Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

KVS の動的ノード追加時間の短縮

御代川翔平† 徳田大輝† 山口実靖†

情報処理システムが扱うデータ量の増大に伴い，高いスケーラビリティを持つデータベース管理システムである KVS に注目が集まっている．代表的な KVS の一つである Cassandra は動的なノードの追加・削除が可能であり，負荷量に応じてノード数の増減を行うことが可能である．よって，高負荷時には多くの計算機を使用して高い性能でデータベース処理を行い，低負荷時には使用計算機数を減らして消費電力や消費計算機資源量を減らすといった動的最適化が可能となる．そして，動的最適化を効果的に行うには動的なノード追加，削除時間の短縮が重要となる．本稿では，動的ノード追加時間に着目し，その評価と短縮方法について考察を行う．まず，ノード追加処理におけるボトルネックの調査を行い新規ノードの Disk I/O がボトルネックであることを示す．そして，新規ノードの遅延書き込み時間の延長と Cassandra におけるページキャッシュ同期処理の削除を行うことによりノード追加時間の短縮が可能であることを示す．

1. はじめに

近年，大規模 SNS にかかる負荷は社会の動きに合わせて変動してきている．Facebook における時間単位ごとの投稿数は最多が 18:00 から 19:00 の間で 10.53%を占め，また 03:00 から 04:00 の間が最少であり 0.1%を占めている．そのため約 100 倍の投稿数差が生じている[1]．これら大規模 SNS の普及などにより，データベース(DB)管理システムのスケーラビリティの確保や動的スケールが重要視されるようになり，分散型 DB の KVS(Key-Value Store)が注目されるようになった．KVS はシンプルなデータベース構造であるため高いスケーラビリティを持つ．またサーバ(ノード)稼働中に新規ノードの追加や既存ノードの削除が可能といった動的性能伸縮性を持っているため，高負荷時ではノード数を増やしデータベース管理システムの性能を向上させ，負荷が低くなればノード数を減らし省電力化させるといったことができる．

本稿では，動的性能伸張性について着目し，KVS の一つである Cassandra を用いて動的ノード追加性能について調査し，遅延書き込み時間の拡大と Cassandra の改変によるノード追加性能向上についての考察を行う．

2. KVS(Key-Value-Store)

KVS(Key-Value Store)は，Key を指定して Value を得る仕組みのデータベース管理ソフトウェアである．機能が RDBMS などより単純になっているが，高いスケーラビリティを得ることができる．KVS の代表的な実装の一つに Cassandra[2]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[3]の分散ハッシュテーブルと Google 社の BigTable[4]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 のノード配置には仮想ノードを用いる手法と，用いない手法の二通りの方法がある．本稿では，前者を仮想ノード配置と，後者を非仮想ノード配置と呼ぶ．まず，非仮想ノード配置について述べる．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．これを図 1 に示す．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．KVS の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，そのハッシュ値を担当するノードが読み込みや書き込みを実行するノードとなる．ただし，後述のレプリカが存在する場合，それを持つノードも実行するノードの対象となる．また，稼働中のシステムに新規ノードを追加する場合，新規ノードのトークン値から新規ノードの担当範囲が決まり，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．稼働中のシステムからノードを動的に削除する場合，削除されるノードは担当している範囲のデータを新しく担当するノードに転送する．

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，2 以上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．レプリカは上記の担当ノードの後続ノードに配置される．

次に仮想ノード配置について述べる．仮想ノード機能は Cassandra-1.2.0 から実装された．仮想ノード配置では，物理ノードが複数（初期設定では 256 個）の仮想ノードを持ち，各物理ノードは自身が持つ仮想ノードの担当範囲の合計を担当する．仮想ノードのトークン値はランダムに決め

†工学院大学大学院
Kogakuin University graduate school

Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

られる．概要を図 2 に示す．仮想ノードを用いることにより，物理ノードが数台しか存在しない環境であっても，ハッシュ空間上では仮想的に数百台のノード（仮想ノード）が存在していると認識させることが可能である．非仮想ノード配置でも定常時のデータの分散を均等にする（あるいは偏りを持たせる）ことは，トークン範囲を均等にする（あるいは偏りを持たせる）ことにより実現できるが，ノード追加，削除時は（あるノードの担当範囲は連続している 1 つであるため）データの追加，削除は特定のノードに対して集中的に行われることとなる．これに対して，仮想ノード配置では，ノード追加，削除時のデータの追加削除も，分散して行われ，均等に（あるいは偏りを持たせて）行うことができる．

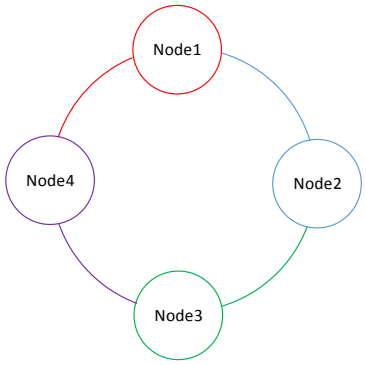

図 1　非仮想ノード配置

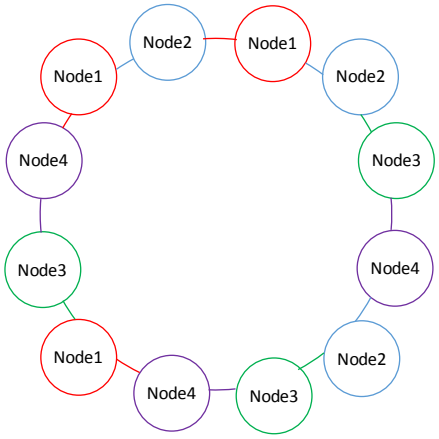

図 2　仮想ノード配置

3. 基本性能評価

Cassandra は，システム稼働中にノードを追加することによって動的に性能を拡張することができる．本章では，Cassandra システムにおけるノード追加処理に要する時間(join 命令を発行した時刻から，システム状態が Joining から Normal に変わる時刻までの時間)の評価を行う．

評価環境は既存ノードの PC3 台，クライアント PC 1 台，新規追加ノード用の PC1 台で構成される．すなわち，3 ノードに構成される KVS システムに 1 台のノードを追加し 4 ノードのシステムに変更する時間を測定する．仮想ノード配置を使用し，各ノードが持つ仮想ノード数は 256 である．データベースの作成にはベンチマークソフトの YCSB(Yahoo Cloud Serving Benchmark)[5]を使用した．測定はレコード数 1600 万件(約 17[GB])のデータベース，レプリカ数 3 で行った．使用した計算機の仕様は表 1 の通りである．

表 1　測定環境

	既存ノード			新規ノード
	Node1	Node2	Node3	Node4
OS	CentOS 6.3	CentOS 5.10	CentOS 5.10	CentOS 6.3
CPU	Intel Core i3CPU 530 @ 2.93GHz	Intel Celeron(R)CPU G1101 @ 2.27GHz	AMD Athlon Processor 1640B 2.7GHz	intel i3-2120 CPU @ 3.30GHz
Memory[GB]	2	2	2	8
HDD	VB0160EAVEQ	VB0160EAVEQ	Hitachi HDS72202	VB0250EAVER

ノード追加を行ったときの既存ノードと新規ノードの CPU 使用率の推移を図 3 に，Disk I/O 使用率の推移を図 4 に，新規ノードの read・write 量の累積の推移を図 5 に，各ノードの受信速度の推移を図 6 に，送信速度の推移を図 7 に示す．Node1~3 は既存ノード，Node4 は新規ノードである．

図 3 から CPU 使用率は既存ノードにて約 0[%]，新規ノードにて 20～30[%]であることがわかる．これより CPU 負荷は高くなく，CPU 処理はノード追加のボトルネックでないことが分かる．

図 4 から，Disk I/O 使用率はノードによってさまざまであることがわかる．Node1 ではほぼ 0[%]であり，Node2，3 においては，主に 10[%]程度から 40[%]程度の使用率であり，ある程度の負荷がかかっているがボトルネックとなっていないことが分かる．新規ノードである Node4 の Disk I/O 使用率はほぼ 100[%]であり，新規ノードの Disk I/O 処理がノード追加のボトルネックとなっていることが分かる．また図 5 は新規ノードの read，write 量別の累積量の推移を示し，read 要求はほとんどなく，大半が write 要求となっていることがわかる．

図 6 から新規ノードのみにおいて受信量が多いことがわかる．また図 7 から既存ノードである Node2，3 が送信量が多いことが分かる．Node2，3 の送信速度は 30[MB/s]以下であり，ネットワーク機器の性能上限（1Gbps）と比べ，十分に小さく，ネットワークがボトルネックとなっていないことが分かる．

以上のことからノード追加処理のボトルネックは新規ノードの Disk write 処理であることがわかる．

4. 参考実験

前章の実験より，ノード追加処理のボトルネックは新規ノードの Disk write 処理であることが分かった．本章にて，新規ノードの物理メモリ量とノード追加時間の関係についての考察を行う．

Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

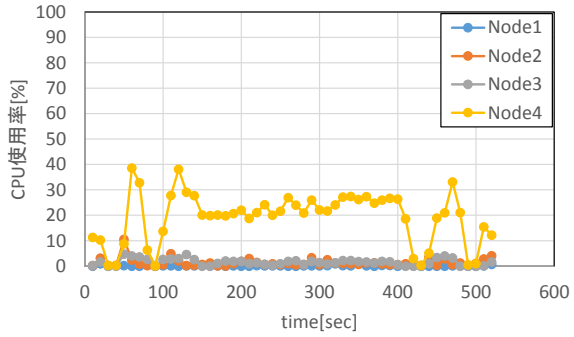

図 3　CPU 使用率

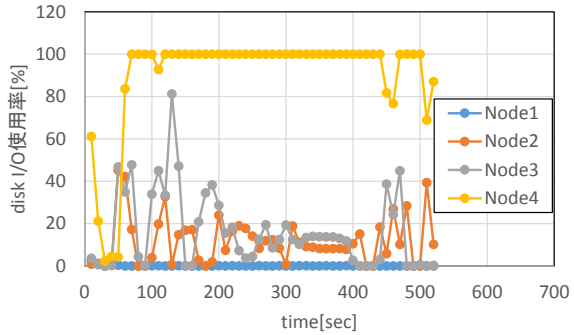

図 4　Disk I/O 使用率の推移

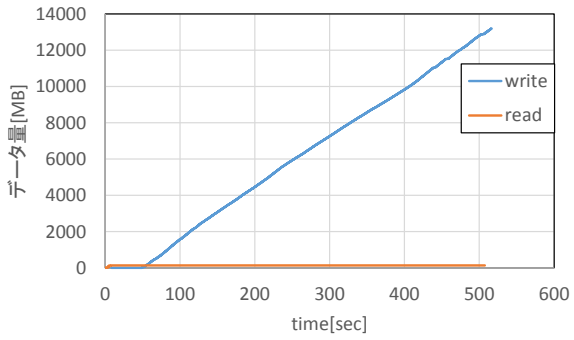

図 5　read・write の累積データ量の推移

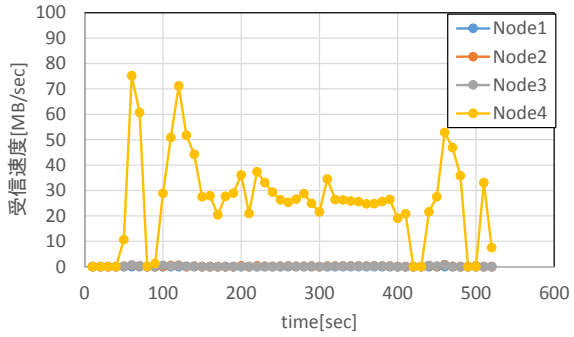

図 6　受信速度の推移

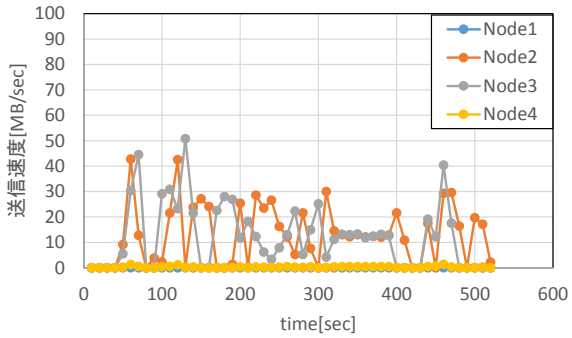

図 7　送信速度の推移

新規ノードの物理メモリ量が 2，4，6，8，10，12，14[GB] のときのノード追加時間の測定結果を図 8 に示す．図 8 から物理メモリ量を増やしてもノード追加時間が短縮しないことが分かる．これは物理メモリ量を用いるページキャッシュによる遅延書き込みが Cassandra のノード追加処理に対して効果がないことを意味しており，Cassandra が新規ノード追加実行中に新規ノードの HDD と物理メモリの同期を複数回にわたって行っていることが原因である．

同期は DB ファイル 1 個ごとに行われる．

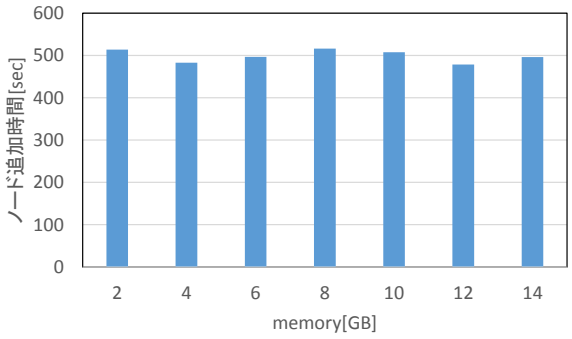

図 8　物理メモリ量によるノード追加時間

5.　ノード追加時間の短縮

5.1　遅延書き込み時間の延長

基本性能評価より新規ノードの Disk write 処理がボトルネックであることが分かった．本節では，新規ノードの遅延書き込み時間の延長によるノード追加時間の短縮について考察する．具体的には新規ノードの遅延書き込みの設定を表 2 のように修正する．

また，本稿の実験環境を含む KVS が稼働するサーバ計算機では，通常 KVS ソフトウェアのプロセスのみが主として稼働している．よってプロセス間の公平性の向上を目指す I/O スケジューラである CFQ は効果的に機能しないと予想される．これを踏まえ，上記に加え I/O スケジューラを初期設定の CFQ から Deadline に変更する手法についても考察する．

Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

以下に性能評価結果を示す．表 2 の変更前と変更後をそれぞれ“通常遅延”と“遅延拡大”と呼ぶ．実験は 3 ノードの KVS システムに 1 ノードを加える処理を行った．使用計算機は 3 章，4 章と同一である．

表 2　遅延書き込み時間の評価内容

	変更前（通常遅延）	変更後（遅延拡大）
dirty_expire_centisecs[10ms]	3000	60000
dirty_writeback_centisecs[10ms]	500	50000
dirty_ratio[%]	20	80
dirty_backgroud_ratio[%]	10	40

ノード追加時間の測定結果を図 9 に，新規ノードの Disk I/O 使用率を図 10 に，新規ノードの平均 Disk I/O 使用率を図 11 と図 12 に示す．図 11 がノード追加命令発行時（0 秒）からノード追加処理終了時までの平均であり，図 12 が Disk I/O 使用率が上昇して（80 秒）からノード追加処理終了までの平均である．また新規ノードに発行された Disk 書き込み命令（SCSI コマンド）を図 13 から図 16 に示す．各図の横軸は Disk 書き込み命令が発行された時刻であり，縦軸は発行 I/O 要求のアクセスアドレス（書き込みアドレス）である．またそれぞれの拡大図を図 17 から図 20 に示す．

まず図 10 について述べる．“通常遅延＋CFQ（初期設定）”と比較し“通常遅延＋Deadline”，“遅延拡大＋CFQ”，“遅延拡大＋Deadline”ではそれぞれ 5，10，8[%]のノード追加時間の短縮が実現されていることがわかる．図 10 から“通常遅延＋CFQ”と比べ，“通常遅延＋Deadline”，“遅延拡大＋CFQ”，“遅延拡大＋Deadline”は断続的に Disk I/O 使用率が下がっていることがわかる．図 11，12 から遅延書き込み時間の延長を行うことによって，Disk I/O 使用率が小さく低下していることがわかる．

図 13，図 14 と図 15，16 を比較すると，遅延時間拡大前は，2[GB]付近への書き込み要求と，220[GB]付近への書き込み要求が実験中を通して処理されており，拡大後は同アドレス付近への書き込み要求は短時間で集中的に処理されていることが分かる．すなわち遅延書き込み時間の延長を

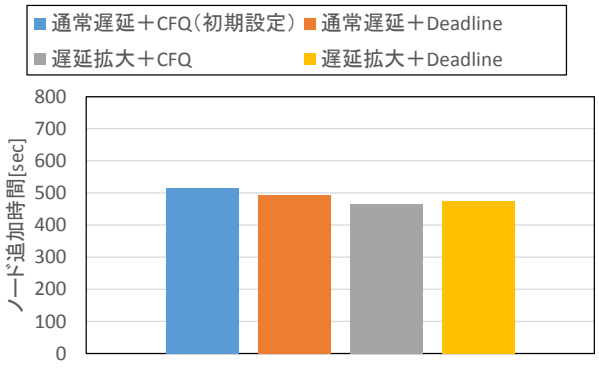

図 9　ノード追加時間（1）

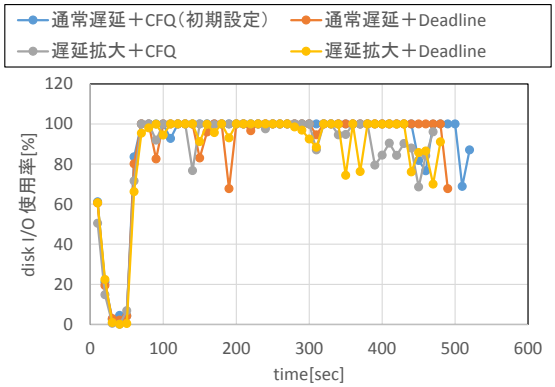

図 10　新規ノード Disk I/O 使用率（1）

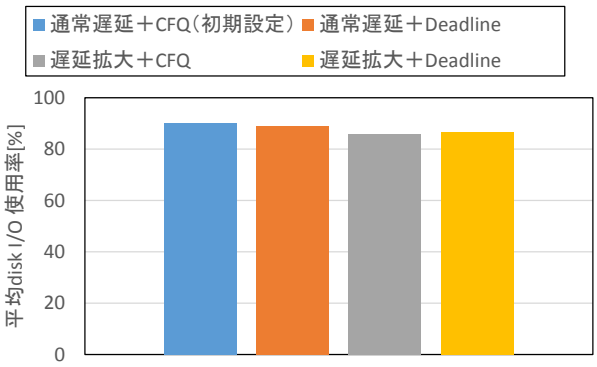

図 11　新規ノード Disk I/O 平均使用率（1）

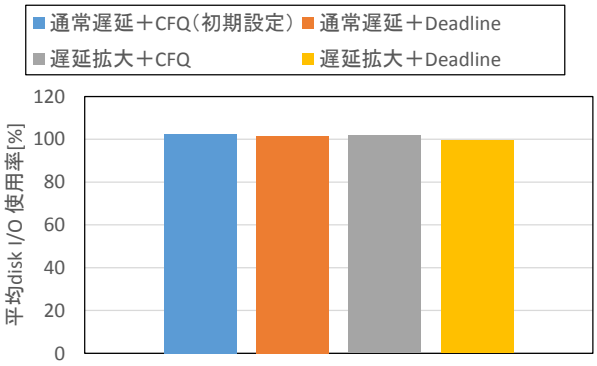

図 12　新規ノード Disk I/O 平均使用率

行うことによって，ディスクヘッドの長距離シーク（2[GB]付近と 150[GB]付近の間の移動や，150[GB]付近と 220[GB]の間の移動）の回数が減少していることが分かる．図 17，19 と図 18，20 を比較すると，CFQ を用いている図 17 や図 19 ではアドレス 153[GB]から 157[GB]の領域中で高頻度で 1[GB]以上の移動を行っているが，Deadline を用いる図 18 や図 20 では基本的にアドレスが増加する方向にのみディスクヘッドが移動していることが分かる．

以上より，遅延書き込み時間の拡大や I/O スケジューラの変更により，50[GB]以上の長距離シークや，1[GB]以上の中距離シークが減少し，ノード追加時間が短縮されていることが分かる．

Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

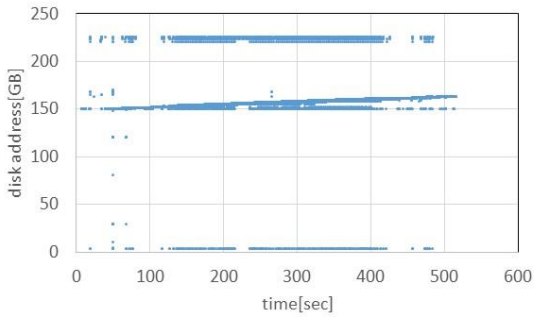

図 13　通常遅延＋CFQ における Disk I/O 要求

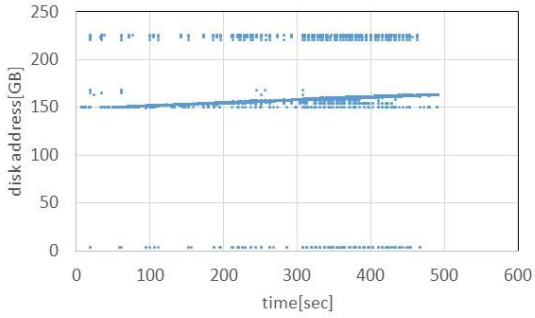

図 14　通常遅延＋Deadline における Disk I/O 要求

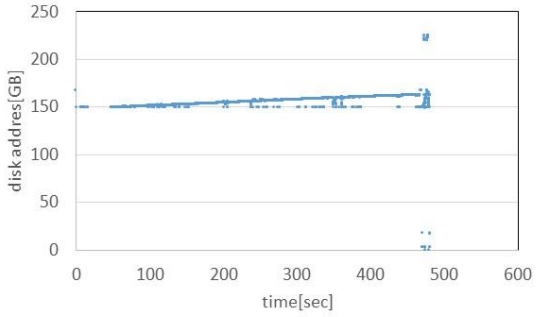

図 15　遅延拡大＋CFQ における Disk I/O 要求

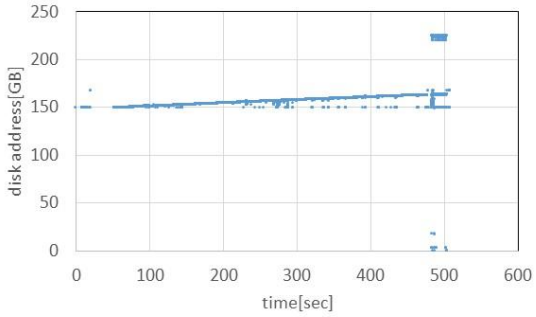

図 16　遅延拡大＋Deadline における Disk I/O 要求

5.2　Cassandra 内の同期命令の削除

本節では 4 章にて紹介した Cassandra 実装内にあるページキャッシュと HDD の同期命令を取り除き，ノード追加時間を短縮する手法について考察する．

同期命令を削除した Cassandra によるノード追加時間を図 21 に，新規ノードの Disk I/O 使用率を図 22 に，新規ノードの発行された Disk 書き込み命令（SCSI コマンド）を図 23 から図 26 に示し，またそれぞれの拡大図を図 27 から図 30 に示す．図 21 より遅延書き込み時間の拡大，I/O スケジューラの変更，同期命令の削除によりノード追加時間が最大で 19[%]短縮できることが分かる．また，図 22 より新規ノードの Disk I/O 使用率ほぼ 100%であることがわかる．

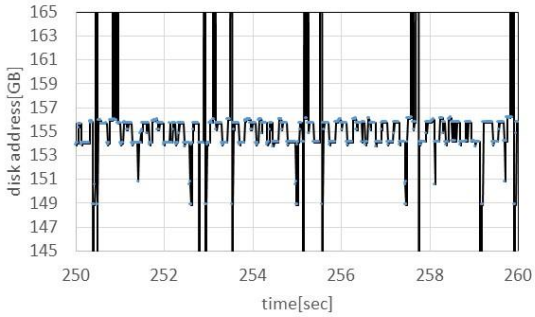

図 17　通常遅延＋CFQ における Disk I/O 要求（拡大）

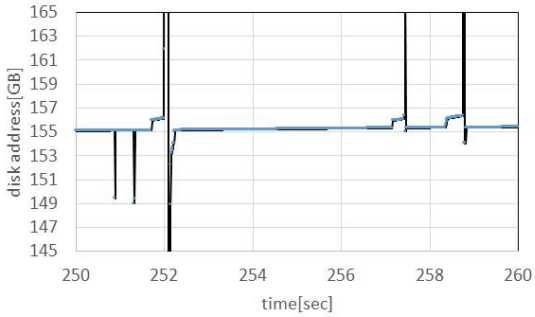

図 18　通常遅延＋Deadline における Disk I/O 要求（拡大）

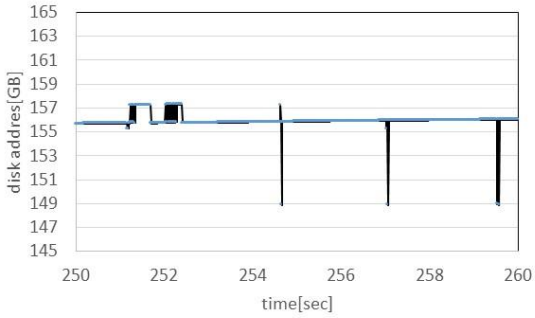

図 19　遅延拡大＋CFQ における Disk I/O 要求（拡大）

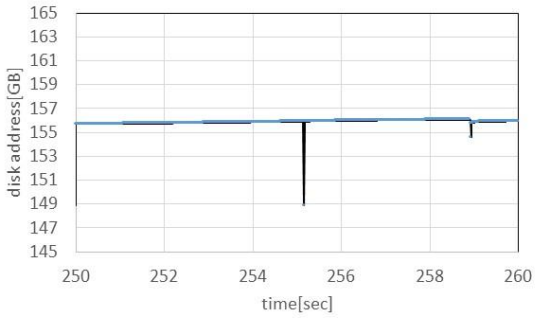

図 20　遅延拡大＋Deadline における Disk I/O 要求（拡大）

Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

図 23，24 と図 25，26 を比較すると遅延書き込み時間の延長を行うと 2[GB]付近と 220[GB]付近の Disk I/O 要求が短い時間に集中して処理されていることがわかる．また図 27，29 と図 28，30 を比較すると，I/O スケジューラを Deadline に変更することにより，1[GB]以上の中距離シークの頻度が減少することが分かる．

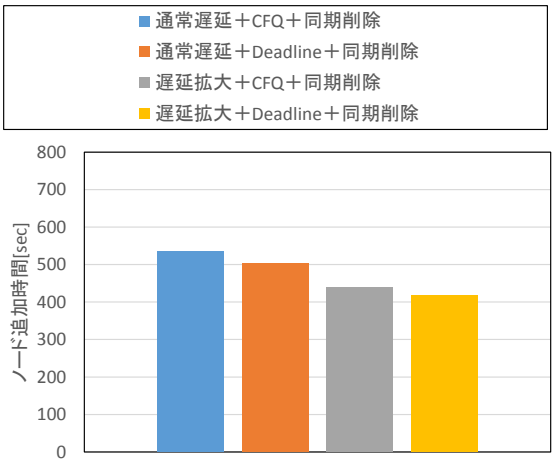

図 21　ノード追加時間（同期削除）

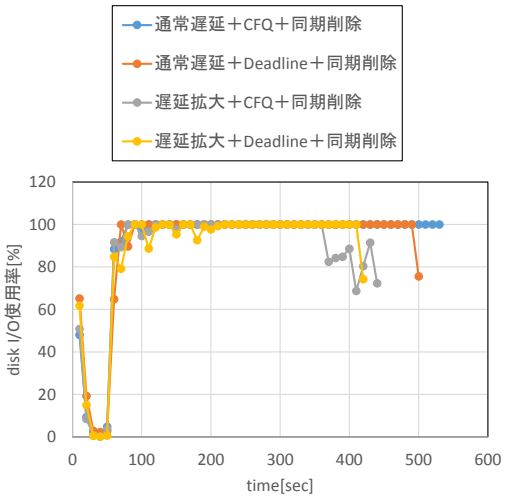

図 22　新規ノード Disk I/O 使用率（同期削除）

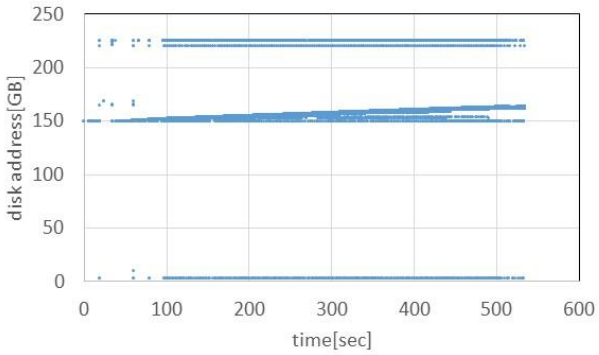

図 23　通常遅延＋CFQ における Disk I/O 要求（同期削除）

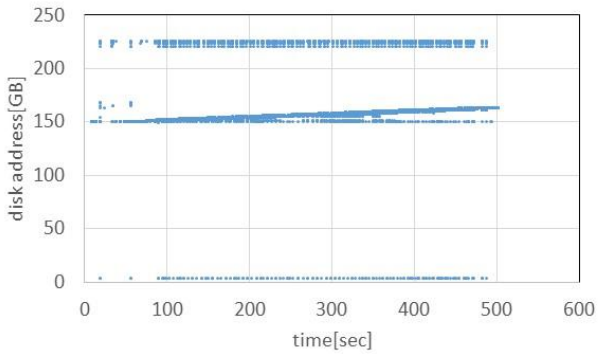

図 24　通常遅延＋Deadline における Disk I/O 要求（同期削除）

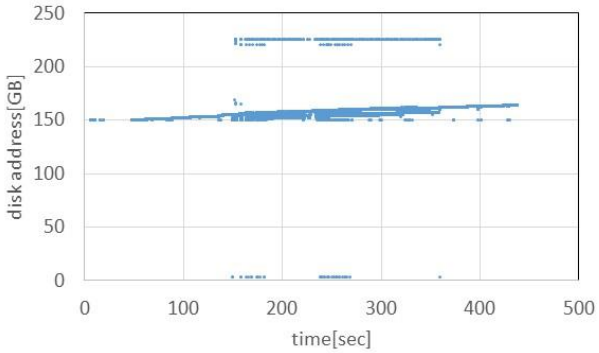

図 25　遅延拡大＋CFQ における Disk I/O 要求（同期削除）

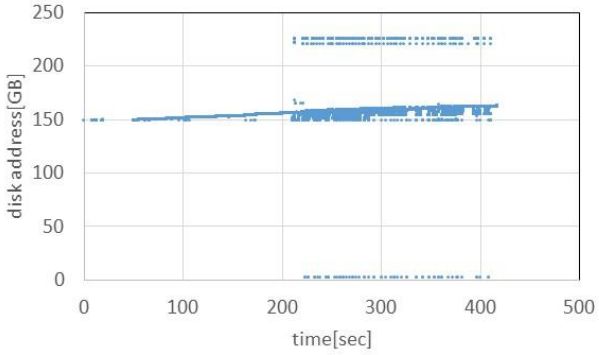

図 26　遅延拡大＋Deadline における Disk I/O 要求（同期削除）

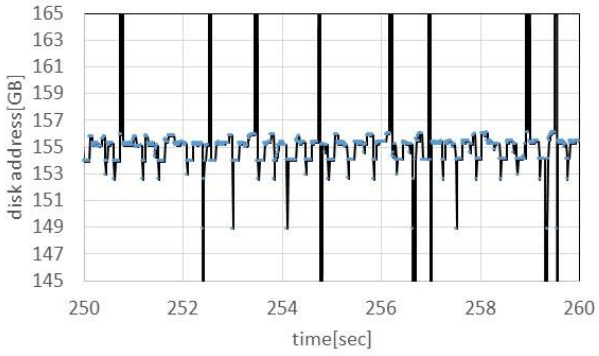

図 27　通常遅延＋CFQ における Disk I/O 要求（同期削除，拡大）

Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

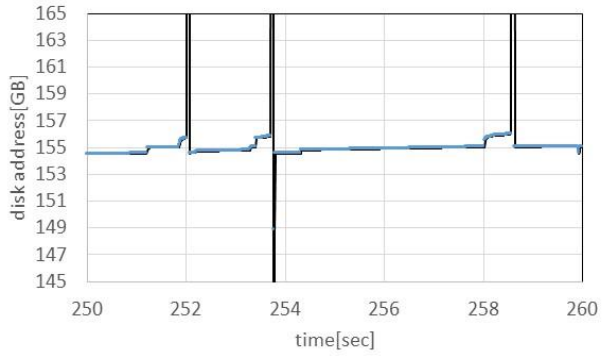

図 28　通常遅延＋Deadline における Disk I/O 要求（同期削除，拡大）

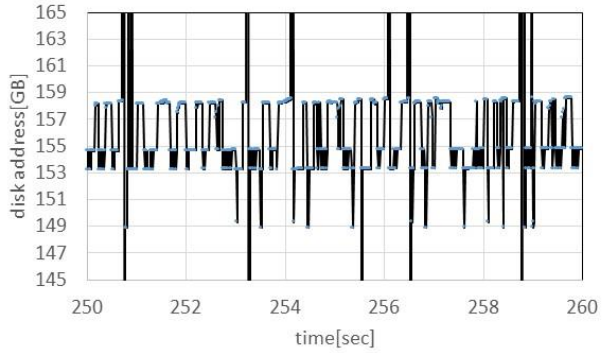

図 29　遅延拡大＋CFQ における Disk I/O 要求（同期削除，拡大）

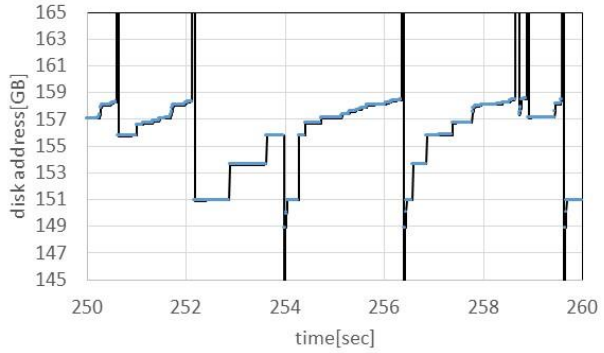

図 30　遅延拡大＋Deadline における Disk I/O 要求（同期削除，拡大）

5.3 同期命令の実行

ノード追加中の同期命令を取り除いたことにより，ノード追加時間の短縮が実現されたが，これは多くのデータをまとめてディスクに書き込むことにより HDD に高いシーケンシャル性でアクセスできた(HDD は多くの要求をまとめ，ソートしてからアクセスした方がより高いスループットでアクセスできる)ことと，ノード追加時にデータベースの一部のデータが dirty page としてページキャッシュ(物理メモリ)内に保存されており HDD に書き込まれていないことが理由と考えられる．よって，Cassandra システムにノードを追加するまでにかかる時間は短縮されるが，データの保存性は低下していると考えられる．

保存性とノード追加時間の関係を調査するために，ノード追加完了後に sync コマンドを実行し，sync が完了するまでの時間について調査した．調査結果を図 31 に示す．図 9 の“通常遅延+CFQ”と図 31 の“通常遅延+CFQ+sync”を比較すると，後者の方が時間が長く，初期設定で用いた Cassandra においても未同期のページが多く残されていることが分かる．

またノード追加後に同期(sync)を行うと，I/O スケジューラ CFQ においては大幅な処理(ノード追加+sync)時間の増加があり，deadline においては処理時間の増加が少ないことが分かる．

図 21 と図 31 を比較すると，sync を行う場合は遅延書き込み時間がノード追加時間に影響を与えないことが分かる．例えば，“通常遅延+CFQ+同期削除+sync”と“遅延拡大+CFQ+同期削除+sync”の処理時間はともに 700 秒程度である．これは遅延書き込み拡大により dirty page として残っていたページが sync コマンドにより HDD に同期されるためであると考えられる．“通常遅延＋CFQ”と“遅延拡大＋Deadline＋同期削除＋sync”を比較すると，両者ともに 500 秒程度である．これより，5.1 節，5.2 節の修正を行うことによりノード追加時間の増加なく保存性の向上（sync の実行）が実現されていることがわかる．

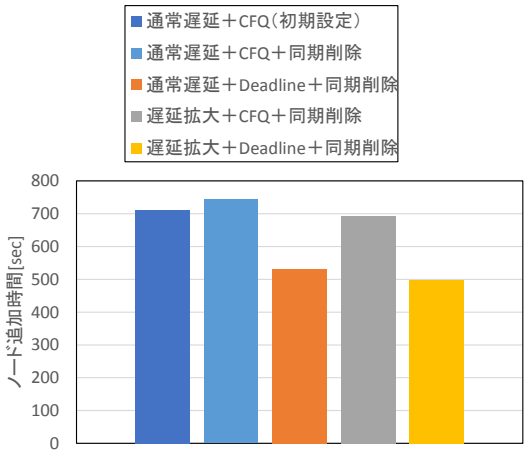

図 31　ノード追加完了後 sync コマンド完了までの時間

5.4 ボトルネック処理より導く性能上限

前節まで，ノード追加時間の短縮手法についての考察を行ってきた．本節にて，短縮されたノード追加時間と，ボトルネック処理から導くノード追加短縮の限界値との差について考察を行う．

3 章および 5 章の測定結果より，ノード追加のボトルネックは新規ノードの disk write 処理であることが分かった．よって，ノード追加に要する時間は，主として新規ノードの disk write 処理の速度により決定される．通常 HDD はシーケンシャル書き込みにおいて最も高い書き込み性能を提供できるため，シーケンシャル書き込み性能と同等の速度

Vol.2014-DBS-160 No.24
Vol.2014-OS-131 No.24
Vol.2014-EMB-35 No.24
2014/11/18

でボトルネック処理である disk write 処理を行った場合のノード追加が，その環境における最も時間の短いノード追加であると考えられる．同様にして，ノード追加時の新規ノードにおける disk write 性能と理想の書き込み性能(シーケンシャル書き込み性能)の差が，実際のノード追加時間とボトルネック処理から導かれる理想値との差を表していると考えられる．

以下にて，新規ノードとして用いた計算機におけるシーケンシャル write 性能と，ノード追加時の新規ノードの write 性能の比較を行う．シーケンシャル write 性能の評価には dd コマンドを用い，ブロックサイズを 1MB (`bs=1024k`)，回数を 12,288 回(`count=12288`)として約 12GB の書き込みを行い性能を評価した．この dd による性能をシーケンシャル write 性能(理想の書き込み性能)とする．図 32 は，シーケンシャル write 性能と，遅延書き込み時間を延長した場合のノード追加処理中の新規ノードにおける write 処理の速度を表しており，図 33 はシーケンシャル write 性能と同期処理を削除したときのノード追加中の write 処理の速度を表している．

図 32 から，遅延書き込み時間の延長を行うことによりノード追加時の write 処理の性能がシーケンシャル書き込み速度に近い値となっていることが分かる．よってノード追加時間が短縮され，それが限界に近いことが分かる．図 33 からも同様に，同期処理を削除した場合も write 処理の性能は限界値に近いことが分かる．

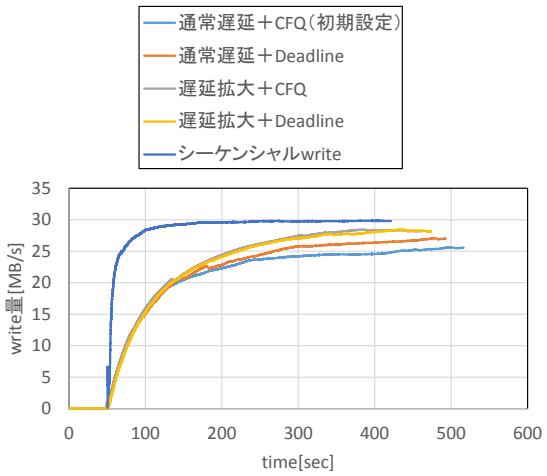

図 32　シーケンシャル write 性能と遅延書き込み時間延長によるノード追加時の write 性能

6. おわりに

本稿では，Cassandra のノード追加時間に着目し，評価と短縮手法に関する考察を行った．評価より，新規ノードの disk write 処理がノード追加のボトルネックであることが分かった．また，遅延書き込み時間の延長や Cassandra 実装内の同期処理を取り除くことによりノード追加時間の短縮が可能であることがわかり，達成された性能がボトルネック処理から導かれる理想値に近いことが確認された．

今後は，高負荷環境でのノード追加性能について考察する予定である．

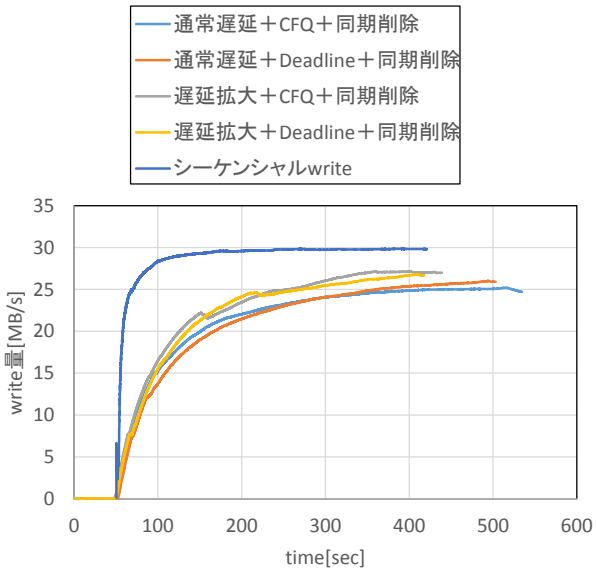

図 33　シーケンシャル write 性能と同期削除によるノード追加時の write 性能

謝辞

本研究は JSPS 科研費 24300034，25280022，26730040 の助成を受けたものである．

参考文献

[1] Facebook ページの投稿が一番多いのは、何曜日の何時？　～ 2,000 社の Facebook ページを集計 http://comzenbunosez.com/?p=1219

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

[3] 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).

[4] 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, (2006).

[5] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears, “Benchmarking Cloud Serving Systems with YCSB” ACM symposium on Cloud computing, (2010).

[6] 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）,Vol.2013-DPS-154(39),No.10 (2013)