Dans les coulisses de Google BigQuery

Aurélie Vaché
Développeuse Web chez atchikservices à Toulouse, Duchess Toulouse et contributrice pour le blog Duchess France. @aurelievache

"Big Data" != "Hadoop"

Lorsque l'on pense à des technologies liées à la Big Data, on pense de suite à l'écosystème **Hadoop**, ou bien à **Elasticsearch,** ou bien ces temps-ci, beaucoup à **Spark**. Mais il y a un « petit service » de Google qui ne fait pas beaucoup parler de lui qui peut cependant tirer son épingle du jeu dans différents cas de figure. Pour ma part, comme beaucoup d'entreprises, nous avons besoin de faire parler la donnée pour nos équipes en interne et nos clients. La première architecture technique que nous avions mise en place reposait sur l'écosystème Hadoop avec un cluster sous Cloudera d'une dizaine de serveurs hébergés. Les performances étaient au rendez-vous, mais le temps passé pour la maintenance et l'opérationnel est élevé. Lorsque BigQuery fut assez mature et répondait à nos besoins sur le papier, nous avions décidé de créer une branche de notre projet, et de tester cette nouvelle technologie de Google.

Résultats : une centaine d'euros d'économie par mois pour l'électricité, les coûts concernant l'opérationnel sur ce projet ont disparu, et la facture de Google s'élève à environ **0,20$ par mois** pour des performances au final qui sont comparables avec celles dans le cluster Hadoop, et qui répondent toujours au besoin des utilisateurs finaux.

Histoire Fig.1

Comme beaucoup d'outils et de services, BigQuery a été conçu pour résoudre un problème. Les ingénieurs de Google avaient du mal à suivre le rythme de croissance de leurs données. Le nombre d'utilisateurs de Gmail augmentait constamment et était de l'ordre de centaines de millions; en 2012, il y avait plus de 100 milliards de recherches Google effectuées chaque mois. Essayer de donner un sens à toutes ces données prenait un temps fou et était une expérience très frustrante pour les équipes de Google. Ce problème de données a conduit à l'élaboration d'un outil interne appelé **Dremel**, qui a permis aux employés de Google d'exécuter des requêtes SQL extrêmement rapides sur un grand ensemble de données. Selon Armando Fox, professeur d'informatique à l'Université de Californie à Berkeley, *« Dremel est devenu extrêmement populaire chez Google. Les ingénieurs de Google l'utilisent des millions de fois par jour. »* Le moteur de requêtes Dremel a créé une façon de paralléliser l'exécution des requêtes SQL sur des milliers de machines. Dremel peut scanner 35 milliards de lignes sans un index en une dizaine de secondes. Il existe deux technologies que Dremel utilise pour atteindre la lecture d'1 To de données en quelques secondes. Le premier est appelé Colossus : un système de fichiers distribués et parallélisables développé chez Google comme un successeur de Google File System (GFS). Le second est le format de stockage, appelé ColumnIO, qui organise les données d'une manière à ce que ce soit plus facile de les interroger. En 2012, lors du Google I/O, Google a lancé publiquement **BigQuery**, qui a permis aux utilisateurs en dehors de Google de profiter de la puissance et la performance de Dremel.

Fig.3

Fig.1

BigQuery, c'est quoi ?

Google BigQuery est une solution de type **AaaS - Analytics as a service** qui repose sur la plateforme Cloud de Google et de ce fait sur sa puissance de calcul. Grâce à BigQuery on peut stocker, effectuer des requêtes et analyser des grands volumes de données : requêter des tables de plusieurs Tera/Peta Octets de données ne prend que quelques secondes. Cette solution s'intègre bien avec Google App Engine mais également avec d'autres plateformes. Techniquement, le service BigQuery est juste un serveur qui accepte les requêtes HTTP et renvoie les réponses au format JSON. Il communique avec Dremel, le moteur de requêtes qui communique avec Colossus Fig.2.


Fig.2

Pourquoi utiliser BigQuery ?

- Sécurisé,
- SLA 99.9% (https://cloud.google.com/bigquery/sla),
- Infrastructure de Google,
- Pas de coût de serveurs, d'opération et de maintenance,
- Moins complexe que l'écosystème Hadoop,
- BigQuery SQL,
- Scalabilité,
- Rapide,
- « Pay only for what you use »,
- Requêtes synchrones et asynchrones,
- Facilité d'interconnexion avec outils tierces.

Inconvénients/Limitations

- « Append-only tables » : on ne peut pas modifier (UPDATE) ou supprimer (DELETE) des entrées dans une table, seulement y ajouter des entrées,
- Des latences réseau peuvent se produire,
- Performant sur des énormes tables (Go, To, Po), moins sur de petites tables.

Quotas

Requêtes : 20 000 requêtes par jour (et jusqu'à 100 To de données)
Chargement/Upload de données :

- Limite quotidienne : 1000 jobs de chargement par table par jour, 10 000 jobs de chargement par projet et par jour (y compris les échecs pour les deux).
- 1000 fichiers max par job de chargement.
- Taille maximum par type de fichier : Fig.3.

Streaming : 100 000 lignes insérées par seconde par table maximum.
A noter que toutes les données qui sont en cache, ne sont pas comptées dans les quotas.
Vous pouvez consulter la liste complète des quotas sur le site de Google BigQuery qui peut être sujet à des changements : https://cloud.google.com/bigquery/quota-policy.

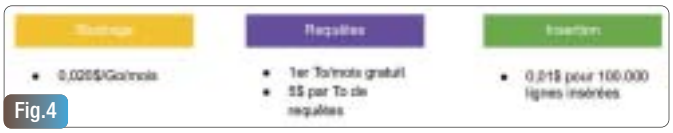


Fig.4

Coûts

Les utilisateurs de BigQuery sont actuellement facturés pour deux choses : le stockage et les requêtes. Les deux coûts sont proportionnels à la taille des données.
Importer des données via un fichier CSV ou JSON est gratuit Fig.4.
Les prix sont sujets à changement donc veuillez vous tenir informés sur le site officiel : https://developers.google.com/bigquery/pricing.
Depuis le 12 Août 2015, les frais d'insertion ont subi une modification : Google ne facture plus par lignes insérées mais par octets insérés, il faut donc compter 0.01$ pour 200 Mo insérés.

Architecture technique

Comme on l'a déjà vu ci-dessus, BigQuery est une implémentation externe de Dremel. Pour essayer de comprendre en quelques mots comment fonctionne BigQuery, il faut connaître les deux technologies de base de Dremel :

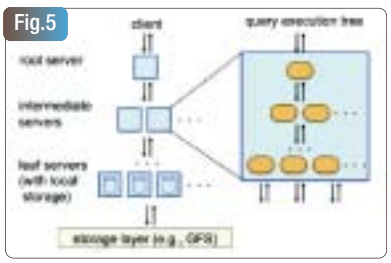


Fig.5

Source : Google BigQuery Technical White-paper

Architecture en arbre Fig.5.

Ce type d'architecture est utilisé pour dispatcher les requêtes et agréger les résultats à travers des milliers de machines en quelques secondes.

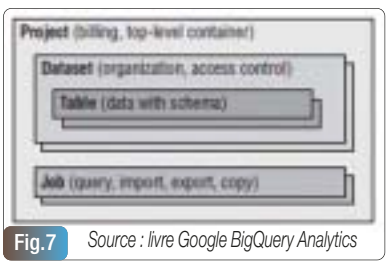


Fig.7 *Source : livre Google BigQuery Analytics*

Fig.8

Base de données orientée colonne Fig.6.

Dremel utilise BigTable et ColumnIO (des bases de données orientées colonne) et stocke les données sous forme de colonne. Grâce à cela, on ne charge que les colonnes dont on a vraiment besoin. Dremel stocke les enregistrements par colonne sur des volumes de stockage différents, alors que les bases de données traditionnelles stockent normalement l'ensemble des données sur un seul volume. Cela permet un taux de compression très élevé.

Composants Fig.7.

- Projects : Toutes les données dans BigQuery sont stockées dans des projets. Un projet est identifié par un ID unique ainsi qu'un nom. Il contient la liste des utilisateurs autorisés, les informations concernant la facturation et l'authentification à l'API.
- Dataset : Les tables sont groupées par dataset qui peut être partagé à d'autres utilisateurs. Un dataset peut contenir une ou plusieurs tables.
- Table : Les tables sont représentées comme ceci : <project>:<dataset>.<table_name>. Pour comprendre cette règle de nommage, regardons de plus près une requête SQL :

```
SELECT date, title FROM [programmez:184.article]
```

Les champs date et title sont sélectionnés dans la table *article* qui est dans le dataset *184* qui est dans le projet programmez.
Lorsque l'on crée une table il faut définir son schéma. Exemple : Fig.8.

```
date:TIMESTAMP, author:STRING, title: STRING, word_count:INTEGER
```

Premiers pas

Si vous n'avez jamais utilisé Google BigQuery, ce paragraphe est fait pour vous, sinon, vous pouvez passer au à la section suivante.

- Il faut en tout premier accéder à la Google API Console : https://console.developers.google.com/ Fig.9.
- Vous pouvez créer votre tout premier projet en cliquant sur le bouton "Créer un projet" : Fig.10.
- Il faut maintenant que vous cliquiez sur le lien "Profiter d'un essai gratuit" afin de pouvoir accéder aux services de Cloud de Google, dont BigQuery. Une fois le formulaire rempli (attention même l'essai gratuit nécessite de fournir les informations d'une carte bancaire), vous devez cliquer sur le projet que vous avez créé pour pouvoir accéder à Google BigQuery puis créer votre premier dataset si vous le désirez, et effectuer vos premières requêtes Fig.11 et 12.

Moyens d'accès

Pour communiquer avec BigQuery il existe plusieurs moyens :

- Via la Google API Console (Web UI : https://bigquery.cloud.google.com/)

Cette interface permet d'effectuer la plupart des opérations dans l'API : lister les tables disponibles, afficher leur schéma et leurs données, partager des dataset avec d'autres utilisateurs, charger des données et les exporter vers Google Cloud Storage. La console de Google est très pratique lorsque l'on veut exécuter/tester/fine-tuner (optimiser) des requêtes. En quelques clics on crée sa requête, on l'exécute et on obtient les résultats en quelques secondes Fig.13.

- En ligne de commande :

Si l'on souhaite par exemple ajouter une colonne dans une table, en ligne de commande c'est possible.


Fig.9


Fig.10

Source : livre Google BigQuery Analytics

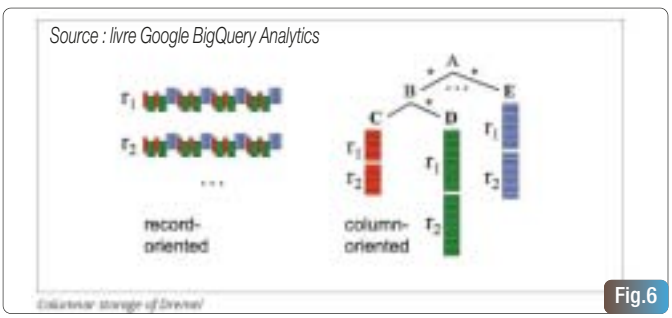


Fig.6

0. Pré-requis : vous devez installer ou avoir Python d'installé sur votre machine. Puis vous devez vous authentifier :

```
gcloud auth login
```

1. Il suffit de récupérer le schéma de la table :

```
bq --format=prettyjson show yourdataset.yourtable > table.json
```

2. Il faut maintenant modifier le fichier, tout supprimer excepté le schéma (garder [{ « name »: « x » ... }, ...]) et ajouter la nouvelle colonne à la fin par exemple.

3. Mettre à jour le schéma de sa table :

```
bq update yourproject.yourtable table.json
```

- Via les API :

– Il existe une dizaine de librairies clientes qui sont mises à disposition par Google. Elles sont disponibles pour Java, Python, PHP, Ruby, .NET ...
– Et bien entendu il existe une API de type REST.
A noter que les APIs utilisent l'API d'authentification OAuth2.

- Via des connecteurs JDBC (Starschema),
- Via le connecteur Excel,
- Via le connecteur pour Hadoop,
- Via le connecteur pour le nouveau service de Google : Google Cloud DataFlow.

Fig.13

Source : livre Google BigQuery Analytics

Chargement des données

Il existe deux moyens pour charger des données dans une table :

- Soit, via la Google console, lors de la création de la table, on définit une source de données. Ce peut être un fichier au format CSV, JSON ou AppEngine Datastore, que vous uploadez ou que vous importez du Google Cloud Storage Fig.14.

Puis il faut bien penser à définir le schéma de cette table : Fig.15.
Le job qui a procédé au chargement des données et à la création de la table a bien fonctionné : Fig.16.
* Soit, via l'API, on peut charger un fichier CSV, ou JSON également, ou bien insérer des entrées dans une table :

```
List<TableDataInsertAllRequest.Rows> rowList = new
ArrayList<TableDataInsertAllRequest.Rows>();
rowList.add(new TableDataInsertAllRequest.Rows()
   .setInsertId(""+System.currentTimeMillis())
   .setJson(new TableRow().set("adt", null)));

TableDataInsertAllRequest content = new TableDataInsertAllRequest().setRows(rowList);
TableDataInsertAllResponse response = bigquery.tabledata().insertAll(
   PROJECT_NUMBER, DATASET_ID, TABLE_ID, content).execute();
System.out.println("kind="+response.getKind());
System.out.println("errors="+response.getInsertErrors());
System.out.println(response.toPrettyString());
```

BigQuery SQL

Les requêtes sont écrites en utilisant une variante de l'instruction SQL SELECT standard. BigQuery prend en charge une grande variété de fonctions telles que COUNT, les expressions arithmétiques, et les fonctions de chaîne. Vous pouvez consulter la page Query Reference (https://cloud.google.com/bigquery/query-reference#top-function) pour tout savoir sur le langage de requête de BigQuery. Tout comme le SQL standard, l'instruction SELECT s'écrit de cette manière :

```
SELECT expr1 [[AS] alias1] [, expr2 [[AS] alias2], ...]
   [agg_function(expr3) WITHIN expr4]
   [FROM [(FLATTEN(table_name1|(subselect1)] [, table_name2|(subselect2), ...)]
   [[INNER|LEFT OUTER|CROSS] JOIN [EACH] table_2|(subselect2) [[AS] tablealias2]
    ON <em>join_condition_1</em> [... AND <em>join_condition_N</em> ...]]+
```

Fig.15

Fig.14

Fig.12

Fig.11

Fig.16

```
[WHERE condition]
[GROUP [EACH] BY field1|alias1 [, field2|alias2, ...]]
[HAVING condition]
[ORDER BY field1|alias1 [DESC|ASC] [, field2|alias2 [DESC|ASC], ...]]
[LIMIT n]
;
```

A noter quelques fonctionnalités intéressantes de BigQuery SQL :

- TABLE_DATE_RANGE :

Au lieu de lister toutes les tables quotidiennes dans votre SELECT, comme ceci :

```
SELECT TIMESTAMP_TO_MSEC(rdt), uid, uname FROM myproject.MEETUP_20150317, myproject.MEETUP_20150318, myproject.MEETUP_20150319 WHERE rdt >= '2015-03-17 16:45:00' AND rdt < '2015-03-19 10:50:00' ORDER BY uname LIMIT 500
```

Voici ce qu'il est possible de faire avec cette fonction de wildcard :

```
SELECT TIMESTAMP_TO_MSEC(rdt), uid, uname FROM (TABLE_DATE_RANGE(myproject.MEETUP_, TIMESTAMP('2015-03-17'), TIMESTAMP('2015-03-19'))) WHERE rdt >= '2015-03-18 16:45:00' AND rdt < '2015-03-19 10:50:00' ORDER BY uname LIMIT 500
```

Lorsqu'il s'agit de faire un SELECT sur une liste de tables quotidiennes sur plusieurs jours/semaines/mois, imaginez le gain pour l'écriture de la requête et les performances avec cette fonction.

- REGEXP_MATCH :

Vous pouvez également utiliser des expressions régulières directement dans vos requêtes :

```
SELECT TIMESTAMP_TO_MSEC(rdt), uid, uname FROM [myproject.mytable_20141108] WHERE cl=18 AND rdt >= '2014-11-07 23:00:00' AND rdt < '2014-11-08 22:59:00' AND REGEXP_MATCH(uid, '(?i)^(123456789|55884772)$') AND REGEXP_MATCH(mid, '(?i)^(7423456|3465465415)$') ORDER BY uname LIMIT 123
```

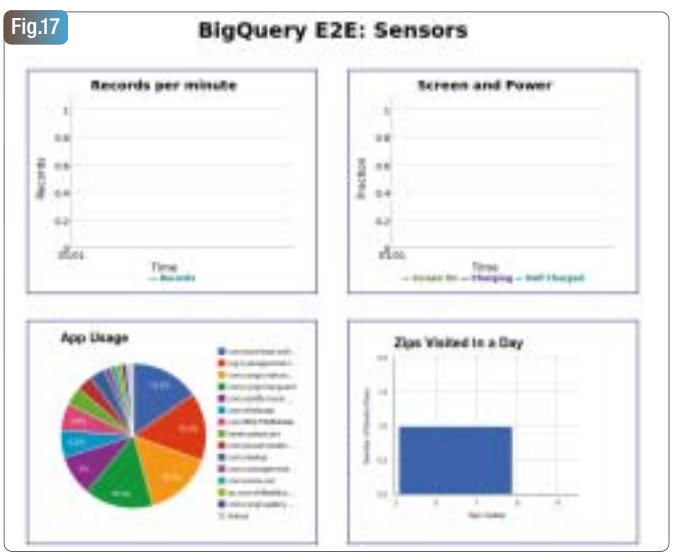
Cas d'utilisation

Il existe plusieurs solutions pour visualiser les données qui sont stockées dans BigQuery.

- Utiliser Google App Scripts pour écrire les requêtes et visualiser les résultats dans des graphiques dans Google Sheets. Un module complémentaire existe sur le Chrome Store également mais je ne l'ai pas encore testé : OWOX BI BigQuery Reports.
- Développer une application (Web/mobile ...) via les APIs et librairies clientes mises à disposition afin de visualiser les données avec l'API Google Charts par exemple.

Voici quelques exemples du rendu que l'on peut avoir : Fig.17, 18 et 19.
Utiliser des outils tiers qui s'interconnectent très bien avec Big Query : Fig.20.

Fig.17



Source : http://bigquery-sensors.appspot.com/console

Conclusion

Je tenais à vous parler de Google BigQuery parce qu'il s'agit d'une techno que j'utilise au travail et qui ne fait pas assez parler d'elle à mon goût. Ce n'est pas une solution miracle, cela n'existe pas, mais elle peut répondre à un besoin donné. Le fait de notamment pouvoir interroger la base de données en SQL est vraiment très pratique.
En quelques secondes on peut exécuter sa requête sur la Web UI et obtenir un export en CSV.

Fig.18

Source : Un des outils de reporting et de BI d'atchikservices.

Fig.19

Source : Twitter for BigQuery.

Fig.20