} = \sum_k (N_i^k * T_{m1}^k)$
S_1	Site de référence pour estimer les capacités de calcul des autres sites
T_{m1}^{k}	temps moyen de traitement local d'une requête de type k au site S_1
P_i	Rapport des capacités du site de référence sur celles du site S_i: $P_i = T_{m1}^{k} / T_{mi}^{k}$
$T_{tr\ j}^{\ i}$	temps de transfert d'un fragment du site S_j vers le site S_i
T_{atri}	Temps d'attente et de transfert des sites qui doivent effectuer des transferts vers le site S_i
L	sites ordonnés de façon croissante suivant les valeurs des temps d'attente
$T_{trans\ p}$	temps nécessaire (toutes les attentes + transfert) du site correspondant à $L(p)$
$T_{trans\ (pmax)}$	temps nécessaire (toutes les attentes + transfert) du site qui a le plus grand temps d'attente
NS_i	Nombre de site devant faire des transferts vers le site S_i

Nous définissons la fonction récurrente suivante pour le calcul de T_{atr}.

(1) : $T_{trans\ 1} = Ta_{L(1)} + T_{trL(1)}^{\ i}$ *(2) :* $T_{trans\ (p+1)} = MAX\ (T_{trans\ p}\ ,\ Ta_{L(p+1)}) + T_{tr\ L(p+1)}^{\ i}$ *,(3) :* $T_{atr} = T_{trans\ (pmax)}$ *où* T_{atr} temps du dernier élément de la liste (qui a le plus grand temps d'attente).

Le temps de réponse, T_{ri}, lorsque la requête complète est traitée au site Si est donc :

$T_{ri} = MAX\ (T_{ai}\ ,\ T_{atri}) + NSi * T_{mi}^{I} + T_{ti}$ *avec* $T_{atri} = T_{trans\ (pmax)}$, $T_{ai} = \sum_k (N_i^k * T_{m1}^k) / P_i$, *et* $T_{ti} = Tm_1^k / P_i$.

Cette fonction montre que le temps de réponse pour une requête distribué, dépend à la fois, des charges de tous les sites impliqués (T_{ai}, T_{atri}) , des différents débits entre les sites (T_{atr}) et des performances de chaque site (T_{atr}, 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 des performances (P_i) du site. On constate que ce coût est respectivement proportionnel et inversement proportionnel à la charge et aux capacités du site.
Afin de mieux exploiter les ressources locales, lorsque $T_{ai} < T_{atri}$, toute nouvelle requête locale reçue à l'instant t (par rapport au calcul de T_{atri}) sera directement soumise au Local Manager si son temps estimé (attente + traitement) T_{rni} est tel que : $T_{rni} < T_{atri} - t$ *si* $T_{ai} < t$ *ou* $T_{rni} < T_{atri} - T_{ai}$ *si* $T_{ai} \geq t$
Ce mécanisme permet d'utiliser les ressources disponibles au plus possibles et augmentent ainsi leur débit transactionnel. 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 à celui pour lequel la valeur de Tr_i est minimale. Cette estimation présente l'avantage de tenir 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 les charges ne sont pas identiques et s'adapte bien à une stratégie de réplication complète avec l'équilibrage de la charge en fonction des charges et capacités des sites.

4.8.3. Méthodes de validation

Nous avons commencé l'implémentation d'un prototype pour la validation de notre approche sur le traitement parallèle de requête. L'objectif de nos premières expériences est de mesurer le débit du système en termes de nombre de requêtes traitées par unité de temps. Les résultats obtenus seront comparés à des solutions de : réplication totale sans collaboration des sites, de traitement de la requête au site qui l'a reçue, de minimisation du coût de transfert, de traitement au premier site libre, de traitement au site le plus performant.

Dans la suite des expériences, nous allons étudier la convergence du système à long terme et comparer avec une solution de réplication avec équilibrage de la charge entre sites.

5. Etat de l'art et contributions

Notre travail se positionne dans plusieurs contextes dont les solutions d'exploitation des données de biodiversité, la réplication dynamique de données, l'indexation de données dans les environnements distribués et le traitement parallèle de requêtes dans les systèmes répartis.

5.1. Exploitation des données de biodiversité

Des initiatives visant à améliorer l'exploitation des données de biodiversité ont été menées [3, 4, 7, 8, 9]. La plupart de ces solutions avec des fonctionnalités limitées, préconisent la mise en place de portails dédiés selon une thématique [7] ou selon un pays [4]. Les portails dédiés à des thématiques partagent les données relatives à la thématique et les portails nationaux partagent les données des fournisseurs du même pays. Ce qui est loin de l'objectif initial du GBIF qui est de rendre publiquement accessibles toutes les données de biodiversité à l'échelle mondiale [1, 3]. Notre solution quand à elle, place les données aux endroits où elles sont fréquemment utilisées indépendamment de leurs origines et de leurs thèmes. Les utilisateurs peuvent aussi accéder aux données sans se soucier de leurs emplacements. Ce qui permet de conserver l'objectif initial du GBIF. Notre solution apparait donc comme un complément de ces solutions qui ne partageaient pas directement les données de sources ou thématiques diverses et/ou de ressources de calcul pour répondre à tous les besoins des usagers. Du point de vue des fonctionnalités, des solutions [4, 8, 9] ont intégré des applications limitées d'analyse de données de biodiversité. La plus avancée de ces solutions est MostiquoMap [7] qui est une initiative très intéressante permettant à l'utilisateur de lancer ses analyses via une interface où il choisit des valeurs pour quelques paramètres et visualiser les résultats sous forme de carte. Cependant, cette application reste très rigide et concerne uniquement les analyses sur les moustiques et donc ne tient pas compte de toutes les analyses possibles sur un ensemble plus large notamment toutes les données de biodiversité à l'échelle mondiale. Notre solution préconise l'assistance des usagers dans l'exploitation qu'ils auront à faire des données primaires de biodiversité. En effet, la solution supporte l'intégration de nouvelles fonctionnalités telles que la possibilité de traiter des requêtes plus complexes et la mise en place d'applications d'analyse des données de biodiversité.

5.2. Répartition de données à la demande

Le processus de répartition des données est une phase essentielle dans le contexte des bases de données distribuées et du traitement parallèle. Les performances du système dépendent fortement de ce processus. La fragmentation qui consiste à découper la base globale en un ensemble de morceaux plus petits durant le processus répartition, permet de réduire le volume de données à transférer et le coût de stockage. La réplication qui consiste à placer un fragment dans un ou plusieurs sites a pour objectif d'assurer la disponibilité de système en rapprochant les données des endroits où elles sont utilisées. La plupart des politiques de répartition de données définit le schéma de la fragmentation et d'allocation avant l'exploitation du système. Ces stratégies [10, 11, 12] s'adaptent bien aux contextes où les patterns d'accès ou la charge du système restent statique. Cependant, ils ne s'adaptent pas aux systèmes dynamiques à forte charge fluctuante. C'est ainsi que des mécanismes [10,13] de distribution dynamique de données ont émergés. Loukopoulos et al. [10] ont proposé un algorithme adaptatif d'allocation de données dans le contexte dynamique du web qui supposait que les fragments primaires existaient déjà dans les sites. Récemment, Gope [14] proposa une technique d'allocation dynamique de données pour les systèmes de bases de données distribuées. Cette technique, basée sur un système de non-réplication, réalloue les données avec le respect des changements des patterns d'accès aux données avec la contrainte temporelle. Ce qui suppose que les fragments sont déjà placés dans les sites et aussi qu'ils sont déjà définis. Ces mécanisme diffèrent de notre approche où les fragments sont définis en fonction des prédicats des requêtes, et sont placés aux sites où les requêtes les sollicitant sont soumises et considère la possibilité de réplication d'un fragment au cours du traitement de requête. Avec l'abondance des bandes passantes, le déplacement d'un fragment au cours du traitement peut se faire très rapidement.

5.3. Localisation de données dans un environnement distribué

Une bonne stratégie de localisation des données est décisive pour espérer des performances dans un système de base de données distribué. Des études sont menées dans ce sens dont la plupart consiste à la mise en place d'un index global qui est soit placé en un site accessible aux autres soit répliqué à travers tout le système. Lorsque cet index est placé en un seul site, celui est consulté à chaque fois qu'on veut traiter un fragment distant. Ce qui est très coûteux lorsque les traitements distants sont énormes et ne garantit pas le passage à l'échelle. Notre stratégie de localisations des données permet de retrouver rapidement les fragments à traiter sans surcharger le trafic ou le site qui héberge l'index global. Lorsque l'index est répliqué totalement ou partiellement [16] à travers les sites, les mises à jour fréquentes pour la cohérence des différents index, augmentent fortement le trafic réseau. Notre approche, différente des stratégies totalement décentralisées des systèmes de P2P (DHT, Freenet, Gnutella) qui utilisent une table de hachage ou un mécanisme de diffusion pour la recherche, exploite les avantages de l'index global pour la gestion de la cohérence et décentralise la recherche des fragments souvent demandés pour réduire le trafic réseau, accélérer la recherche et réduire la charge de l'index global. Ce mécanisme est un peu similaire

au principe de RT-CAN [15] qui consiste à utiliser un index global réparti à travers plusieurs super-nœuds qui gère chacun une zone CAN. La différence apparait dans la mesure où chaque nœud de notre système indexe directement et seulement les fragments qu'il traite. De ce fait, l'index global est utilisé pour compléter les index locaux et gérer la cohérence de l'indexation des fragments distants.

5.4. Optimisation du traitement distribué de requête

Le traitement de requêtes dans les environnements distribués est assez complexe, vu les caractéristiques et les paramètres qui peuvent impacter le coût. Plusieurs études [20, 21, 22, 24, 25, 27, 31, 32] sont menées dans ce sens. En 2000, Kosmann et al. [17] ont fait un état de l'art du domaine qui récapitulait une bonne partie des travaux et techniques qui existaient jusqu'à cette époque. La plupart de ces techniques de base sont utilisés dans des technologies récentes pour améliorer l'optimisation du traitement et/ou de l'utilisation de la bande passante. D'autres travaux ont suivi notamment pour le traitement intensif de gros volumes de données. En 2003, Google développa GFS [18] et BigTable[19] pour la gestion de ses applications. Ensuite, il proposa MapReduce [20] qui est un modèle de programmation pour le parallélisme des traitements de gros volumes de données dans de grands clusters de machine en découpant un traitement en un ensemble de tâches qui vont être exécutées parallèlement sur des nœuds différents. Beaucoup d'autres travaux qui ont suivi se sont basés sur les principes de MapReduce. Yahoo développa Hadoop [21] qui est une implémentation de MapReduce et l'a mis libre via Apache Foundation. Hadoop est un système de traitement de données évolutif pour le stockage et le traitement par lot de très grande quantité de données. Il utilise HDFS [23] comme système de gestion de fichiers distribué. Des extensions de Hadoop ont suivi. Hive [22,26] est une infrastructure de datawarehouse basé sur Hadoop et utilisant HDFS. Il intègre de fonctionnalités en proposant le langage de requête HiveQL qui est proche de SQL. Hive a été conçu par Facebook qui l'utilise dans les traitements de ses gros volumes de données [26]. Cependant, il importe de noter que Hive tout comme Hadoop ne présente pas des avantages (en termes de performances) lorsque les données manipulées dans une requête sont de petites tailles (de l'ordre de quelques Méga). Ces solutions basées sur MapReduce, sont critiquées d'être lentes vu leurs mécanismes d'exécution de requêtes dans lequel, elles stockent les résultats intermédiaires dans le système de fichiers (HDFS ou GFS). Ainsi pour bénéficier des avantages des SGBD relationnelles en termes d'optimisation locale des traitements, HadoopDB [27, 28] a vu le jour. Il stocke ses données dans des bases de données relationnelles avec une implémentation de PostGreSql et utilise Hadoop et Hive pour la gestion des communications afin de bénéficier du parallélisme offert par ces systèmes. Comme Hadoop et Hive, HadoopDB aussi n'excelle pas pour les traitements locaux sur de petites quantités de données. Shark [32], Une solution plus récente, basée sur Hadoop et Hive, combine les avantages de ces systèmes avec les fonctionnalités des RDDs (pour Resilent Distributed Datasets) pour charger et partager les données en mémoire. Ses performances sont 40 à 100 fois meilleures que Hadoop [32]. Shark partage avec notre solution les idées de mettre en mémoire les données manipulées (RDDs et SGBD mémoire) et de supporter davantage des fonctionnalités de traitement de requêtes. Cependant, Shark partitionne et alloue les données comme Hadoop, Hive, HadoopDB, avec une fonction de hachage globale. Ce qui l'empêche de s'adapter aux contextes où la distribution se fait à la demande. Ces solutions ne s'adaptent pas donc aux cas d'usages des données de biodiversité où un traitement porte sur une quantité limitée (quelques Mo) de données. D'autres mécanismes d'optimisation de requêtes dans les environnements large échelle ont été proposées [33, 34, 35, 36]. La plupart de ces derniers se focalise sur un type de traitement spécifique ([36], [34]) et ne s'adapte donc pas à un contexte plus général. ESQP [33] propose une solution plus avancé qui poussent les traitements vers les sites qui disposent des données tout en tenant compte de leurs charges pour la sélection de répliques à traiter et suppose un retour asynchrone des résultats à l'utilisateur. Il découpe une requête en un ensemble de sous-requête en fonction des fragments nécessaires. Chaque sous-requête est dupliquée autant de fois que le fragment à traiter dispose de réplique et en fonction de la localisation de la réplique. La sous-requête sera traitée au premier qui est disponible (libre). Cette solution n'a pas considéré le cas où des traitements supplémentaires seraient effectués sur les résultats issus des sous-requêtes ou encore que le site libre serait très faible en termes de performances. Dans notre stratégie d'optimisation, nous avons tenu compte de tous ces aspects qui peuvent affecter les fonctionnalités et les performances. Des travaux ont continués à être menés dans l'optimisation des traitements de requêtes dans les architectures décentralisées. Certains se consacrés aux traitements transactionnels qui exigent la cohérence des données alors d'autres se concentrent sur des traitements analytiques spécifiques. Si les premières solutions ajoutent un surcoût pour les traitements uniquement analytiques, les secondes limitent les fonctionnalités. Ces solutions ne s'adaptent donc pas au contexte de l'interrogation des données de biodiversité où les traitements nécessitent des opérations complexes.

6. Conclusions

Dans ce rapport de mi-parcours, nous avons d'abord présenté le contexte de ma thèse. Nous avons présenté l'architecture et le fonctionnement actuel du système du GBIF. Ensuite, nous avons exploré les traitements généraux communs en analyse de données de biodiversité avec plus de détails sur le cas de la modélisation de niche écologique. Ce qui nous a permis de voir les manquements liés à la nature de l'architecture centralisée du

GBIF qui ne facilite pas le passage à l'échelle des données, des usagers et de leurs traitements et à la nature des traitements supportés qui limitent l'expressivité des requêtes par rapport aux besoins croissants et complexes des usagers. C'est ainsi que nous avons proposé une architecture décentralisée (non-intrusive) dont l'objectif est de supporter davantage de requêtes sans limitation de calcul afin d'assurer le passage à l'échelle des données et des traitements. Nous avons ensuite décrit le mécanisme de fragmentation selon la structure hiérarchique des données de biodiversité et avons proposé une stratégie de réplication à la demande afin de réduire les coûts de stockage et de transfert des données. Nous avons également proposé un mécanisme de traitement de requêtes dans les environnements distribués dont l'objectif d'augmenter à la fois la réactivité du système et le débit transactionnel. C'est ainsi que nous avons proposé une fonction d'optimisation basée sur le coût qui calcule tous les plans possibles et exécute celui qui minimise le temps de réponse. Le coût d'une requête est estimé à la fois en fonction des transferts, des charges et des capacités des sites. Afin d'exploiter au mieux les ressources disponibles, nous avons proposé un algorithme d'avancement de nouvelles requêtes locales lorsque les requêtes distribuées attendent des données distantes. Nous avons commencé l'implémentation du prototype pour le traitement parallèle de requêtes dans un environnement distribué hétérogène et allons comparer notre approche avec d'autres stratégies de traitement distribué de requêtes. Nous avons également établi l'état de l'art de notre travail par rapport à l'exploitation des données du GBIF, la répartition de données à la demande, à la localisation de données dans les systèmes distribués et le traitement de requêtes dans les environnements distribués à très large échelle. Nous comptons, dans un futur proche aborder l'intégration d'autres sources de données notamment environnementales et d'applications d'analyse des données de biodiversité.

7. Travaux à faire dans le futur

Dans la suite du travail de cette thèse, je vais implémenter un prototype (que j'ai déjà commencé d'ailleurs) qui est composé de quatre modules pour optimiser le traitement des requêtes dans l'architecture que nous avons proposée :

- Pour détecter les relations d'inclusions entre fragments
- pour coordonner les lectures de fragments à partir du GBIF et gérer la propagation des mises à jour de fragments modifiés au GBIF
- pour coordonner les traitements locaux et supporter des fonctions de requêtes complexes dans chaque site local afin de bénéficier des avantages de SGBD en mémoire
- pour coordonner les traitements distribués des requêtes d'utilisateurs

Toujours dans l'optique d'optimiser l'utilisation des ressources, nous étudierons la gestion des pannes pour remédier aux éventuelles défaillances de site.
Nous allons également investiguer sur les possibilités de combinaison des données de biodiversité avec d'autres sources de données notamment les données environnementales.
Et pour terminer, nous comptons intégrer des applications d'analyse de données de biodiversité notamment pour la modélisation de niche écologique.

8. Références

[1] D. Hobern, *GBIF Biodiversity Data Architecture*, *GBIF Data Access and Database Interoperability (DADI)*, 2003.
[2] GBIF Secretary, GBIF Annual Report 2012, pages 08-11, 2012.
[3] GBIF international www.gbif.org, data.gbif.org
[4] GBIF France, www.gbif.fr. Et Canadian BIF, www.cbif.gc.ca
[5] H. Saarenma. *Sharing and Accessing Biodiversity Data Globally through GBIF*, *ESRI User Conf.*, 2005.
[6] *The GBIF Data Portal: A practical "hands-on*" accessible au http://data.gbif.org.
[7] *MosquitoMap, www.mosquitomap.org*
[8] http://uat.gbif.org/newsroom/uses, 20/09/2013
[9] Map of Life au www.mappinglife.org
[10] T. Loukopoulos and I. Ahmad, *Static and adaptive data replication algorithms for fast information access in large distributed systems*, ICDCS, pp 385-392, 2000
[11] P.M.G. Apers, "Data Allocation in Distributed Database Systems," ACM Trans. on Database Systems, Vol.13, No.3, September 1988.
[12] Y.K. Kwok, K. Karlapalem, I. Ahmad and N.M. Pun, "Design and Evaluation of Data Allocation Algorithms for Distributed Database Systems", IEEE Journal on Sel. areas in Commun.(Special Issue on Distributed Multimedia Systems), Vol. 14, No. 7, pp. 1332-1348, Sept. 1996.

[13] S. Kamali and al. , 'Dynamic data allocation with replication in distributed systems', Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30 th International, pp 1-8 , 2011
[14] D. C. Gope, ' Dynamic Data Allocation Methods in Distributed Database System', American Academic & Scholarly Research Journal, Vol. 4, No. 6, Nov 2012
[15] J. Wang and al. ,' Indexing Multi-dimensional Data in a Cloud System', SIGMOD, 2010
[16] Sai Wu and K. L. Wu, 'An Indexing Framework for Efficient Retrieval on the Cloud', IEEE,
[17] D. Kossmann et al., *The State of the Art in Distributed Query Processing*, ACM Computing Surveys, 2000, pp. 422–469.
[18] S. Ghemawat et al., *The Google file system*, SIGOPS Oper. Syst. Rev. 37, 2003.
[19] F. Chang and al. , 'BigTable: A Distributed Storage System for Structured Data', OSDI'06: seventh Sysmposium on Operating System Design and Implementation, Seattle, Wa, November 2006
[20] J. Dean, S. Ghemawat. *MapReduce: simplified data processing on large clusters*, *CACM*, 2008.
[21] Apache Hadoop http://wiki.apache.org/hadoop.
[22] Hive wiki http://wiki.apache.org/hadoop/hive.
[23] D. Borthakur, *The Hadoop Distributed File System: Architecture and Design,* The Apache Software Foundation, 2007
[24] M. Brantner et al. Building a Database on S3, *SIGMOD*, 2008.
[25] D. J. Abadi: *Data Management in the Cloud: Limitations and Opportunities*. *IEEE Data Eng. Bull.* 32(1), 2009.
[26] A. Thusoo et al. Hive - A Petabyte Scale Data Warehouse Using Hadoop, *ICDE,* 2010.
[27] K. Bajda-Pawlikowski et al., *Efficient processing of data warehousing queries in a split execution environmen*t. *SIGMOD,* 2011:
[28] A. Abouzied et al. *HadoopDB in action: building real world applications*. *SIGMOD,* 2010.
[29] J. P. Sampoux, V. Badeau , Modélisation de la niche écologique des fétuques à feuilles fines : quels apports pour la conservation et la valorisation des ressources génétiques ?, Innovations Agronomiques, 7, 79-91, 2009
[30] H2 Database Engine http://www.h2database.com.
[31] D. Agrawal, S. Das, A. El Abbadi, 'Big data and cloud computing: current state and future opportunities'. *EDBT* 2011.
[32] R. S. Xin and al., 'Shark: SQL and Rich Analytics at Scale' , SIGMOD, 2013
[33] J. Zhao, '*ESQP: An Efficient SQL Query Processing for Cloud Data Management*', CloudDB'10, October 30, 2010, Toronto, Ontario, Canada, 2010
[34] M. Liu, 'Efficient Optimization and Processing for Distributed Monitoring and Control Applications', SIGMOD/PODS'12, pp 69-73, 2012
[35] L. Rupprecht, 'Exploiting In-Network Processing for Big Data Management', SIGMOD'13, 2013
[36] D. Cokuslu and al., 'Resource Allocation Algorithm for Relational Join Operator in Grid System', IDEAS12, 2012