4K-1 Cassandraによる局所性を考慮した分散並列処理の提案と実装

菱沼 直子† 竹房 あつ子‡ 中田 秀基‡ 小口 正人†
†お茶の水女子大学 ‡産業技術総合研究所

1. はじめに

クラウド技術の普及により，個人が生み出す情報が大量にネットワーク上に保存されるようになり，そのデータ量が爆発的に増加している．その結果，大量のデータを高速に処理することが必要となり，分散環境で一貫性の制限を緩和して処理性能を重視する分散KVS(Key Value Store)と呼ばれるNoSQL型データベース管理システムが注目され始めた．分散KVSでは，RDBMSでは管理が困難な規模の大容量データを分散環境で管理することができる．

管理されている大容量データを効率よく活用するためには，対象となるデータに対して複数計算機を利用して並列処理を行う必要がある．しかしながら，分散KVSからデータを取り出した後に再度データを複数計算機に分散させて並列処理を行うと，データの転送オーバヘッドが非常に大きくなってしまう．

そこで本研究では分散KVSの実装のひとつであるApache Cassandra[1](以下Cassandra)に着目し，大規模データをより高速に処理するための手法を提案する．本稿では，Cassandra上に保存された異なる値に対し，事前に指定された処理を並列に実行できる機能を追加し，その性能を評価する．

2. 提案手法/設計

Cassandraに保存されたデータに対して任意の処理を行うには，Cassandraから対象となる値を取得し，その後任意の処理を行うのが通常である．しかし，Cassandraの読み出し処理性能はあまり高くなくない上に，対象とする値のデータ量が多いと値を取得するための通信処理が遅くなってしまうことが考えられる．

そこで本提案手法では，Cassandra上の値に対し任意の処理を効率よく行えるようにするために，まずUDF(User Defind Function)と類似した機能をCassandraに追加する．この機能を用いてユーザが実行したい処理をプラグインとして定義し，定義された処理を各データノード上で実行して処理結果のみをクライアントに返す．これにより通信データ量を抑えることができる．また，処理対象の値を複数指定することで異なる値に対して並列処理が可能になり，より高速に処理が行えると考えられる．

本提案手法の概要を図1に示す．

(1) クライアントから各データノードへリクエストを送信．

(2) 各データノード上にある，異なる値A,Bに対してプラグインとして定義された処理を並列実行し，その結果を新たな値A',B'として保存．各値のレプリカに対しても同様の処理を行う．図1では値Bのレプリカは省略．

(3) 処理を行った結果を格納した値A',B'をリクエストの答えとして返す．

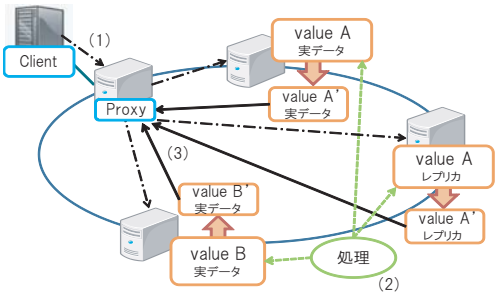

図1: 提案手法の概要

本稿ではCassandraに保存された複数の異なる値に対し，事前に指定した処理を各データノード上で実行し，処理結果のみをリクエストの答えとして返す機能を実装した．事前に指定される処理は，引数にパス名を指定することにより，各実行時に指定された任意のLinuxコマンドを対象データを管理しているデータノード上で実行可能となっている．

3. 評価

本研究では，Cassandraに保存された値に対し処理を行う際に，Cassandraを通常利用した場合と，前章で説明した実装を用いた場合の性能比較を行う．

3.1 実験環境

ノード数3台，8台からなるクラスタに，機能拡張を行ったCassandraをインストールした．今回の開発では，Cassandraバージョン1.1.0を用いた．測定に用いたノードの性能を表1に示す．

表1: machine spec

OS	Linux 2.6.32-5-amd64 Debian GNU/Linux 6.0.4
CPU	Intel(R) Xeon(R) CPU @ 2.66GHz x4 Intel(R) Xeon(R) CPU @ 3.10GHz x4
Memory	8GByte

3.2 測定概要

Cassandraによるクラスタを構築し，Cassandraの標準コマンドmultiget[2]を使用し複数の値を取得してからワードカウントコマンド(wc)を実行した場合以下（クライアント側処理）と，今回実装した，複数の値に対し各データノード上でwcを実行してその結果のみをクライアントに返す機能を使用した場合（以下サーバ側処理）の，クラスタ参加ノード数，処理対象の値の数の変化に伴う実行時間の変化を比較した．図2,3に処理対象の値の数を2とした際のそれぞれの処理の流れを示す．

クライアント側処理の流れ

Toward and implementation Data Locality-aware distributed parallel processing on Cassandra
† Naoko Hishinuma, Masato Oguchi
‡ Atsuko Takefusa,Hidemoto Nakada
Ochanomizu University (†)
National Institute of Advanced Industrial Science and Technology (AIST)(‡)

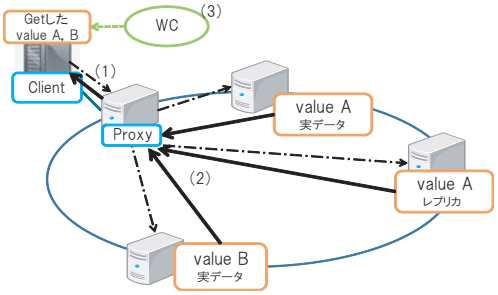

図 2: クライアント側処理の流れ

(1) 読み出しリクエストをクライアントから送信.

(2) 担当するデータノードはリクエストに対する答えとして値 A，B をプロキシに返し，プロキシはその値をクライアントに返す.

(3) 読み出した値 A，B に対して wc を実行する.

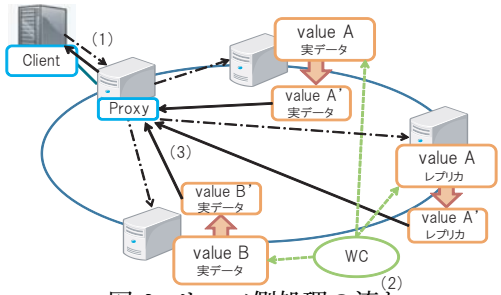

図 3: サーバ側処理の流れ

サーバ側処理の流れ

(1) 読み出しリクエストをクライアントから送信.

(2) 担当するデータノードはリクエストに適する値 A，B に対して wc を実行し，処理結果を新たな値 A'，B' とする.

(3) データノードは値 A'，B' をプロキシに返し，プロキシはその値をクライアントに返す.

クライアント側処理，サーバ側処理どちらも (1)〜(3) を一つの処理とする．ノード数が 3 台，8 台からなるクラスタを用い，Cassandra のレプリケーション数は 3（デフォルトは 1[オリジナルデータのみ]），一貫性レベルは ALL で固定し，処理対象の値の数を 5〜30 まで変化させ，実行時間を測定した.

3.3 測定結果/評価

図 4 にノード数が 3 台，8 台からなるクラスタを用いて処理対象の値の数を 5〜30 まで変化させた際の実行時間の変化を示す．縦軸が実行時間 (sec) で，横軸が処理対象の値の数となっている．まず，ノード数が 8 台からなるクラスタを用いて測定を行った結果（server，client 側処理__8 台）に着目すると，サーバ側処理の方がクライアント側処理に対し実行時間が半分以下になっていることがわかる．これは，クライアント側処理では wc を 1 台のクライアントノードで実行しているの対し，本提案手法であるサーバ側処理では，値が保存されている各データノード上で処理を行うため，wc が並列分散実行されているためである．また，本提案手法では wc の結果のみをリクエストの答えとして返しているため，通信データ量を削減できていることも実行時間に影響を与えている．以上より，本提案手法は Cassandra 上に保存されている複数の値の対し処理を行う場合に有効であることが示せた.

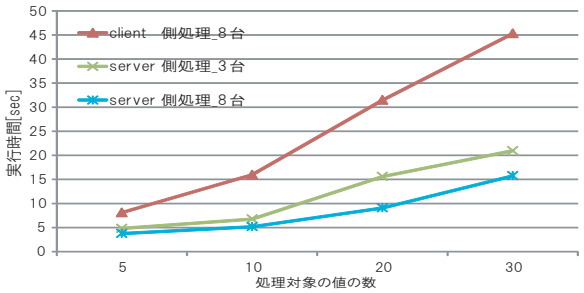

図 4: サーバ側処理とクライアント側処理の実行条件を変化させた際の実行時間

次に，サーバ側処理をノード数が 3 台，8 台からなるクラスタを用いて測定を行った結果（server 側処理__ 3 台，8 台）に着目すると，ノード数が 3 台からなるクラスタを用いた場合の方が処理対象の値の数に関わらず実行時間が長くなっていることが確認できる．これは，今回の測定がレプリケーション数 3，一貫性レベル ALL に固定し実行しているため，3 台からなるクラスタを用いた際にはすべてのノードで処理が行われ，8 台からなるクラスタを用いた際には処理が分散するのでこのような結果が出たと考えられる.

4. おわりに

分散 KVS の実装の 1 つである Apache Cassandra に着目し，データの局所性を考慮した分散処理を提案した．本稿ではユーザが指定した複数の異なる値に対し，その値が保存されている各データノード上でユーザ指定した処理を実行し，処理結果をカラムの新たな値としてクライアントに返すサーバ側処理機能を実装した.

実装した機能の処理時間を評価したところ，クライアント側処理に対し提案手法であるサーバ側処理では，処理対象の値の数に関わらず実行時間が半分以下に抑えられていた．また，クラスタ参加ノード数を変化させて測定したところ，ノード数が多い方が実行時間が短くなる傾向が見られた．以上より処理を分散並列実行させている本手案手法の有効性が示せた.

今後の課題としては，Cassandra 上に保存されている値が依存関係にあった際の処理追加すること，UDF と類似した機能を追加することがあげられる.

参考文献

[1] Avinash Lakshman, Prashant Malik, "Cassandra - A Decentralized Structured Storage System," The 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware, October 2009.

[2] Eben Hewitt(著), 大谷晋平, 小林隆 (訳):Cassandra, オライリー・ジャパン,2011