3B-02

低更新頻度データを対象としたSQL-on-Hadoopエンジンの時空間効率の比較と評価

趙 漢哲†

NHN comico株式会社 データ研究室†

1. はじめに

Hadoopの登場は多量のデータの蓄積とそれを利用した高度な分析を可能にした。しかし、従来のデータ蓄積・分析方法をHadoop環境へ移行する場合、さまざまな問題に直面する可能性がある。弊社では日単位の重要業績評価指標(Daily KPI)分析をRDBMSからHadoop環境へ移行した時に複数の日に跨る重複データでストレージを効率よく利用できなかった経験がある。

本論文ではこのような重複データを効率よく格納するSQL-on-Hadoopエンジン(以下、SQLエンジン)を利用する解決策を提案する。この手法は既存のデータ取得・分析ロジックを変更する必要がないため、他の方法と比べて実装に掛かるコストが比較的に低い長所がある。

2. 重複データ問題と解決策

日単位のデータ分析を行う場合、毎日データのスナップショットを取って利用する方法と差分データだけを取得して使う方法がある。スナップショットを利用する場合、データの使い勝手がいい半面、保存するデータの量が大きくなる問題がある。差分データを取得する場合はデータサイズが小さくなる長所があるが、分析を行う度に差分データから元データを復元しなければいけない。

弊社では今まで前者の方法を使っており、更新頻度が低いマスター系データによる重複データ問題を抱えていた。マスター系データ（例、ユーザデータ、商品データなど）は1日単位ではほぼ変化がなくInsert、Update、Deleteの頻度が低い。長期間の時系列分析のため数年に渡りスナップショットを取り続ける場合、複数の日に跨って重複するデータが多量発生し、ストレージの効率的な利用ができない。

多くの解決策の中から、ここでは複数のSQLエンジンを比較し、重複データを効率よく格納するものを利用する方法を提案する。この方法では既存のデータ取得と分析ロジックを変更･検証する必要がないため適用に掛かるコストが大幅に節約できる。

次の章では現在弊社で利用するHive[1]と新しい候補であるHive on HBaseそしてPhoenix [2]を紹介する。

Evaluating Space-Time Efficiency of SQL-on-Hadoop Engines Regarding Data with Infrequent Updates

† Han-Cheol Cho

† Data Laboratory, NHN comico Corporation

3. SQLエンジンとその特徴

Hiveは初代SQLエンジンとして今でも幅広く使われており、弊社でもBI集計のメインツールとして使用している。Hiveでは日/週/月単位のように特定期間に関して集計を行う場合、最小単位（例、日単位）でパーティションを作ることが望ましい。それによって集計時に必要なデータだけを読み込むことが可能になり、処理速度が速くなる。しかし、パーティションによって分離されたデータは物理的に違うディレクトリーへ格納されるため、パーティションを持つテーブルのデータを横断的に圧縮することが出来ない。

Hive on HBaseはHiveのデータストレージレイヤーとしてHBase[3]を利用する。この場合、クエリをHBaseテーブルスナップショット上で実行することができるため、パーティションを作る必要がない。次の日のデータを書き込む前にHBaseテーブルのスナップショットを取ればいつでもその日のデータを復元･利用することが出来る。

Phoenixもデータストレージレイヤーとして HBaseを利用する。Phoenixが持つ一つの重要な特徴はHBaseのcell versioningを利用することができることだ。HBaseは特定のcellへアクセスするために{row, column, version}という三つのインデックスを利用する。Versionはデータに変更（Insert/Update/Delete）が起きた時間を示す。テーブル生成時に十分大きい最大保存バージョン数（MAX_VERSION）を指定し、Phoenix接続時にクエリ実行基準時間を指定すれば、その時間のデータを利用してクエリを実行することが出来る。パーティションによってデータが物理的に分離されないため同一データへの圧縮効率が高くなる。

4. 比較実験と結果

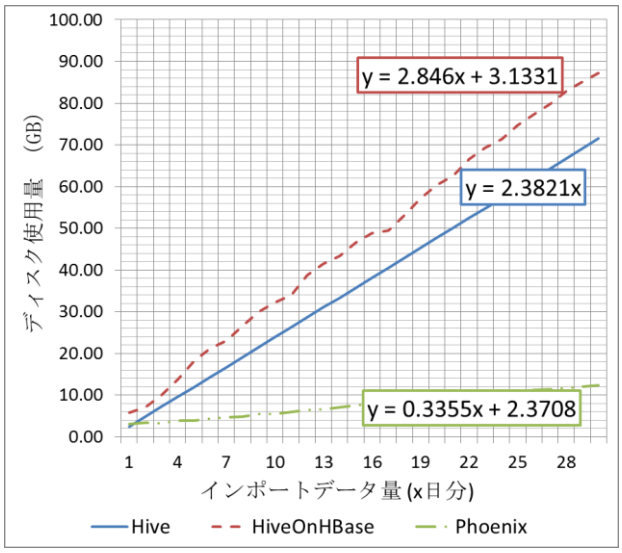
実験で使われたHadoopクラスタは5台のワーカーノードを持ち、それぞれ二つの6コア2.0GHzのCPU、64GBのメモリ、そして10台の2TBs HDDで構成されている。

実験データとしてはHortonworksのhiveベンチマークツール[4]を利用した。詳しくは、二種類のデータセットの中でTPC-Hデータセットを選択し、100GBサイズでテストデータを生成し、マスター系データであるcustomerテーブルを実験に利用した。このテーブルは8カラムで構成されており、総1500万レコードが格納されている。

4.1. ストレージ使用の空間効率

上で生成した customer テーブルを 1 日分のデータとみなし、同じデータを各 SQL エンジンへ 30 日分(３０回)インポートした。マスター系データではデータの Insert/Delete 頻度がとても低いが、Update はそれと比べて相対的に頻繁である（例、ユーザの最終ログイン日付など）。今回のテストデータの作成では Insert/Delete は無視し、Update だけを反映した（一つのカラムの最後尾へデータインポート日付をつける）。

図 1 は 30 日分のデータをインポートするときにどれぐらいのストレージが使用されるのかを示したグラフである。



〈図 1　SQL エンジンによる空間効率〉

1 日目のデータをインポートした時は Hive が約 2.4GB、Phoenix が 2.7GB、Hive on HBase が 6.0GB のストレージを使用している。この結果では同じ HBase をストレージレイヤーとして使用する Hive on HBase と Phoenix の間に大な差がみられる。Hive on HBase の場合、データインポート後に HBase テーブルスナップショットを取る段階があるが、その時にメモリ上にあるデータを全て HDFS へ書き込む。Phoenix の場合はこのステップがないため実際よりデータサイズが小さく見えている。HBase がメモリ上で管理するデータ(MemStore)を考慮すると約 2GB の容量を追加で使用していると判断される。

2 日目からの傾向を見ると、Hive は追加したデータ分データサイズが増えている。Phoenix の場合は、cell レベルで同じデータを纏めるため圧縮効率が高く、Hive と比べてとても少ない容量でデータの格納が可能だった。Hive on HBase は Hive と比べても多くの容量を使うが、これは HBase テーブルスナップショット機能の仕組みの問題である。HBase テーブルスナップショットはスナップショットを取った時点以前と以後のデータを分離して格納する。そのため違うスナップショットの間のデータを横断的に圧縮することが出来ない。

4.2. データインポート時の時間効率

HBase を利用する Phoenix や Hive on HBase は Hive と比べてランダムアクセスには有利だが、スループットが低い[5]。そのためデータインポート時間が長くなり、他のジョブに影響を及ぼす恐れがある。データインポート方法が違うため厳密な比較はできないが、参考のため、表 1 に実験で使った 1 日分のデータをインポートする時に掛かった時間(5 回分の平均)を示した。

〈表 1　データインポート時間〉

SQL エンジン名	時間(秒)
Hive	112s　(stdev: 12s)
Hive on HBase	1052s　(stdev: 127s)
Phoenix	665s　(stdev: 288s)

Hive とくらべて HBase を使っているエンジンはデータインポートに 5-8 倍の時間が掛かっていることが分かる。これは HBase の特性であり、解決のためには全データではなく差分データを取得するようにデータ取得ロジックを変更する必要がある。

5. まとめ

本論文では企業の BI システムを Hadoop 環境へ移行する時に直面することが可能な問題の中で、マスター系データによるストレージの非効率的な利用に関して述べた。そして重複データを効率的に格納する SQL エンジンである Phoenix を利用することで既存のデータ取得・分析方法を変更することなくこの問題が解決できることを示した。

提案手法を適用した場合、データ取得時の スループットが低下する問題があるが、これは HBase の特性でありデー タ取得ロジックを全データのインポートから差分データのインポートへ変更することで改善可能である。

参考文献

[1] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. 2009. Hive: a warehousing solution over a map-reduce framework. Proc. VLDB Endow. 2, 2 (August 2009), 1626-1629.
[2] Phoenix - Apache Software Foundation project home page (URL: https://phoenix.apache.org/)
[3] HBase - Apache Software Foundation project home page (URL: http://hadoop.apache.org/hbase)
[4] Hortonworks hive-testbench homepage (URL: https://github.com/hortonworks/hive-testbench)
[5] Performance comparision: Phoenix vs. related products (URL: https://phoenix.apache.org/performance.html)