1N-1

KVS の性能伸縮性の評価

堀内　浩基†　　山口　実靖‡

工学院大学大学院工学研究科電気・電子工学専攻

1. はじめに

近年，クラウドコンピューティングの普及に伴いデータベースのスケーラビリティの確保が重要視される様になり，Key-Value Store (KVS)が注目されている．KVS は実行時にノード数の増減を容易に行うことができ，クラウド環境などで動的に性能の伸縮させることができる．

そこで本稿では，代表的な KVS の一つである Cassandra を用い，この性能伸縮について調査し，その考察を行う．

2. Key-Value Store

KVS とは，Key と Value の組を書き込み，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ソフト管理ウェアである．読み書きの条件を簡単なものにすることによって，高い性能と高いスケーラビリティを達成することができる．代表的な KVS に Cassandra がある．

2.1 Cassandra

Cassandra は Dynamo の分散ハッシュテーブルと BigTable のデータモデルを併せ持った Eventually Consistent な分散システム構造の KVS である[1]．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．保存または検索する際は 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する．

3. 伸縮性の評価

KVS は実行時に性能を伸縮させることができる．本章でその応答性(ノード数増加にとなる性能向上が，どの程度の時間で達成されるか)の評価を行う．

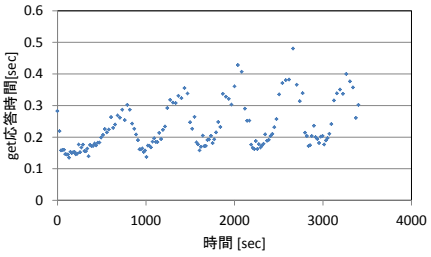

図 1 ノード数 1 台における平均 get 応答時間の変化

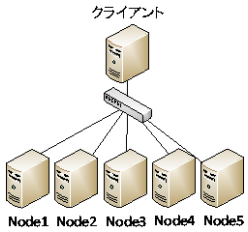

図 2 実験環境

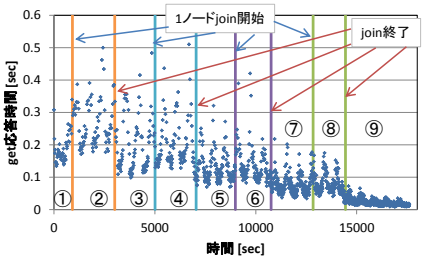

図 3 動的ノード追加時の性能変化 (1 ノードずつ)

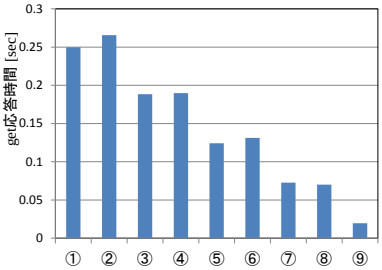

図 4 各状態の平均 get 応答時間の変化(1 ノードずつ)

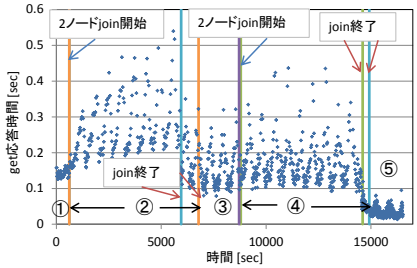

図 5 動的ノード追加時の性能変化 (2 ノードずつ)

「タイトル」英文による記述

†Kohki Horiuchi †Saneyasu Yamaguchi

†Graduate School of Electrical and Electronics Engineering,Kogakuin University

3.1 ノード数固定時の性能

まずノード数 1 台の環境において KSV の get 命令を多数回発行し，get 応答時間(KVS の get 命令 1 回にかかった時間)の変化を調査した．

Value サイズは 16[KB]，Key と Value の組み合わせは 655,360 組であり，全体のデータサイズは 10[GB]である．結果を図 1 に示す．図より，get 応答時間は安定せず周期的に増減を繰り返すことがわかる．

3.2 動的ノード追加時の性能 (1 ノードずつ)

図 2 の環境に KVS システム構築し，読み込み負荷をかけている状況でノード台数を 1 台から 5 台まで 1 台ずつ追加して性能の変化を調査した。

1 台時は，初期ノードがトークン値の 100%を担当し，その後追加されるノードのトークン担当範囲は全て 20%とした．よって，1 台追加されるごとに初期ノードのトークン担当範囲が 20%ずつ減少していく．get 応答時間の測定結果を図 3 に示す．図内の縦実線はノード追加開始時刻(join 処理開始時刻)であり，図内の縦点線はノード追加終了(join 処理終了時刻)である．図内の②④⑥⑧がノード追加中であり，図内の①③⑤⑦⑨がノード数が 1～5 でノード追加中でない状態である．各状態の平均 get 応答時間を図 4 に示す．

図より，ノード追加中は性能が劣化すること，負荷中にノード追加を実行した場合はノード追加(join 処理)に 2000 秒から 1600 秒かかること，ノード追加が終了するまで性能向上が得られないことがわかった．

3.3 動的ノード追加時の性能 (2 ノードずつ)

次に，1 ノードの状態から実験を開始してトークン範囲を 20%もつ持つノードを 2 ノードずつ同時に追加する実験を行った．測定結果を図 5 に示す．図より，2 ノード同時に追加した場合は，ノード追加に要する時間が 1 ノード追加時の？倍程度であり，追加時の性能劣化も 1 ノード追加時より程度が大きいことがわかる．よって，ノード追加を行ってから性能向上が得られるまでの応答時間は 1 台追加時より悪いと言える．

3.4 動的ノード追加時の性能 (4 ノード)

トークン範囲を 20%持つノードを 4 ノード同時に追加する実験を行った．測定結果を図 5 に示す．

本実験では，join 処理を測定期間(3 時間)以内に終了させることができず，ノードを追加することができなかった．

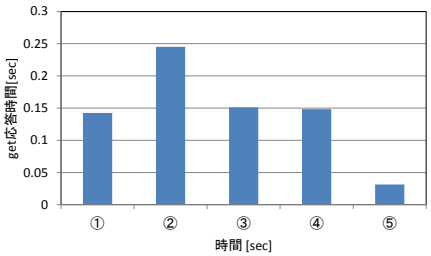

図 6 各状態の平均 get 応答時間の変化(2 ノードずつ)

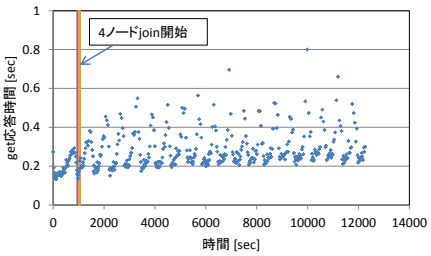

図 5 動的ノード追加時の性能変化 (4 ノード)

4. まとめ

本稿では，KVS にて読み込み負荷中にノードを追加し，性能向上の応答性について考察した．調査の結果，ノード追加中の性能が低下すること，少ないトークン範囲の変更を繰り返した方が応答性が良くなることが確認された．

今後は，追加中の性能の向上方法，応答性の改善手法について考察していく予定である．

謝辞

本研究は科研費（22700039）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