Pourquoi NoSql ?

Les données

Depuis le début des années 60 les informaticiens ont centré une grande part de leur activité sur la manipulation et le stockage de données. Au fil des années, des logiciels permettant de faciliter cette tâche de collecte et de restitution ont été utilisés comme le système de gestion de bases données (SGBD) qui s'appuie sur des formalismes mathématiques éprouvés : l'algèbre relationnelle et les normes[1].

Simplement, un SGBD relationnel range ses données dans des tables contenant des colonnes définissant ainsi un schéma de données qui doit être préalablement défini avec un langage particulier ; le SQL.

Des évolutions majeures

De nombreuses sociétés fournissent des SGBDR avec différentes versions de SQL ; Oracle et Sybase sont actuellement les plus vendus. En mode open-source, Mysql et Postgresql gagnent chaque année des parts de marché.

Les applications ont évolué, plus de données sont enregistrées pour produire des statistiques et des rapports pour affiner la stratégie commerciale.

Ce mouvement a pris de l'ampleur avec la chute des prix des disques et de la mémoire, de plus l'utilisation d'applications internet a décuplé le volume de données à sauvegarder. Le prochain concept des objets connectés (*IoT*) va dépasser les limites de stockage actuelles.

En retard, les bases de données n'ont pas suivi ce mouvement, construites à partir d'une structure de données figée préalablement définie, il est très difficile d'en changer l'organisation lorsque d'importants volumes sont présents. C'est pour palier ce problème que certaines bases de données contiennent des enregistrements au format XML dans des champs de type textuel ou binaire qui sont évidement très difficiles à maintenir.

Toutes ces évolutions ont bousculé les équipes de développement et d'administration confrontées à de nouveaux problèmes, comme par exemple : « *comment sauvegarder des milliers de GigaBytes de données et de les répliquer ensuite vers un site de reprise d'activités* ? » Pas facile n'est-ce pas ?

Pourquoi Nosql ?

Conscientes des enjeux, les grandes sociétés de l'internet, comme Google et Amazon, ont créé en interne leur propre gestionnaire de données. En 2006, Google a publié un document technique sur leur système BigTable[2], plusieurs projets open-source se sont appropriés ces concepts ; une multitude de gestionnaires NoSQL ont ainsi vu le jour.

Le terme *NoSql* est ainsi né. Il ne signifie pas que vous ne devez plus utiliser SQL mais vous propose plutôt des alternatives de stockage de données qui sont « *not only sql* ». La théorie sous-jacente de la majorité des systèmes Nosql est le *CAP theorem*, formalisé en 1998 qui signifie :

[1] http://en.wikipedia.org/wiki/Database_normalization
[2] http://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf

- **Consistency** ou la consistance des données faites par leur mise à jour avant leur lecture ;
- **Availability** ou la disponibilité, assurée par une réplication de la base de données sur différents serveurs physiques ou virtuels en mode cluster;
- Et enfin, le **Partitionning** qui est la division de blocks de données suivants des critères fonctionnels pour en faciliter l'accès.

Seuls les couples de lettres CP et AP sont assurés par les applications NoSQL, le couple CA correspond plutôt aux bases de données traditionnelles.

Le stockage des données est fait simplement par une clé d'accès et une valeur qui prennent place dans une structure de type de containers « *document* » ou « *column family* » en fonction de l'application Nosql que vous utilisez. Par abus de langage, les bases Nosql sont dites « *schema-less* », c'est-à-dire sans schéma, mais il faudra toujours définir les containers et appliquer des normes pour obtenir des performances optimales.

Avantage, inconvénients

Les avantages sont réels, une plus grande flexibilité, l'ajout ou le retrait d'une structure de données est très simple. La duplication de données sur un ensemble de serveurs offre la possibilité de limiter les sauvegardes et leur déport vers un site de reprise d'activités. Des solutions de filtrage par une base NoSQL en mémoire pour mettre en échec des attaques par déni de service[3] et des capacités de stockage énorme peuvent être envisagées.

Au registre des inconvénients, c'est évidement une technologie qui n'est pas encore mature. Le mouvement a commencé il y a seulement une dizaine d'années, contre plus d'une trentaine d'années pour les SGBDr. Autre problème, c'est un rêve pour les développeurs d'utiliser une base NoSQL mais un cauchemar pour les administrateurs qui doivent jongler avec les aspects « métier » des données et les contraintes physiques des serveurs pour réaliser un partitionnement efficace[4]. De plus, les aspects transactionnels et les retours arrière ne sont que rarement supportés.

Les différents acteurs

Je vous renvoie vers la page web qui est très bien faite, les bases Nosql sont triées par type de containers utilisés : http://nosql-database.org/

Conclusion

Pour favoriser la migration de votre système SGDBr vers NoSQL allez voir du côté de PostgreSQL, la dernière version 9.4 contient un nouveau type de donnée « *document* » au format JSON[5] ; c'est une gestion presque équivalente à la base NoSQL MongoDB.

Autre exemple, la base de données MariaDB, une version vraiment libre de Mysql, fournit un moteur de stockage Cassandra[6] offrant la possibilité de faire des requêtes SQL sur des « columns family » .

[3] http://voltdb.com/news/sakura-internet-defeats-ddos-attacks-using-voltdb%E2%80%99s-high-speed-memory-database-1
[4] Scaling MongoDB, Kristina Chodorow, editeur O'Reilly
[5] http://www.pgcon.org/2014/schedule/attachments/328_9.4json.pdf
[6] https://mariadb.com/kb/en/mariadb/documentation/storage-engines/cassandra/cassandra-storage-engine-overview/

Essayons de pressentir l'avenir de tout ça. Ce que nous apprend le mouvement Nosql c'est le retour vers une structure de données simple : une clé-valeur, des données dupliquées, redondantes et distribuées, stockées en mémoire et/ou sur disques le tout orchestré par des communications asynchrones, ne serait-ce pas là la base d'un futur système d'exploitation ?

Aujourd'hui, les habitudes ont la vie dure, il sera difficile de passer d'une base SQL existante vers une solution NoSQL mais les enjeux sont là, il faut être prêt pour demain.