様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ４ 月 ２２ 日現在

機関番号：１３９０３
研究種目：基盤研究(C)
研究期間：2012～2014
課題番号：２４５００１１３

研究課題名（和文）スループット性能向上を目的とした範囲検索可能分散キーバリューストア

研究課題名（英文）Improving Response Time for Distributed Key Value Store with Query Scheduling

研究代表者
松尾　啓志（Matsuo, Hiroshi）

名古屋工業大学・工学（系）研究科（研究院）・教授

研究者番号：００２１９３９６

交付決定額（研究期間全体）：（直接経費）　　4,000,000円

研究成果の概要（和文）：分散KVSはデータをKeyとValueという単純な構造で管理するため，容易にスケールアウト可能であるという利点がある。分散KVSの実装によっては，Keyの範囲を指定することによりそのKey間に属するValueを取得する検索処理が可能な実装もある．しかし，単一検索と範囲検索が混在する環境においては，高速に実行終了が可能な単一検索命令の実行が待たされてしまい，結果として平均応答時間が増加するという問題がある．そこで本研究では複数の検索クエリーをスケジューリングすることにより，データを要求するクエリーの平均応答時間を短縮させる手法を提案し評価を行なった結果，平均応答時間の短縮を確認した．

研究成果の概要（英文）：Distributed Key Value Store(KVS) is a datastore which manages data across multiple machines. Since distributed KVSs manage data which consists of simple key-value pair, they can achieve scalability easily.
Distributed KVSs are widely used in many services managing large-scale data.
Distributed KVSs provide interfaces to access key-value pair by simply specifying the key. We refer to a query which only obtains a value from a key as a single query. Some distributed KVSs support a range query which obtains multiple values from a key range. However, under mixed query workloads that consist of single and range queries, single queries(which can be executed faster) are forced to wait until preceding range queries are finished. And this results in the increase of average response time. We propose an approach to reduce the average response time by query scheduling. We implemented our method on Cassandra, and evaluation results showed a reduction of the average response time.

研究分野：計算機システム

キーワード：分散処理　分散ＫＶＳ

１．研究開始当初の背景

Facebook や Amazon に代表されるように、今や１０億人が同時に１つのデータベースにアクセスする時代が到来した。このような大量アクセス環境では、従来の高度な検索性能を有するリレーショナル DB では対応できず、新しく NoSQL と呼ばれる概念が提案され、その実装の代表例が分散キーバリューストア(Distributed Key Value Store:以下 D-KVS)である。 図１に D-KVS での検索の流れを示す。D-KVS は、Key を指定し、その Key に紐付いた値 Value を返答する単純な機能しか有しない。

D-KVS の導入が進むにつれ、キーの範囲を指定した検索要求が高まっている。しかし、大部分の D-KVS の実装はキーをハッシュ化するため、範囲検索時にはすべてのノードに検索要求が行われ、事実上実装は不可能である。ハッシュ化を行わない場合、それぞれのノードに格納される対象 Key は、距離的に近い Key 群となり、従って範囲検索の場合でも、必要最小限のノードに対してのみ検索要求を行うことが可能となる。しかし、そもそものハッシュ関数導入の目的の１つである負荷分散ができず、特定のノードに対して負荷が集中することが避けられない。D-KVS の実装の一つであるカーネギーメロン大学で開発された Mercury は、ノードの守備範囲を動的に増減させることにより、負荷分散を行う手法を導入した。しかし、ノードの守備範囲を動的に変更する場合、変更時に D-KVS のサービスを局所的ではあるが、停止する必要があり、特にノードが高負荷時の場合、排他制御が高コストになることが知られている。

我々は、従来から範囲検索と複数属性の処理に対応した MerDy を開発している。この MerDy は、検索を行うためのノード群と、データを保存するノード群をそれぞれ別の特性を持つ D-KVS で構成する。さらに検索要求を、最初にプロキシ層で受け取り、２つの D-KVS に分配することにより、負荷分散機能を付加した範囲検索を行うことに成功した。しかし、プロキシ層の追加により、オーバヘッドが増大し、スループットの低下を招いた。この開発の過程で、範囲検索を行うパケット群と、単一検索を行うパケットを各ノードの処理待ち行列で適切なスケジューリングすることにより、Mercury のような守備範囲変更を行わずに D-KVS のスループット性能を向上させる可能性に気がついた。つまり複数 PC からの単一検索と、範囲検索の要求パケットが大量に到着する環境においては、複数のパケットに分割された範囲検索パケットのうち、一番処理の遅いパケットの処理終了時刻を目標にしたデッドラインスケジューリングを行うことにより、結果として単一検索のパケットの処理が優先され、データベースシステム全体のスループットが向上すると考えられる。我々は、すでに Facebook 社が公開している D-KVS Cassandra 内の改造を行う初期実験により、スループットの向上を確認している。

しかし、本格的なスケジューリングを行うためには、多数のノード間での複雑かつ動的な環境下でのスケジューリング最適化問題を解く必要が生じる。さらに集中アルゴリズムの適用は、それ自体が並列性能のボトルネックとなるため、スケジューリングアルゴリズムの分散アルゴリズム化も重要な課題となる。

２．研究の目的

本研究の目的は、スケールアウト性能を有する範囲検索可能 D-KVS を作成することである。その重要なマイルストーンが、分散スケジューリングアルゴリズムの開発である。しかし、動的環境下であり、しかもスケジューリング対象のノード数も 100 から 1000 を想定した分散アルゴリズムは、まだ開発されていない。この研究に於いての理論的な面での課題はまさにこの一点であると言っても過言ではない。

さらに、我々はすでに D-KVS を対象としたオブストラクションフリートランザクションの実装を終えている。従って単に論文として成果を発表するだけではなく、業務で運用可能な、D-KVS の開発を行うことを最終目標とする。

３．研究の方法

本研究は、以下の３つのプロジェクトに分割して遂行する。

プロジェクト１：分散制約最適化による協調スケジューリング

従来から、たくさんのパラメータや環境が存在する、複数 CPU コアやグリッドコンピューティングムにおけるスケジューリング問題は、経験的な知見により行われる場合が多い。このプロジェクトでは、ネットワーク遅延が存在する 100～1000 台程度の多数ノードに大量の情報検索パケットが到着する待ち行列内の協調スケジューリング環境を想定しており、従来からの経験的な知見ではなく、理論的な枠組みを構築することを目的とする。

さらに今回の想定する環境のように動的に変化する環境においても、分散制約最適化を行うことができる階層化分散制約最適化手法の提案を行った。従って我々の研究グループでは本プロジェクトを遂行する上での基礎的な手法はすでに行われている。本プロジェクトを遂行するには、(1)スケジューリングアルゴリズムの最適化、(2)分散制約最適化の実問題への適用の２つの問題が考えられる。しかし、システム実装に関しても多数の経験があり、さらに分散制約最適化に関しても理論的な面から様々な手法を提案している松尾、松井が協力して全力で立ち向かうことにより、この問題を解決可能であると

考える。

プロジェクト２： Cassandra を用いた実装

このプロジェクトでは、Facebook 社がオープンソースとして公開している分散キーバリューストアである Cassandra を基本に、パケットキュー部分の改造を行うとともに、ノード間の通信には、初期段階では標準で実装されているゴシッププロトコルを考えているが、さらに高速なメッセージ交換手法も実装する予定である。

プロジェクト３：実フィールドによる新しいアプリケーション開発

このプロジェクトでは、本研究で開発する D-KVS を実際のフィールドで運用することにより、さらなる改良を図ることを目的とする。従って研究期間の後半で行うプロジェクトである。本学情報基盤センターでは、IC カード出席システムなどの様々なシステムを自主開発するとともに、運用を行っている。そのデータをマイニングすることにより、学習履歴と最終成績との関係からの学習改善など、複数の業績を上げている。本プロジェクトにおいても、開発した D-KVS を用いて様々なフィールドワークを行う予定である。

４．研究成果

4.1 提案手法

本研究の提案手法を説明する．本研究では，Cassandra の検索処理の応答性能を改善するために，以下の 3 つの手 法を提案する。

・ 単一検索の優先実行
・ 範囲検索の並列要求
・ 範囲検索同士のスケジューリング 以降で各手法について詳細を説明する．

(a)単一検索の優先実行

単一検索と範囲検索のリクエストのスケジューリングに より，単一検索の応答時間を短縮する．このスケジューリ ングでは高速に処理可能な単一検索の優先度を上げ，低速 な範囲検索の優先度を下げることで，単一検索を範囲検索 よりも優先して実行する．具体的には単一検索リクエスト 命令が各ノードのリクエストキューに投入される場合に， 他のすべての範囲検索リクエスト命令よりも前方に単一検 索リクエスト命令を投入する．本スケジューリングにより，高速な処理が低速な処理に待たされることによる応答時間の増加を抑制する．

(b) 範囲検索の並列要求

範囲検索の並列要求により，範囲検索の応答時間を短縮する．オリジナルの Cassandra の範囲検索の実装では，検索範囲が複数のノードにまたがる場合に，各ノードへのリ クエストを逐次的に送信する．この各ノードへのリクエス トの送信を，検索範囲がまたがるノードに一度に送信することで，各ノードでの検索処理を並列に実行し，範囲検索の応答時間を短縮する．

(c) 範囲検索同士のスケジューリング

範囲検索同士のスケジューリングにより，範囲検索の応 答時間を短縮する．検索範囲が狭い範囲検索の優先度を上 げ，検索範囲の広い範囲検索の優先度を下げることで、より少ないノードからの検索結果で要求を満たすことのできるクエリーの優先度を上げる．検索範囲の広さは，範囲検索の並列要求で一度に送信したノード数，つまり並列に検索処理を実行しているノード数を基準とする．

4.2 検索範囲のみでの優先度付けの問題点

単純に検索範囲のみで優先度を設定した場合，範囲検索 の平均応答時間が増加する場 合 が あ る ． 例 え ば Client1 が A-F,G-L,M-R を管理する 3 ノードにまたがるデータを要求 する範囲検索クエリーを送信し，次に Client2 が M-R,S-Z を管理する 2 ノードにまたがるデータを要求する範囲検索 クエリーを送信した場合を考える．この時，S-Z を管理する ノードが高負荷である場合，各ノードのリクエストキューは図 4 のように，A-F,G-L のキューに 1 つずつ，M-R の キューに 2 つ，S-Z に多数のリクエストが挿入されている 状態になる．

この場合，検索範囲による優先度を考慮して Client2 が送信した M-R,S-Z を管理する 2 ノードにまたがるデータを要求する範囲検索クエリーを Client1 が送信したクエリーよりも優先すると，他のノードからの複数の要 求がリクエストキューに含まれる S-Z を担当するノードのために Client2 の応答時間は短縮されず，Client1 の応答時間は増加する．結果として Client1 と Client2 の範囲検索 クエリーの平均応答時間は増加する．

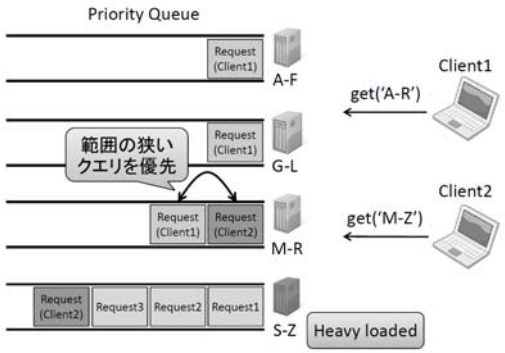

図 1 検索範囲のみの優先度付けで応答性能が低下する場合

検索範囲のみでの優先度設定による平均応答時間の増加 を抑制するためには，各ノードが他のノードの負荷を考慮 してスケジューリングする必要がある．すなわち，多くのリクエストがキューに溜まっているような負荷の高いノードを範囲に含むクエリーの優先度を下げることで，平均応答時間の増

加を抑制することができると考えられる.

4.3 実装

Cassandra は Stage Event Driven Architecture (SEDA)を実装している . SEDA はアプリケーションを, 複数の Stage と呼ばれる単位に分割して管理する. 各 Stage は Event Queue, Thread Pool, Event Handler で構成され, 各 Stage で Event を受け渡しながら処理を進める. 例えば検索処理の場合, 検索 リクエスト命令を Event Queue に投入する. Thread Pool は Event Queue 上のリクエスト命令を取り出し, Event Handler によって実行する. このように Event の実行とリ クエストの受信処理を分離することで, リクエスト数が増 えた場合でも, スレッド数が増加してスループットが落ちることはない.

本提案手法では, 検索処理を管理する Stage である READ Stage の Event Queue に優先度付きキューを利用すること により, 検索処理のスケジューリングを実装した. 優先度 は, 自ノードからの単一検索リクエスト命令, 他ノードか らの単一検索リクエスト命令, 範囲検索リクエスト命令の 順に高く設定した.

5. 評価

複数クライアントによる単一検索と範囲検索のクエリー の送信実験により, 提案手法を実装した Cassandra とオリジナルの Cassandra との性能比較を行った.

5.1 評価環境

本評価実験では Cassandra バージョン 1.1.2 を用い, 12 台のノードからなる Cassandra クラスタを構築した.

5.2 実験内容

検索クエリーを送信する前に, あらかじめ 1 件あたり約 1KB のデータ 10,000,000 件を Cassandra ノードに挿入した. 各データは Key に応じて 1,000 個の ColumnFamily に分けて格納している. Cassandra の範囲検索では, ColumnFamily ごとにデータを検索する.

評価対象は, 以下の 6 つである.

1) (O)オリジナル
2) (ReS)単一検索の優先実行のみを実装
3) (P)範囲検索の並列要求のみを実装
4) (ReS+P) 単一検索の優先実行と範囲検索の並列要求を実装
5) (P+RaS) 範囲検索の並列要求と範囲検索の優先度付けを実装
6) (A) 単一検索の優先実行と範囲検索の並列要求, 範囲検索の優先度付けを実装

評価をそれぞれ各 10 回ずつ行い, 最もスループットの良いものを採用した.

5.3 評価結果

単一検索と範囲検索を 9:1 の割合で送信した場合の単一検索の平均応答時間を図 2, 範囲検索の平均応答時間を図 3 に, 単一検索と範囲検索を 5:5 の割合で送信した場合の単一検索の平均応答時間を図 4, 範囲検索の平均応答時間を図 5 に示す.

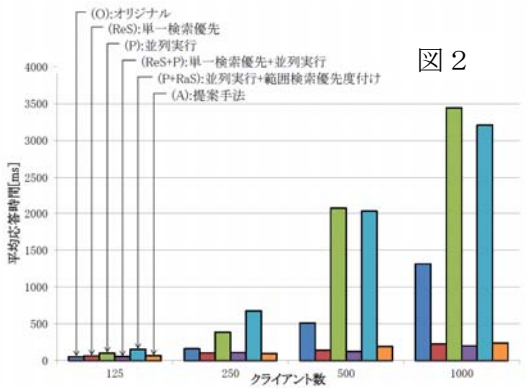

図 2

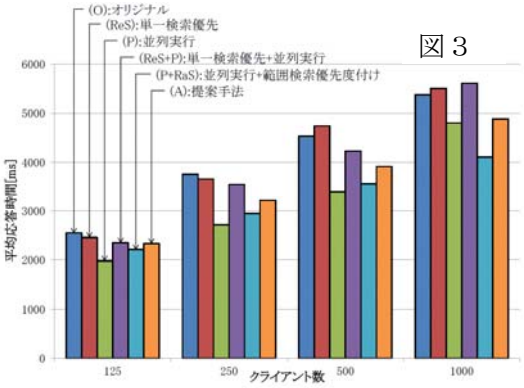

図 3

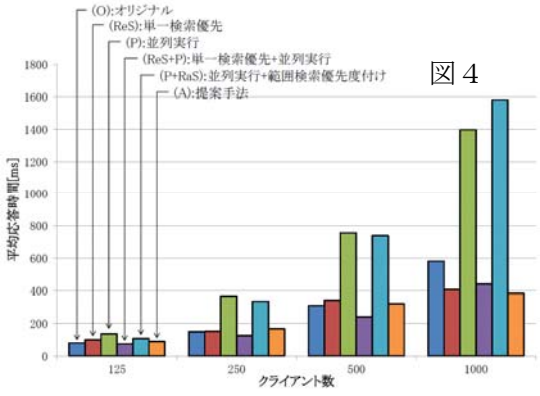

図 4

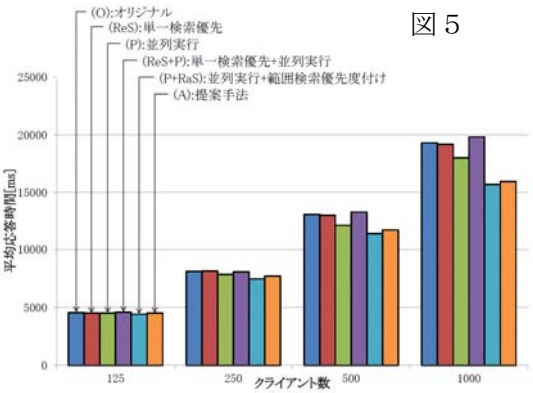

図 5

6.まとめ

分散 Key Value Store での検索の応答性能を改善するために，検索処理のスケジューリングと範囲検索の並列要求を提案した．本研究では単一検索と範囲検索という 2 種類の検索クエリーが混在する状況を想定し，それらのクエリーの優先度を付けることによりスケジューリングを実現した．

提案手法を分散 Key Value Store の実装の一つである Cassandra 上に実装し，ベンチマークプログラム YCSB により評価した．評価の結果，オリジナルの Cassandra では，クライアント数の増加に伴い単一検索の応答時間が大 幅に悪化したのに対し，提案手法を実装した Cassandra では，これを改善することを確認した．また範囲検索についても，並列要求とスケジューリングの効果により応答時間が改善されることを確認した．

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計６件）

1) Shiori Kurazumi, Tomoaki Tsumura, Shoichi Saito, Hiroshi Matsuo:” Dynamic Processing Slots Scheduling for I/O Intensive Jobs of Hadoop MapReduce”, 2012 Third International Conference on Networking and Computing (ICNC 2012),pp.288-292 (2012)
2) Kazuki Yamazaki, Ryota Kawashima, Shoichi Saito, Hiroshi Matsuo:” Implementation and Evaluation of The JobTracker Initiative Task Scheduling on Hadoop”, The First International Symposium on Computing and Networking (CANDAR'13), pp.622-626 (2013)
3) Masaya Matsuno, Ryota Kawashima, Shoichi Saito, Hiroshi Matsuo:” Reducing the load of Metadata Server by changing cache policy dynamically in Distributed File System”, The First International Symposium on Computing and Networking (CANDAR'13), pp.173-179 (2013)
4) Satoshi Fukuda, Ryota Kawashima, Shoichi Saito, Hiroshi Matsuo:” Improving Response Time for Cassandra with Query Scheduling”, The First International Symposium on Computing and Networking (CANDAR'13), pp.128-133 (2013)
5) 福田 諭, 川島 龍太, 齋藤 彰一, 松尾 啓志:” 検索範囲を考慮したクエリースケジューリングによるCassandraの応答性能向上”, 情報処理学会論文誌, vol.56, no.2, pp.492-502 (2015)
6) 中居新太郎, 川島 龍太, 齋藤 彰一, 松尾 啓志:” 更新履歴に基づいたメモリページ転送順序スケジューリングによる仮想マシンライブマイグレーションの高速化”, 情報処理学会論文誌, vol.56, no.2, pp.516-524 (2015)

〔学会発表〕（計７件）

1) 福田 諭, 津邑 公暁, 齋藤 彰一, 松尾 啓志:” クエリースケジューリングによる分散キーバリューストアの応答性能向上”, 情処研報 (SWoPP2013), vol.2013-HPC-140, no.21, pp.1-7 (2013)
2) 松野 雅也, 津邑 公暁, 齋藤 彰一, 松尾 啓志:“キャッシュポリシの動的変更による分散ファイルシステムのメタデータサーバー負荷低減手法”, 情処研報 (SWoPP2013), vol.2013-OS-126, no.3, pp.1-9 (2013)
3) 山崎 一樹, 津邑 公暁, 齋藤 彰一, 松尾 啓志:” Hadoop における JobTracker 主導型タスクスケジューリングの実装と評価”, 信学技報 (SWoPP2013), vol.113, no.169, pp.85-90 (2013)
4) 中居新太郎, 川島 龍太, 松尾 啓志:” アクセスタイミングを考慮したメモリページ転送順序スケジューリングによる仮想マシンライブマイグレーションの高速化”, 信学技報 (CPSY2013), vol.113, no.417, pp.61-66 (2014)
5) 佐藤 貫大, 川島 龍太, 松尾 啓志:” MongoDB におけるデータの自動バランシング機能の優先度付き Chunk による性能改善”, 信学技報 (CPSY2013), vol.113, no.417, pp.55-60 (2014)
6) 田中 俊也, 福田 諭, 川島 龍太, 齋藤 彰一, 松尾 啓志:” 信学技報 (SWoPP2014), vol.114, no.155, pp.173-178 (2014)
7) 松野 雅也, 太田 篤, 川島 龍太, 松尾 啓志,” 複数の一貫性レベルを保証可能なバックエンドベースレプリケーション”, 情処研報, vol.2015-OS-132, no.14, pp.1-11 (2015)

６．研究組織

(1)研究代表者

松尾啓志 (MATSUO HIROSHI)

名古屋工業大学・大学院工学研究科・教授

研究者番号：00219396

(3)連携研究者

松井 俊浩 (MATSUI TOSHIHIRO)

名古屋工業大学・大学院工学研究科・准教授

研究者番号：60437093

津邑 公暁 (TSUMURA TOMOAKI)

名古屋工業大学・大学院工学研究科・准教授

研究者番号：00335233

齋藤彰一　(SAITO SHOICHI)
名古屋工業大学・大学院工学研究科・准教授
研究者番号：70304186