読み出し性能と書き込み性能を両立させるクラウドストレージ

中 村 俊 介† 首 藤 一 幸†

1. はじめに

分散データストアの Read/Write の性質は, 従来の RDBMS などと同様に分散データストアにおいても, これを決めるのはおよそストレージエンジンである, という推測を立てた. もしそうであれば, ストレージエンジンを替えることで,write-optimized/read-optimized を切り替えることができ, 目的に応じてデータストア自体を使い分ける必要は本来無く, ストレージエンジンの差し替えで対応できる. そこで実証のために, ストレージエンジンを差し替え可能な分散データストア MyCassandra を開発し, その性能を評価した.

2. 研究背景

近年従来の RDBMS のスケールアウトの限界に伴い, NoSQL, Key/Value Store と呼ばれるスケーラブルな分散データベースが数多く現れているが, システムによってこれらの分散の仕組みやデータモデル, トランザクションなどの機能の有無など性質は大きく異なっている.

例えば Apache Cassandra[1] は Consistent Hashing という単一故障点の無い分散の仕組みを取り入れており, データの読み書きには書き込み側がシーケンシャルに行えるような SSTable を利用した差分ベースの書き込み方式を採用している. 一方,Yahoo PNUTS[2] や RDB の代表格である MySQL はデータの読み書きに従来のバッファープール方式をサポートしており, 1 回の I/O で最新レコードを取り出せるような読み出しに有利な設計がされている.

ところで, このような多くのクラウドシステムからクライアントのニーズに忠実に沿うものを選択するのは困難であり, 例えば Redis のようなスタックやキューが使えるデータモデルでかつ,Cassandra のような可用性の高いシステムを利用したい場合は利用者のアプリをそのシステムに無理やり適用を強いるか, 自身で新しいシステムを作ることになるのが実状である.

† 東京工業大学

そこで我々はデータストアを分散の仕組みとストレージエンジンに分離できることに着目し, 利用用途でストレージ部分を差し替えることで同じシステム内で読み書きの性能を調整できるような MyCassandra を実装した.

3. 実　　装

実装は下図のようにリクエスト処理と各ストレージエンジン間に Storage Engine Interface を設け, 設定ファイルに指定されたストレージに対してクエリが飛ぶようにした. Cassandra は Java で書かれている為,Java から各データベースにアクセスする為に JDBC を用いており, 各データベースごとにサポートされている JDBC コネクタを用いてインスタンス初期化と put/get メソッドを実装すれば容易に MyCassandra に取り込むことが可能である.

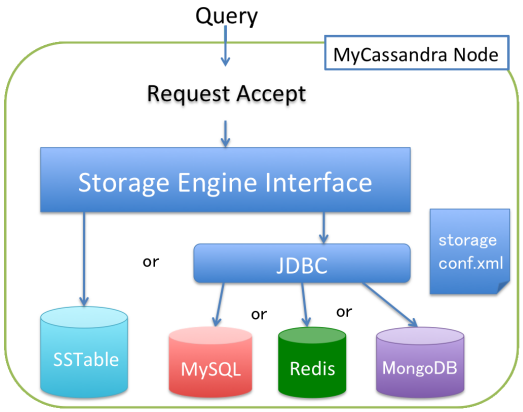

図 1 MyCassandra ノード

4. ベンチマーク

SSTable, MySQL (InnoDB), Redis を対象に分散データストア用ベンチマーク YCSB[3] を用いて評価した. YCSB はまずデータストアにデータをロードする. 続いてある程度の数のクエリでウォームアップを行う. そして, クライアント数を増やすことで目標ス

ループットを段階的に上げていき, その時のレイテンシを計測する.
Read:Write 比率が異なるワークロードとして Write-Only[100:0],Write-Heavy[50:50], Read-Heavy[95:5], Read-Only[100:0] を用意しそれぞれ計測した.
実験パラメータを以下に示す.

- データノード数: 6 台
- YCSB クライアント数: 1 台
- ロードレコード数: 2,400 万件
- レコード: 10 カラムの 1KB ASCII 文字列
- ウォームアップ命令数: 5 万件

5. 評　　価

図 2 にスループット 5,000 qps 時の Read/Write レイテンシを示す. 書き込みレイテンシは SSTable の方が小さく,MySQL の最大 41.4%であり, 読み出しレイテンシは MySQL の方が速く SSTable の最大 49.4%であるという結果が得られた.
図 3 にクライアントスレッド数を 40 として負荷をかけたときの各 Workload の qps を示す. 書き込みが多い Workload では.SSTable の方が高く MySQL の最大 5.32 倍であり, 読み込みが多い Workload で MySQL の方が高く SSTable の 2.35 倍という結果が得られた. この通り、データストアを置き換えずともストレージエンジンを差し替えることで性能の傾向が異なるデータストアを得ることができた.

Redis は全データをオンメモリで扱う為にどの操作においても高速に行うことができる. しかし, 揮発性しても良いデータしか置けず, メモリサイズからくる保存データ量の制限もある.

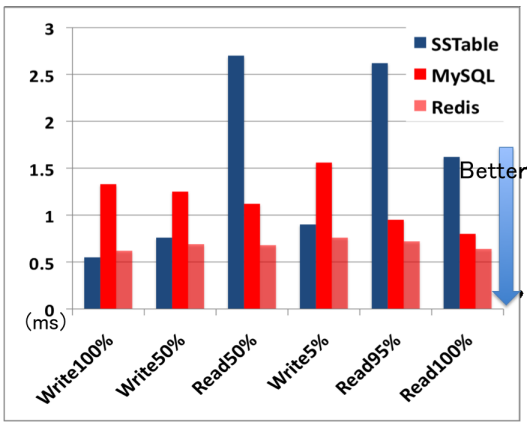

図 2　Workload ごとのレイテンシ

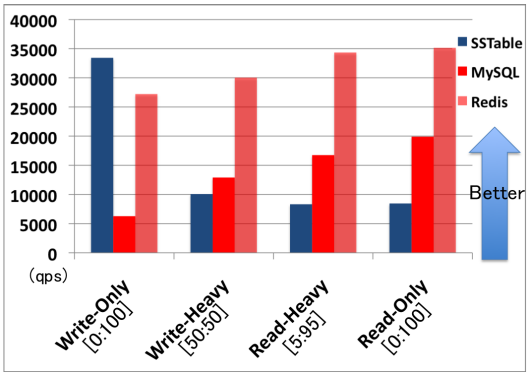

図 3　Workload ごとの qps

6. 今　　後

今後はストレージエンジンの異なるノードを組み合わせてクラスタを構成し, 単一のクラスタで, 読み出しと書き込み性能を両立させる. MyCassandra はノード情報を周囲ノードと交換するので, その際にストレージタイプを織り込むことで実現できる. このときレプリカ間の一貫性やネットワーク近接性との両立が課題となる.

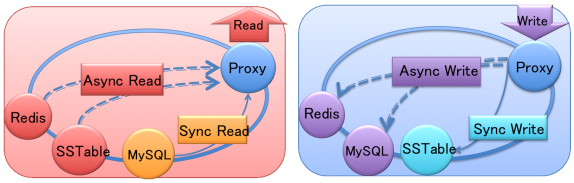

図 4　MyCassandra クラスタ

参　考　文　献

1) Avinash Lakshman,Prashant Malik. Cassandra - A Decentralized Structured Storage System. LADIS '09, 2009.
2) Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-arno Jacobsen, Nick Puz, Daniel Weaver and Ramana Yerneni. PNUTS: Yahoo! 's Hosted Data Serving Platform. VLDB '08, 2008.
3) Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears. Benchmarking Cloud Serving Systems with YCSB. SoCC '10, 2010.