3N-4

KVSにおける仮想ノードを用いた動的ノード追加の性能伸張性に関する一考察

御代川 翔平† 徳田 大輝† 山口 実靖†

工学院大学 工学部 情報通信工学科†

1. はじめに

クラウドコンピューティングの普及によりデータベース管理システムのスケーラビリティの確保が重要視されている．その解決策の一つに Key-Value Store (KVS)がある．KVS は Key と Value のみで構成されるシンプルなデータ構造のデータベースであり，高いスケーラビリティがある．また，実行時にノード数の増減を行うことが可能であるため，高負荷環境ではノード数を増やし，負荷が低くなればノード数を減らすといった動的な性能の伸縮を行うことができる．

本研究では, 実行中の KVS システムにノードを追加して動的に性能を伸張させることに着目し，動的性能伸張性の評価と考察を行う．

2. Key-Value Store

KVS とは，Key を指定することで Value を得ることができるデータベース管理ソフトウェアである．読み書きの機能を簡単なものにすることによって，高いスケーラビリティを得ることができる．KVS の一つに Cassandra[1]がある．

2.1. Cassandra

Cassandra は Facebook 社が開発した KVS であり，現在は Apache Software Foundation のトップレベルプロジェクトである．Amazon 社の Dynamo[2]の分散ハッシュテーブルと Google 社の BigTable[3]のデータモデルを併せ持ち，結果整合性の一貫性を持つ．

Cassandra を構成する各ノードはトークンと呼ばれるハッシュ値を持ち，リング状のハッシュ空間にトークンをもとに配置される．リング上の各ノードは，ハッシュ値が自身のトークン値以下でかつ直前ノードのトークン値より大きい範囲を担当する．読み込みまたは書き込みをする際には Key をハッシュ関数にかけ，そのハッシュ値から担当ノードを特定し読み込み，書き込みを実行する．また，稼働中のシステムに新規ノードを追加した場合，新規ノードはトークン値から担当範囲が決まり，その範囲を担当している稼働ノードから担当範囲分のデータを取得する．

Cassandra ではデータベースの複製(レプリカ)の数を指定することが可能である．レプリカ数は初期設定では 1 であるが，複数の値にすることによって耐障害性を向上させることができる．レプリカは担当ノードの後続ノードに配置される．

Cassandra 1.2 から仮想ノード機能が追加されている．これは 1 つのノードに複数のランダムなトークン値を持たせることにより，担当範囲を複数に分割する機能である．

3. ノード追加時間の評価

Cassandra は，ノードを追加することによって性能を拡張することができる．本章では，Cassandra システムにおける拡張性能を，仮想ノードを使用しなかった場合(以下，非仮想)と使用した場合(以下，仮想)で評価を行う．

評価環境として既存ノードの PC3 台，クライアント PC 1 台，追加ノード用の PC を 5 台用意した．読込負荷にベンチマークソフトの YCSB(Yahoo Cloud Serving Benchmark)[4]を使用した．データベースはレコード数 1600 万件(約 17[GB]), レプリカ数 3 で測定した．

測定内容は非仮想と仮想, 50[ops/sec]の読込負荷ありまたはなしを組み合わせた 4 通りを同時追加と順次追加で追加に要する時間を測定した．読込負荷の一貫性レベルは ONE である．これは，クライアント PC から稼働ノードに読込要求を送り，一つでも返答があれば読み込み完了とするものである．同時追加とは，稼働中のノードに複数の追加ノードを一斉に追加をすることであり，順次追加は，1 台追加が完了したらまた 1 台追加することを繰り返すことを表している．

3.1. 同時追加の追加時間

同時追加の実験結果を図 1 に示す．図 1 より同時追加では追加台数が 3 台以上から仮想が非仮想よりも追加時間が短いことがわかった．また，読込負荷による影響は見られないことがわかった．

3.2. 順次追加の追加時間

順次追加の実験結果を図 2 に示す．図 2 から順次追加では非仮想が仮想よりも追加時間が短いことがわかり，追加台数が増えるにつれて追加が完了するまでにかかる時間の差は大きくなることがわかった．また順次追加でも読込負荷の影響をほぼ受けないことがわかる．

A Study on performance extensibility of the dynamic node addition using a virtual node in KVS

Shohei Miyokawa†, Taiki Tokuda†, Saneyasu Yamaguchi†

†Department of Information and Communications Engineering, Kogakuin University

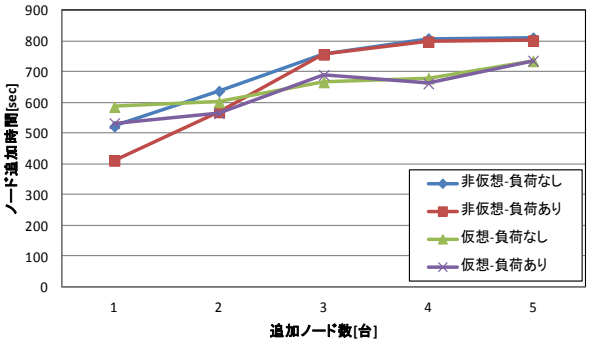

図 1. 同時追加のノード追加時間

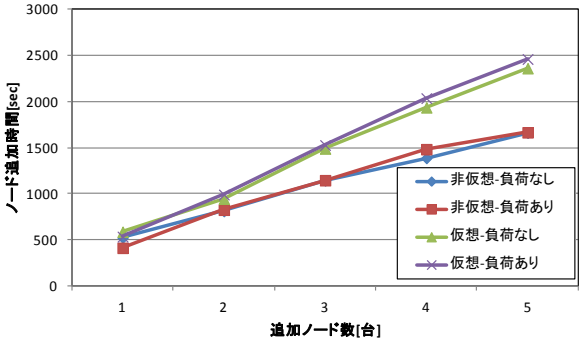

図 2. 順次追加のノード追加時間

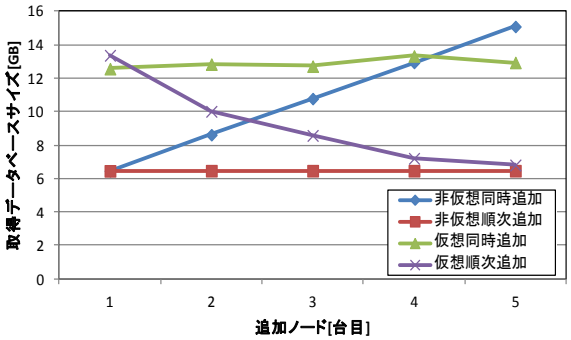

図 3. 各追加ノードが取得するデータ量

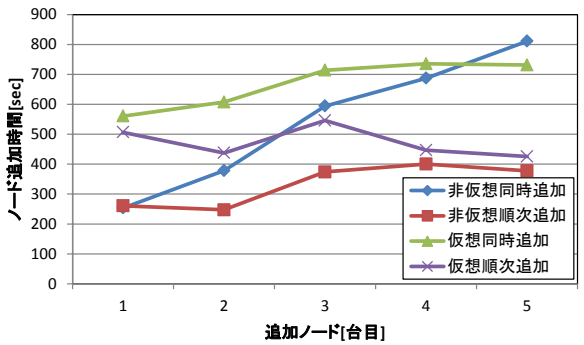

図 4. 各追加ノードの追加時間

図 1，図 2 から短い時間でノード追加を行える手法は同時追加の仮想であると考えられる．

4. 取得データベースサイズと追加時間の関係

次に，追加ノード数が 5 台の場合の各追加ノードが取得するデータサイズと各ノード追加にかかる追加時間を調査した．調査結果を図 3，図 4 に示す．図 3 から取得データ量は，仮想と非仮想ともに順次追加よりも同時追加の方が多いことがわかる．同時追加は順次追加と比べて多くのデータ量を取得してしまっていることがわかった．また，図 3 と図 4 より取得データ量が多い手法ほどノード追加時間が長くなっていることが分かり，取得データ量の増減が追加時間増減の原因となっていることが分かる．

5. 考察

本章でノードを 5 台追加したときの追加時間と各追加ノードの取得データ量について考察する．同時追加では非仮想の追加時間が遅いが，これは非仮想の 5 台目の取得データ量が仮想のどのノードよりも大きいためだと考えられる．また順次追加では仮想の追加時間が遅いが，これは各ノード取得データ量の合計が非仮想よりもデータ量が大きいためだと考えられる．

ノード追加時間が最も短かった手法は仮想同時追加である．しかし，同時追加のデータ転送量は非仮想順次追加のデータ転送量よりも多い．よって，仮想同時追加のデータ量を非仮想順次追加のデータ量に近づけることにより，さらなる時間短縮が可能であると考えられる．

6. おわりに

本研究では，Cassandra のノード追加時間に着目し，評価を行った．評価より，ノード追加時間は仮想同時追加が短いとわかった．また同時追加では余分なデータ量を取得してしまうことがわかった．

今後は，同時追加時のデータ取得量の削減方法について考察を行う予定である．

謝辞

本研究は JSPS 科研費 24300034, 25280022 の助成を受けたものである

参考文献

[1] Avinash Lakshman and Prashant Malik, “Cassandra-A Decentralized Structured Storage System”, LADIS 09, 2009

[2] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, “Dynamo: Amazon’s Highly Available Key-value Store”, SOSP ’07, 2007

[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes and Robert E. Gruber,“Bigtable: A Distributed Storage System for Structured Data”, IOSDI ’06 pages 205--218, 2006

[4] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears, “Benchmarking Cloud Serving Systems with YCSB” ACM symposium on Cloud computing, Pages 143-154, 2010

[5]堀内 浩基, 山口 実靖, “KVS における動的性能伸張性の向上” 研究報告マルチメディア通信と分散処理（DPS） 2013-DPS-154(39), 1-6, 2013.