様々なシステム構成における Cassandra の処理能力に関する考察

菱沼直子[†1]　竹房あつ子[†2]
中田秀基[†2]　小口正人[†1]

近年，クラウドコンピューティングの普及や，分散環境における一貫性の制限を緩和して処理性能を重視するアプリケーションが多く開発されている事に伴い，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．それに伴い従来のデータベース管理システムである RDBMS ではデータ格納や処理の柔軟性に不満が出る場面も見られるようになり，NoSQL と呼ばれる新しいデータベース管理システムが注目され始めた．そこで，本研究では NoSQL の実装のひとつであり，KVS 型データベースでありながら，複雑なデータ管理が可能な Cassandra と呼ばれる分散データベースに着目する．Cassandra に対しベンチマークツール YCSB を用いて書き込みや読み出しにかかる実行時間やスループット等を測定し性能評価を行い，その結果をもとに Cassandra の傾向を明確にするための考察を行う．これまでに小規模なクラスタを用いた性能測定は実行済みなので，本研究ではより詳細な Cassandra の傾向把握のためにより大きなクラスタを用いて Cassandra の傾向を調査し，特に一貫性レベルとレプリケーション数の変化に伴う Cassandra の振る舞いの変化等を調べる．調査の結果，Cassandra はある一定の負荷までは効率良く処理が行えていることが分かり，本実験環境では 1 ノードあたり 2 スレッド程度は無駄なく処理することができていた．また，読み出処理は実行結果が一貫性レベルやレプリケーション数といった設定条件に大きく依存し，書き込み処理はあまり依存していないことが確認できた．

Study on the processing power of Cassandra in a variety of system configuration

Naoko Hishinuma,[†1] Atsuko Takefusa,[†2]
Hidemoto Nakada[†2] and Masato Oguchi[†1]

In recent years, many applications have been developed to focus on the processing performance to mitigate the limitations of consistency in a distributed environment and the spread of cloud computing. As a result, produced personal information will be stored in large quantities on the network, and the amount of data that has been increasing explosively. As a result, discontent began to appear to the flexibility of storing and processing data in RDBMS, a traditional database management systems. In this study, we focus on a distributed database called Cassandra, which is one of the NoSQL implementations and enables complex data management. In order to investigate the performance of Cassandra, we employ the YCSB benchmarking tool and measure the execution time, required for its read and write processes, and the throughputs, etc. Performance measurement using a small cluster has been executed so far. In this study, we investigate the detailed performance of Cassandra using a larger cluster, varying consistency levels, the number of replications, and request workloads.

1. はじめに

近年，クラウドコンピューティングの普及や，分散環境における一貫性の制限を緩和して処理性能を重視するアプリケーションが多く開発されている事に伴い，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，ネットワーク上に存在するデータ量が爆発的に増加している．それに伴い，従来のデータベース管理システムである RDBMS ではデータの格納や処理の柔軟性に不満が出るようになり，NoSQL と呼ばれる新しいデータベース管理システムが注目され始めた．NoSQL では，データベースを簡単にスケールアウトさせることが出来る点や，単一故障点が存在しないことなどが必要不可欠な特徴であり，この様な特徴を有する実装が多く存在している．しかし，多くの NoSQL の実装は未だ発展途中であり，傾向や性能が十分に把握されていない．

そこで，本研究では NoSQL の実装の一つであり，複雑なデータ管理が可能な Apache Cassandra[1] と呼ばれる分散データベースに着目する．Cassandra の傾向を明確にするため，YCSB[2] ベンチマークツールを用いて書き込みや読み出し処理に掛かる実行時間を測定，評価する．これまでにノード数が 5 台のクラスタを用いての性能測定を行った結果，Cassandra が書き込み性能重視の NoSQL であることや，一定の値までの負荷ならば効率良く処理することが可能であることなどが確認済みである.[3] よって，本研究ではさらなる詳細な傾向をつかむために，ノード数が 9 台のより大きなクラスタを用いて性能測定を行

†1 お茶の水女子大学
Ochanomizu University
†2 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

い，評価した．評価では，ユーザが自由に一貫性の程度を変更できるという Cassandra 固有の特徴に着目し，一貫性の程度と処理性の関係を明確にする．本研究では特に，一貫性とレプリケーション数を考慮した性能測定，アクセス負荷を変化させた際の性能測定を行い，Cassandra の傾向を調査した．調査の結果，Cassandra はある一定の負荷までは効率良く処理が行えていることが分かり，本実験環境では 1 ノードあたり 2 スレッド程度は無駄なく処理することができていた．また，読み出処理は，実行結果が一貫性レベルやレプリケーション数といった設定条件に大きく依存し，書き込み処理はあまり依存せず，様々な条件下において高速処理が可能ということが確認できた．

2. Apache Cassandra

2.1 Apache Cassandra の概要

本研究では，KVS 型データベースである Apache Cassandra に着目した．KVS とは Key Value Store の略であり，保存したい value に対し，対応する一意の key を設定し，これらをペアで保存する方式を取るシステムの事である．特に，複数のサーバに分散してデータを保存出来る機能を持ったものを分散 KVS と呼ぶ．

Apache Cassandra(以下 Cassandra) は，Facebook 社が開発し，Apache プロジェクトとしてオープンソース化された分散データベース管理システムである．Cassandra の特徴としては，カラム型データ構造を持つリッチデータモデルである点が挙げられる．KVS 型のデータベースは通常 1 つの Key で 1 つの value を管理しているが，Cassandra はキースペース，カラムファミリ，キー，カラムの 4 つ，もしくはスーパーカラムを加えた 5 つの Key で 1 つの value を管理している．これにより通常の NoSQL よりも複雑なデータ管理が可能となっている．Cassandra のデータモデルを図 1，図 2 に示す．

その他の主な特徴としては，耐障害性の高さ，非中央集中型で単一故障点がない，データの分散保持を考慮した分散特性や柔軟性の高さ，一貫性の程度をユーザが自由に設定可能といった事が挙げられる．

また Cassadnra は書き込み性能を重視した NoSQL として開発されているため，書き込みの際にはディスクへのランダムアクセスが発生せず処理が高速になり，読み出しの際にはディスクへのランダムアクセスが複数回発生し処理が低速になるという実装になっている．

2.2 一貫性レベル

前節で述べた Cassanadra の主な特徴の中で，必要とする一貫性のレベルをクライアントがクエリに記述することで，自由に設定することが出来るという点は，RDBMS はもちろん，他の NoSQL でもあまり見られない，Cassandra 固有の特徴と言える．一貫性の強さと性能は一般にトレードオフの関係となっており，ユーザが一貫性のレベルを自由に設定できることから，ユーザが必要とする一貫性の程度と性能のバランスを考え，レベルを決定する必要が生じる．このための判断指標を提供する事が，本研究の目的の一つである．

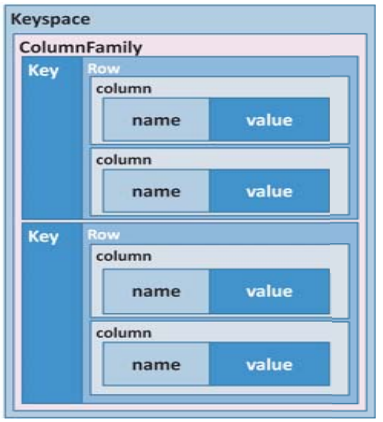

図 1 Cassandra のデータモデル key が 4 階層の場合

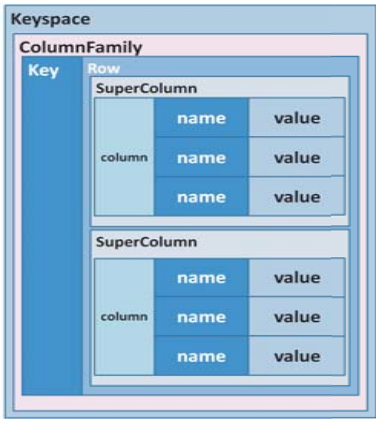

図 2 Cassandra のデータモデル key が 5 階層の場合

一貫性レベルは Cassandra の ConsistencyLevel オプションで書き込み，読み出しそれぞれについて設定できる．表 1 に設定可能な一貫性レベルの一部を示す. 一般的に一貫性レベル ONE，TWO は一貫性が弱いとみなされ、QUORUM，ALL は一貫性が強いとみなされる．Cassandra では，クライアントが書き込みと読み出しの両方において一貫性のレベルを指定することにより，一貫性の強さを制御することができる．一貫性の強さは R + W > N という式を満たしているかどうかで判断可能である．ここで R，W，N はそれぞれ，読み出しレプリカ数，書き込みレプリカ数，レプリケーション数であり，レプリケーション数が複製の数，読み出しレプリカ数と書き込みレプリカ数はそれぞれ，いくつの複製を読み出しまたは書き込みした時点で処理の完了とみなすかを表す．例えば，読み出しと書き込みの一貫性レベルを共に 2 を指定した場合には，読み出しレプリカ数，書き込みレプリカ数が共に 2 になる．この式を満たさない場合，Eventual Consistency（結果整合性）と呼ばれる弱い一貫性となり，この場合，書き込みの直後に読み出しを行うと，書き込みの結果が反映されていない結果が読み出される可能性がある．一方，この式を満たす場合を Strong Consistency (強一貫性) と呼び，読み出し時に，先行する書き込みの結果が反映されてい

ることが保証される[4].

表 1 Cassandra で設定可能な一貫性レベル

一貫性レベル	意味
ONE	各処理を対象ノードで行ない，その内の 1 ノードから返答があったら，処理が完了したとする
TWO	各処理を対象ノードで行ない，その内の 2 ノードから返答があったら，処理が完了したとする
THREE	各処理を対象ノードで行ない，その内の 3 ノードから返答があったら，処理が完了したとする
QUORUM	各処理を対象ノードで行ない，その内の処理が完了したとする過半数のレプリカ ((レプリケーション数/2)+1) から返答があったら，処理が完了したとする
ALL	各処理を対象ノードで行ない，その内のレプリケーション数で指定された数の全てのノードから返答があったら，処理が完了したとする

3. 実験システム

本研究では，Cassandra の性能と一貫性レベルとレプリケーション数の関係，アクセス負荷を変化させた場合の Cassandra の挙動を調査する．

まず，ユーザが一貫性レベルを決定するときの指標とするために，一貫性レベルとレプリケーション数を変化させると Cassandra の振舞にどのような影響が生じるか調査した．次に YCSB 側の設定である，スレッド数を増加させ，Cassandra にかかる負荷の大きさを変化させた時の Cassandra の振舞を調査した．

実験では，Cassandra によるクラスタを構築し，ベンチマークツールを用いて，書き込みや読み出しに掛かる実行時間と遅延，スループットを測定し評価した．測定では，Cassandra のレプリケーション数 (デフォルトは 1[オリジナルデータのみ])，一貫性レベル (デフォルトは ONE)，YCSB のスレッド数を設定する．

3.1 実験環境

クラスタ管理ツール Lucie[7] を用いて，マスターノード 1 台に対し，ワーカノード 9 台のクラスタを構築した．その後，ワーカノード 9 台に Cassandra を導入し，マスターノードにベンチマークツール YCSB[2](次節にて後述) を導入した．各ワーカノード上に，cassandra-1.0.6 をそれぞれインストールし Cassandra によるクラスタを構築した．マスターノードには，YCSB-0.1.3 をインストールし YCSB Client として使用した．ワーカノード，マスターノードの性能は表 2，3 に示した通りで，図 3 に構築したクラスタを示す．

図 3 に作成したクラスタを示す．

3.2 ベンチマークツール -YCSB-

今回の性能測定ではベンチマークツールとして，Yahoo! Research が開発したオープンソースである YCSB(Yahoo!'s Cloud Serving Benchmark)[2] を用いる．YCSB は，実アプリケーションに近いコアワークロードが用意されていて様々な NoSQL を公平に評価することができる．書き込みと読み出し処理の比率や，レコード数など様々な条件をユーザが自由に指定することもできる．

YCSB はロードフェーズで初期データロード後，トランザクションフェーズで測定対象 NoSQL に対して書き込みと読み出し操作を実行し，そのワークロードを実行するのにかかった時間と全体のスループット，各処理を実行する際に生じた遅延の平均値を集計する．

今回の性能測定では，書き込み，読み出しそれぞれの性能特性を調査するため、表 4 に示した書き込みと読み出しの比率の違う，Write-Only，Write-Heavy，Read-Heavy，Read-

表 2 worker node

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

表 3 master node

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

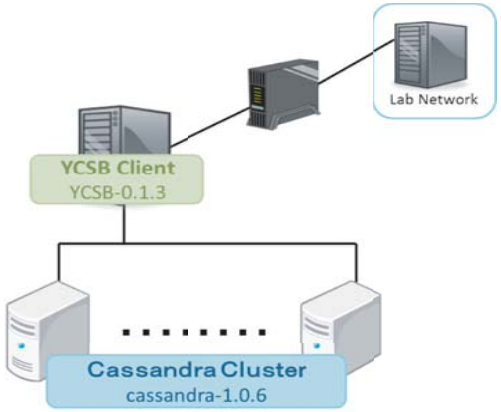

図 3 構築したクラスタ

Only の 4 種類のワークロードの中から Write-Only，Read-Only を使用した．また，4〜5 章の測定で用いたパラメータを表 5 にまとめた．Cassandra では，レプリケーション数（レプリカ数)、一貫性レベル（一貫性）を設定した．YCSB では，レコード数，オペレーション数，スレッド数を設定し，各測定で設定した詳細は表 5 の通りである．

表 4 ワークロード

ワークロード	読み出し比率	書き込み比率
Write-Only	0 %	100 %
Write-Heavy	50 %	50 %
Read-Heavy	95 %	5 %
Read-Only	100 %	0 %

表 5 測定概要

設定	パラメータ	4．アクセス負荷	5．一貫性
Cassandra	レプリカ数	3	3〜7
	一貫性レベル	ONE	ONE/TWO/THREE QURUM/ALL
YCSB	レコード数 (1 レコード 1KB)	100 万件	100 万件 1300 万件
	オペレーション数	10 万件	30 万件
	スレッド数	1〜25	1

4. アクセス負荷を変化させた性能測定・評価

4.1 測 定 概 要

Cassandra に対するアクセス負荷の大きさを変化させた時の Cassandra の振舞を調査した．測定では，Cassandra 側，YCSB 側の設定はそれぞれ表 5 の (4．アクセス負荷) に示した通りで，YCSB のスレッド数を 1〜25 と増加させていく事でアクセス負荷を増加させた．

4.2 性能測定結果

図 4，5 にワークロード Read-Only，Write-Only を行った際のスループットと各処理の際生じる遅延の平均値示す．縦軸がスループット (ops/sec) と各処理の際に生じる遅延 (ms) で，横軸がスレッド数となっている．また，図 6，7 にはワークロード Read-Only，Write-Only を行った際の実行時間を示す．こちらのグラフは縦軸が実行時間 (sec) で横軸がスレッド数となっている．

4.3 評　　価

図 4，5 から，スレッド数が増えるとスループットも増加している事が確認できる．図 4 では，Read-Only を実行するスレッド数が 16 程度まではスループットが増加する傾向がみられ，その後飽和して 20000〜25000(ops/sec) 程度に落ち着いていることが分かる．

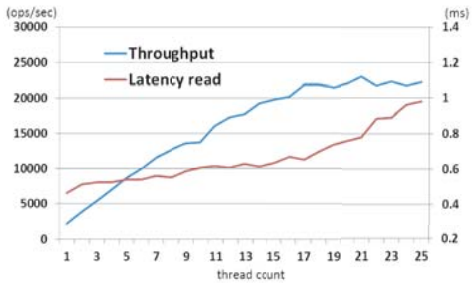

図 4 Read-Only を行った際のスループットと平均遅延

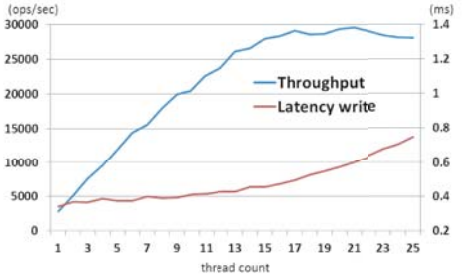

図 5 Write-Only を行った際のスループットと平均遅延

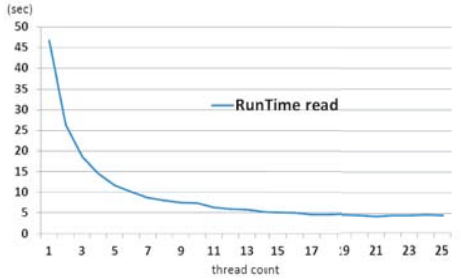

図 6 Read-Only を行った際の実行時間

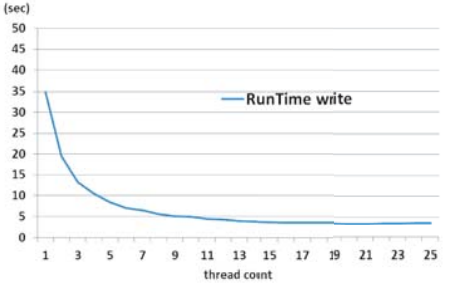

図 7 Write-Only を行った際の実行時間

同様に図 5 では，Write-Only を実行するスレッド数が 1〜14 まではスループットが急激に増加しており，その後飽和して 25000〜30000(ops/sec) 程度の値に落ち着いていることが分かる．以上のことより，Cassandra は読み出し書き込み処理どちらに関してもある一定値までは負荷を大きくしても，ノード数の増加に伴いスループットも増加し，効率良く処理ができていると言える．今回の実験環境においては，1 ノードあたり 2 スレッド程度までは効率良く処理ができ，書き込み処理の方がスループット値が高いことが確認できた．

また，実行時間のグラフ図 6，7 に着目してみると，どちらのワークロードを実行した際もスレッド数が 9，10 程度から緩やかに減少し、スループットが飽和する 17 スレッドあたりでほぼ横ばいとなっている．

5. 一貫性を考慮した性能測定

5.1 測定概要

次に一貫性のレベルの違いが Cassandra の振舞にどのような影響を与えるかを調査する．

測定では，Cassandra 側の設定がレプリケーション数は3〜7 と変化させ，一貫性レベルは ONE，TWO，THREE，QUORUM，ALL と変化させる．一貫性レベルは読み出しと書き込みそれぞれで同一なレベルを指定した．レコード数についてはメモリの大きさを考慮し，Write-Only を実行する際には 100 万件，Read-Only を行う際には 1300 万件としている．読み出しの際にレコード数が 100 万件だと，データがすべてメモリ上に乗ってしまう現象が起きる．その現象を防ぐために今回はレコード数を 1300 万件と大きくして，ディスクへの読み出しも発生するようにした．一方書き込みの際はレコード数が 100 万件でも 1300 万件でも実行結果に差がみられる事はほぼなかった．それ以外の設定は表 5 の (5．一貫性) に示したとおりである．

ただし，レプリケーション数が 3 の時は THREE=ALL となり，レプリケーション数 4，5 の時は QUORUM=THREE となるため，どちらか一方のみを実行した．

5.2 性能測定結果

図 8，図 9 はそれぞれ，一貫性レベル THREE，ALL を指定して，レプリケーション数を 3〜7 に増加させながらワークロード Write-Only を行った際のスループットを示している．縦軸がスループット (ops/sec) で横軸がレプリケーション数となっている．図 10，11 には，レプリケーション数 4 をとし，一貫性レベルを ONE〜ALL まで変化させてワークロード Write-Only，Read-Only を行った際のスループットを示す．縦軸はスループット (ops/sec)，横軸は指定した一貫性レベルである．

図 12，図 13 にはそれぞれ，一貫性レベル THREE，ALL を指定して，レプリケーション数を 3〜7 に増加させながらワークロード Read-Only を行った際のスループットを示している．縦軸がスループット (ops/sec) で横軸がレプリケーション数となっている．

5.3 評　　価

図 8 より，一貫性レベル THREE を指定してワークロード Write-Only を実行時にはレプリケーション数が 3〜7 に増加してもスループットの値に大きな変化が生じることはなく，2000〜2200(ops/sec) 程度の値で落ち着いていることが見られる．よって，書き込み処理に関しては，同一な一貫性レベルを指定した場合，レプリケーション数の増減に左右されることなく一定レベルの処理が行えることが分かる．

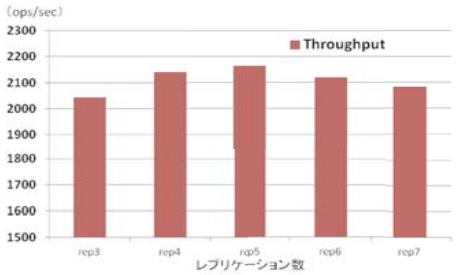

図 8 一貫性レベル THREE を指定して Write-Only を実行した際のスループット

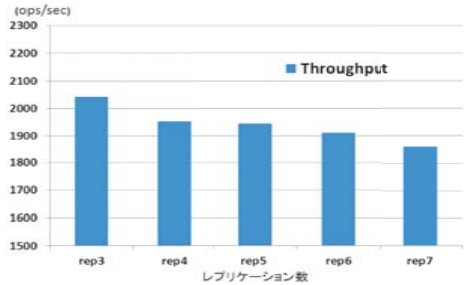

図 9 一貫性レベル ALL を指定して Write-Only を実行した際のスループット

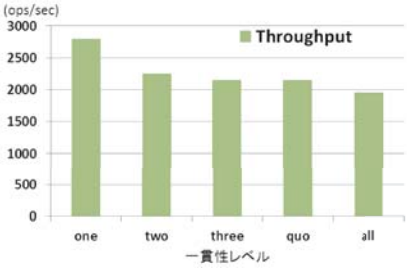

図 10 レプリケーション数が 4 で一貫性レベルを変更し Write-Only を実行した際のスループット

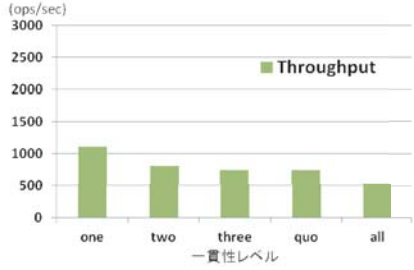

図 11 レプリケーション数が 4 で一貫性レベルを変更し Read-Only を実行した際のスループット

また図 9 では，グラフが右肩下がりになっていることより，一貫性レベル ALL を指定した場合レプリケーション数の増加に伴いスループットの低下傾向があると言える．これは，一貫性レベル ALL を指定した際には，処理の完了に必要とする返答の数がレプリケーションの数になるので，レプリケーション数が増えるにつれてスループットが低下するためである．

レプリケーション数を 4 に固定した図 10，11 に着目すると一貫性が強いレベルになるにつれてスループットが減少している．図 10 を見ると，ONE を指定した場合と TWO を指定した場合にはスループットの値に大きな違いが見られるが，TWO と THREE と ALL を指定した場合ではスループットの値にあまり大きな違いが見られなかった．(今回 THREE と QUORUM は同じ値になっている．) このことより，書き込み処理に関しては高い一貫性レベルをしても，処理性能の低下傾向が少なくてすむと考えられる．読み出し処理の関しては図 11 より，一貫性レベルを高いレベルにすると，明らかなスループットの低下が見ら

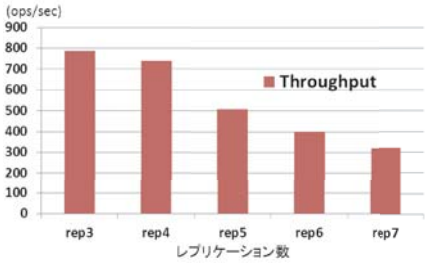

図 12 一貫性レベル THREE を指定して Read-Only を実行した際のスループット

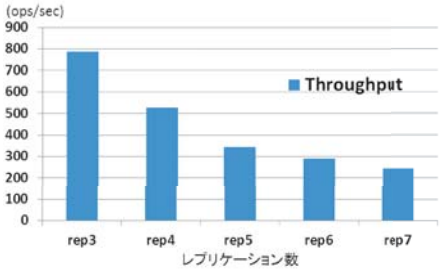

図 13 一貫性レベル ALL を指定して Read-Only を実行した際のスループット

れる事から，一貫性レベルを指定する際は一貫性レベルと処理性のトレードオフの関係を良く考えなければならないと言える．

図 12 については，一貫性レベル THREE を指定してワークロード Read-Only 実行した時にはレプリケーション数が 3〜7 に増加すると，スループットの値が減少している．この事より読み出し処理に関しては，同一な一貫性レベルを指定した場合，実行結果はレプリケーション数に依存すると考えられる．これはレプリケーション数が多いとデータの量が増えるので，読み出しを行う際により多くの処理が必要になる事と，レプリケーション数が少ないとメモリ容量によってはキャッシュを効果的に使用できるとめだと考えられる．図 13 からは，図 9 の書き込み処理の際と同様に，レプリケーション数の増加に伴いスループットの低下傾向があり，その理由も前述した通りである．

これらはの結果は，書き込み処理は一貫性レベルで指定した数だけディスクに sequential に書き込みを行ったら処理完了とみなす，というディスクへのランダムアクセスが発生しない書き込みの実装と，読み出しを行う際には毎回ディスクにアクセスをするという，ディスクへのランダムアクセスが複数回発生する読み出しの実装を考えると妥当な結果である．

6. まとめと今後の課題

現在注目されている NoSQL の実装の 1 つである Cassandra に着目し，Cassandra によるクラスタを構築し，ベンチマークツールである YCSB を用いて性能測定を行った．

これまでに小規模なクラスタを用いて性能測定を行った結果，Cassandra が書き込み性能重視な NoSQL であることや，小規模クラスタにおける一貫性と処理性能の間にトレードオフの関係があることが確認できた．そこで本研究ではより大きなクラスタを作成し，より詳細な Cassandra の性能を調査した．

まず，Cassandra へのアクセス負荷の大きさを変化させて測定を行った結果，Cassandra はある一定の負荷までは効率良く処理が行えていることが分かり，今回の実験環境では 1 ノードあたり 2 スレッド程度は無駄なく処理することができている．スレッド数をそれ以上に増加させると飽和状態となり，生じる遅延が大きくなり結果として性能が向上しなくなる．

次に，Cassandra の特徴である一貫性のレベルに着目し，同様の実験環境を用いて性能測定を行った．測定の結果，書き込み処理に関しては同一な一貫性レベルを指定した場合，レプリケーション数に実行結果が左右されることがないことが確認できた．よって書き込み処理に関しては，自分が達成させたい一貫性レベルに合わせ，著しい性能を低下を伴うことなくレプリケーション数を比較的に自由に設定できると言える．また，一貫性のレベルをより強いものに指定するとスループットの低下がみられたが，これは，一貫性のレベルをより強いものにすると各処理に対する返答をより多くのレプリカから受け取ることを必要とするためであり，妥当な結果を得られたと言える．書き込み処理の関しての以上の結果は，Cassandra の書き込みの実装が一貫性レベルで指定した数だけディスクに sequential に書き込みを行ったら処理完了とみなすという実装になっている点を考えると妥当な結果だと言える．

読み出しの処理に関しては同一な一貫性レベルを指定した場合，実行結果がレプリケーション数に依存していることが確認できた．このことより読み出し処理の際にはレプリケーション数を小さく抑えた方が処理性能が向上すると言える．また，一貫性のレベルをより強いものに指定するとスループットの大きな低下がみられ，読み出し処理に関しては一貫性レベルを決定する際は一貫性レベルと処理性の間のトレードオフの関係を考慮しなければならない．Cassandra の読み出しの実装が一つの処理毎にディスクへのランダムアクセスが発生するようになっていて，その影響を強く受けているため，今回の上記のような測定結果になった．

今後の課題としては，Cassandra と YCSB の様々な設定を用いてより詳細な性能測定を行い，測定結果を元に性能のモデル化や性能ボトルネックの原因分析など，より詳細な考察を行う．その後，より実環境に近い高遅延環境下等での性能測定やバッファサイズ等を考慮した性能測定，評価，RDBMS や他の NoSQL と性能比較も行いたい．最終的には測定した Cassandra の傾向を元に，読み出し処理性能など，Cassandra の性能改善する手法を提案する．

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) Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears, "Benchmarking Cloud Serving Systems with YCSB" ACM Symposium on Cloud Computing, pp143-154, June 2010.
3) 菱沼直子，竹房あつ子，中田秀基，小口正人"Cassandra による KVS データ処理におけるデータ容量と処理性能に関する考察" DEIM Forum 2012，C2-5，2012 年 3 月.
4) Eben Hewitt(著), 大谷晋平, 小林隆 (訳):Cassandra, オライリー・ジャパン,2011
5) 中村俊介，首藤一幸，"読み出し性能と書き込み性能を選択可能なクラウドストレージ" DEIM Forum 2011，C3-3，2011 年 2 月.
6) 中村俊介，首藤一幸, "読み出し性能と書き込み性能を両立させるクラウドストレージ" SACSIS2011，pp171-180，2011 年 5 月.
7) 高宮安仁，真鍋篤，松岡聡"Lucie：大規模クラスタに適した高速セットアップ・管理ツール" 情報処理学会論文誌コンピューティングシステム（ACS），Vol.44，SIG11(ACS3)，pp.79-88(2003).
8) Lucie:https://github.com/yasuhito/Lucie