Chapitre 7
SQL dans Hadoop

1. Introduction

« La valeur d'une technologie est proportionnelle au carré du nombre de personnes qui l'utilise. » (loi de Metcalfe)

La solution conceptuelle au problème du traitement de données dans l'économie numérique est la suivante : les traitements/calculs doivent être divisés en tâches et leur exécution doit être parallélisée dans un cluster d'ordinateurs complètement tolérant aux pannes. Dès lors, l'approche de centralisation qui a prévalu jusqu'ici est tout simplement inenvisageable. La tolérance aux pannes est fournie par un tout nouveau type de système de fichiers appelé "Système de Fichiers distribué" (DFS), et le découpage et le parallélisme des tâches se font à l'aide d'un nouveau modèle de programmation appelé le MapReduce. Une application générique de gestion de ressources comme le YARN ou MESOS permet d'aller au-delà du MapReduce et d'utiliser d'autres modèles de calcul sur le cluster. Hadoop, l'implémentation MapReduce la plus populaire et la plus mature du marché, est en passe de devenir le standard de facto dans l'ère numérique. Nous pouvons dire sans prendre de risque qu'Hadoop va devenir la plateforme de traitement de données par défaut des utilisateurs, un peu comme Excel est progressivement devenu le logiciel par défaut d'analyse de données. Problème : à la différence d'Excel, Hadoop n'a pas été, à la base, conçu pour être utilisé par les *analystes métier*, mais par les développeurs. Nous entendons par analyste métier toute profession liée à un métier qui implique de près ou de loin le traitement de données. Par exemple, le statisticien, l'analyste marketing, l'analyste de crédit, le contrôleur de gestion, le comptable, l'analyste financier… Selon la loi de Metcalfe, *la valeur d'un standard est proportionnelle au carré du nombre de systèmes qui l'utilise*.

Nous pouvons contextualiser cette citation en disant que *la valeur d'une technologie est proportionnelle au carré du nombre de personnes qui l'utilise*. En d'autres termes, l'adoption à grande échelle et le succès d'Hadoop ne dépendent pas des développeurs, mais des analystes métier ! La fondation Apache a bien compris cela, c'est pourquoi, depuis qu'elle a repris Hadoop en 2009, elle s'évertue à rapprocher le plus que possible le MapReduce et ses dérivés des utilisateurs métier. Le marché étant fortement concurrentiel, la réponse des éditeurs logiciels ne s'est pas fait attendre. Ainsi, sur le marché, deux catégories d'acteurs ont pris à cœur cet objectif : le monde de l'open source, centralisé autour de la fondation Apache, et le monde des éditeurs logiciels. Les deux offrent un ensemble d'outils autour d'Hadoop qu'il est aujourd'hui raisonnable de qualifier d'**écosystème Hadoop**. Dans ce chapitre, vous allez avoir un aperçu des outils qui constituent l'écosystème Hadoop, et vous verrez en quoi ces outils facilitent l'adoption d'Hadoop par un public non développeur. Puis il sera question de l'exploitation de SQL (*Structured Query Language*) sous Hadoop. Pourquoi SQL sur Hadoop ? Tout simplement parce que SQL est le langage favori des utilisateurs métier et que pour qu'Hadoop séduise le cœur des analystes métier, il faut qu'il donne la possibilité à ceux-ci d'utiliser leur langage favori. Ce qui est chose faite avec les outils comme Hive, Pig, Impala ou encore Apache Phoenix.

2. Étude de l'écosystème Hadoop

Hadoop est une plateforme qui implémente le modèle de calcul MapReduce et fournit un système de fichiers distribué redondant, fiable et optimisé pour la gestion des fichiers volumineux : le HDFS. En réalité, Hadoop est un ensemble de classes écrites en Java pour la programmation des tâches MapReduce et HDFS. Ces classes peuvent tout aussi bien s'écrire dans d'autres langages tels que C#, Scala, etc. Ces classes permettent à l'analyste d'écrire des fonctions qui vont traiter les données sans avoir à se préoccuper de la façon dont ces fonctions sont distribuées et parallélisées dans le cluster. Pour tirer le meilleur parti d'un cluster Hadoop, la fondation Apache a inclus dans Hadoop une série de logiciels et d'outils. Cet ensemble forme aujourd'hui ce qu'il est judicieux d'appeler l'*écosystème Hadoop* ou le *framework Hadoop*. Pourquoi ? Parce que sans ces outils, il reviendrait à chaque entreprise en fonction de son besoin, de développer elle-même des outils compatibles avec Hadoop afin de déployer ses solutions sur le cluster ; ce qui serait une "daunting task" (tâche herculéenne).

Actuellement, pas mal de développeurs sont en train de travailler sur l'écosystème Hadoop et offrent leurs travaux à la fondation Apache. Yahoo! a développé *ZooKeeper*, un service de coordination distribué, qu'il a donné à la fondation Apache Cloudera a développé le moteur de calcul SQL massivement parallèle *Impala* et l'a passé à la fondation Apache. LinkedIn a développé un système de messagerie publish/subscribe distribué, *Kafka*, etc.
C'est la contribution de tous ces acteurs qui constitue aujourd'hui l'écosystème Hadoop. L'écosystème Hadoop fournit une collection d'outils et de technologies spécialement conçus pour faciliter le développement, le déploiement et le support des solutions Big Data. La définition de cet écosystème est importante, car elle facilite l'adoption d'Hadoop et permet aux entreprises de surmonter les défis du Numérique.

L'écosystème Hadoop est constitué d'outils qui peuvent être rangés par catégories en fonction de la tâche que chacun prend en charge. Quatorze catégories d'outils peuvent être distinguées : les langages d'abstraction (Hive, Pig, Cascading), SQL sur Hadoop (Impala, Phoenix, HawQ), les modèles de calcul (MapReduce, Spark, Mahout, Hama, Tez), les outils de traitement en temps réel (Storm, Samza, S4), les bases de données (HBase, Cassandra), les outils de gestion des données en streaming (Kafka, Flume), les outils d'intégration des données (Sqoop), les outils de coordination de workflows (Oozie), les outils de coordination de services distribués (ZooKeeper), les outils d'administration de clusters (Ambari), les systèmes de fichiers distribués (HDFS), les gestionnaires de ressources (YARN, Mesos), les outils d'indexation de contenu (Lucy, Solr, Lucene) et les interfaces utilisateur (HUE). Un écosystème Hadoop classique est composé d'un produit de chaque catégorie. La figure suivante illustre la composition d'un écosystème Hadoop.

Attention ! Les outils de l'écosystème Hadoop varient beaucoup. Certains outils sont ajoutés, d'autres sont modifiés et d'autres sont simplement supprimés. Tous les outils que nous avons cités ici ne tiennent pas compte des projets en cours d'incubation. Les projets en cours d'incubation sont des contributions logicielles qui n'ont pas encore été testées et validées par la fondation Apache. Considérez donc cet écosystème comme un guide illustratif et non exhaustif.

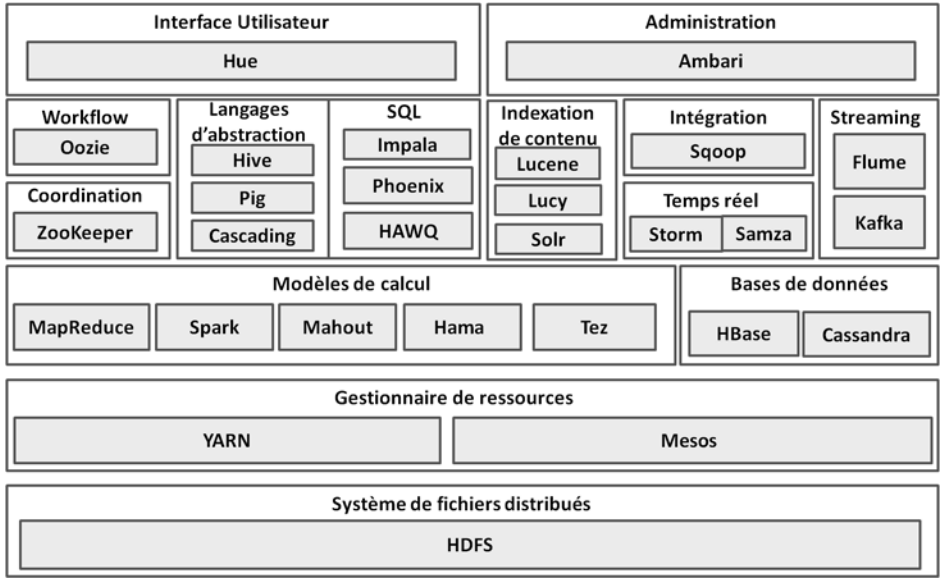

***Figure 33** : Aperçu de la cartographie de l'écosystème Hadoop.*

Nous allons maintenant passer au vif du sujet : SQL sur Hadoop. Il existe deux façons d'exploiter SQL dans Hadoop : soit vous utilisez les langages d'abstraction, soit vous utilisez les moteurs natifs SQL massivement parallèles. Ces deux types d'outils, même s'ils ont la même fonction, sont radicalement différents dans leur structure et leur mode de fonctionnement. Nous allons commencer par l'étude des langages d'abstraction.

3. Langages d'abstraction

Il y a des raisons de croire que le MapReduce va devenir le mode normal de traitement des données dans l'ère numérique et donc qu'Hadoop va devenir l'outil par défaut du traitement de données. Le problème est que le MapReduce est un langage de très bas niveau, c'est-à-dire très proche de la machine. Il implique que le développeur sache interagir avec le cluster, ce qui peut être très difficile pour un développeur néophyte dans le monde du traitement parallèle, ou pour des utilisateurs métier. L'un des moyens de simplifier le développement MapReduce, et Hadoop en général, consiste à fournir ce qu'on appelle un ***langage d'abstraction***. Un langage d'abstraction est un langage à syntaxe relativement proche du langage humain qui permet d'exprimer des problèmes métier sous forme de requêtes simples.
L'abstraction vient du fait que lorsque l'utilisateur exprime son besoin sous forme d'une requête, cette requête est transformée en instructions machine. Ainsi, le langage d'abstraction n'est en réalité qu'une couche qui masque la complexité d'expression des problèmes en langage de bas niveau comme le ferait un développeur. *Plus le niveau d'abstraction offert par le langage est élevé, et plus on est éloigné de la machine, ce qui est plus simple pour les utilisateurs*. La fondation Apache fournit pour le moment trois langages d'abstraction pour le MapReduce : **Hive**, **Pig** et **Cascading**. Ces trois langages, conçus pour un public non développeur, permettent d'exprimer des jobs MapReduce dans un style de programmation similaire à celui de SQL, familier aux utilisateurs. Par la suite, ces langages transforment les requêtes écrites en jobs MapReduce qui sont soumises au cluster pour leur exécution. Globalement, Hive offre un langage de plus haut niveau d'abstraction que Pig et Pig offre une abstraction de plus haut niveau que Cascading. Dans cet ouvrage, nous étudierons uniquement SQL, Hive et Pig. Cascading étant de trop bas niveau pour les analystes métier uniquement habitués à SQL. La figure suivante illustre la relation entre le niveau d'abstraction des langages, la proximité de l'utilisateur vis-à-vis du MapReduce, et la complexité de programmation. Bien évidemment, plus on "s'éloigne" du MapReduce, mieux pour l'utilisateur.

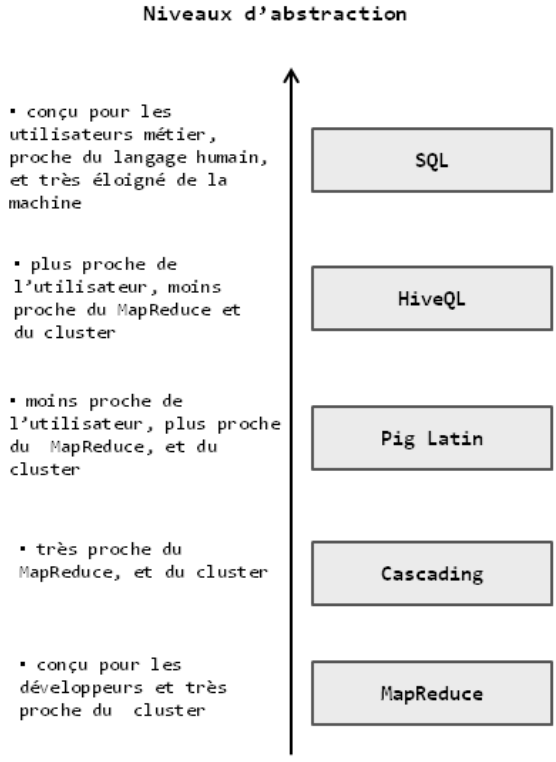

***Figure 34** : Relation entre le niveau d'abstraction du langage et proximité vis-à-vis du cluster. Plus le niveau d'abstraction est élevé, plus le langage est "proche" de l'utilisateur.*

Attention, plus le niveau d'abstraction d'un langage est élevé et plus le niveau de complexité des requêtes qu'on peut exprimer est faible. C'est pourquoi il est bénéfique que vous appreniez à la fois Hive et Pig. Ainsi, si vous n'arrivez pas à exprimer votre problème sous forme de requêtes Hive, alors vous pourrez toujours utiliser Pig. Si vous voulez aller plus loin, vous pouvez aussi apprendre Cascading.