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

菱沼　直子†　　竹房あつ子††　　中田　秀基††　　小口　正人†

† お茶の水女子大学　〒 112-8610 東京都文京区大塚 2-1-1
†† 産業技術総合研究所　〒 305-8568　茨城県つくば市梅園 1-1-1
E-mail: †naoko@ogl.is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp, †††oguchi@computer.org

あらまし　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の爆発的な増加が生じている．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．特に処理データ量に対するスケーラビリティなどの特性を明らかにする．測定から，Cassandra が書き込み性能重視な NoSQL で，一貫性と処理性にトレードオフの関係があることが分かった．また，ノード数を増加させると Cassandra のスループットが向上していくことも確認できた．

キーワード　Cassandra，NoSQL, クラウドコンピューティング，分散コンピューティング

A Study about the data volume and processing performance in KVS data processing by Cassandra

Naoko HISHINUMA†, Atsuko TAKEFUSA††, Hidemoto NAKADA††, and Masato OGUCHI†

† Ochanomizu University　2-1-1 Otsuka, Bunkyouku Tokyo 112-8610 JAPAN
†† National Institute of Advanced Industrial Science and Technology (AIST)　1-1-1 Umezono, Tsukuba, Ibaraki,305-8568, Japan
E-mail: †naoko@ogl.is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp, †††oguchi@computer.org

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．また，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．初めに，基本性能を調査し，その後一貫性に着目した性能測定，レプリケーション数を考慮した性

能測定，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．

その結果，Cassadndra が書き込み性能重視である NoSQL であることや，一貫性と処理性能にトレードオフの関係があることが確認できた．また，Cassandra はある一定の負荷までは効率よく処理が行え，ノード数が増加しても，スループット性能が向上していくことが確認できた．

2. 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，複数のサーバなど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．Cassandra のデータモデルを図 1 に示す．

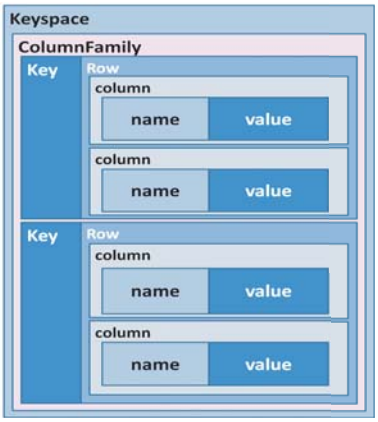

図 1　Cassandra のデータモデル key が 4 階層の場合

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．特に，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た．このための判断指標を提供する事が，本研究の目的である．

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．

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. 実験概要

本研究では，Cassandra の基本性能と一貫性レベル，レプリケーション数，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，Cassandra の基本性能を調査した．その後，Cassandra はユーザが一貫性レベルを指定して使用するので，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 Cassandra のレプリケーション数の違いによる性能測定を行い，最後に 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大きくした時の Cassandra の振舞の変化を調査した．

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

マシン 6 台のうち，ワーカ 5 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．

ワーカノードとしては，Dell PowerEdge SC1430，CPU

が Quad-Core Intel(R)Xeon(R) 1.6GHz，Memory が 2.0GB，OS が Linux2.6.9-55.0.2.EL(CentOS4.5) を用いた．マスターノードとしては，HP WorkStation xw8200，CPU が Intel(R)Xeon(R) 3.6GHz，Memory が 4GB，OS が Linux2.6.9-55.0.2.EL(CentOS4.5) を用いた．

図 2 に作成したクラスタを示す．

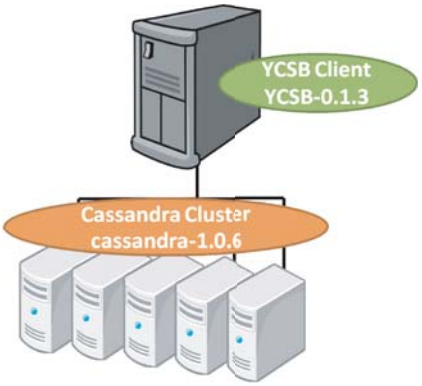

図 2　構築したクラスタ

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レイテンシを集計する．

今回の性能測定では，表 2 に示した a，b，c，g の 4 種類のワークロードを使用した．また，4〜7 章の測定で用いたパラメータを表 3 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．(読／書）では，読み出し，書き出しに対して一貫性レベルをどう指定したかが記述されている．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た．全測定において，レコード数は 100 万件 (1 レコードサイズは 1KB)，オペレーション数は 10 万件とした．

表 2　ワークロード

ワークロード	読み出し	書き込み
書き込みオンリー (g)	0 %	100 %
書き込みヘビー (a)	50 %	50 %
読み出しヘビー (b)	95 %	5 %
読み出しオンリー (c)	100 %	0 %

4.　基本性能測定，評価

4.1　測 定 概 要

まず Cassandra の基本性能を調べた．測定では，Cassandra 側の設定がレプリケーション数は 3，それ以外の設定は表 3 の (4. 基本) に示した概要で測定を行った．

表 3　今回行った測定の概要

設定	パラメータ	4. 基本	5. 一貫性	6. レプリカ	7. スレッド
Cassandra	レプリカ数	3	3	3，4	3
	一貫性	ONE	ONE TWO ALL	ONE TWO ALL	ONE
	(読/書)	同一	それぞれで指定	同一	同一
YCSB	レコード数 (1 レコード 1KB)	100 万件	100 万件	100 万件	100 万件
	オペレーション数	10 万件	10 万件	10 万件	10 万件
	スレッド数	1	1	1	1〜10

4.2　性能測定結果

図 3 に各種ワークロードを行った際の実行時間を示す．縦軸が実行時間 (sec) で，横軸が実行したワークロードの種類となっていて，右に行くにつれて読み出し比率の高いワークロードになっている．図 4 には各種ワークロードを行った際の読み出し，書き込み処理時に生じる遅延の平均値を示し，縦軸が各処理の際に生じる遅延 (ms)，横軸が図 3 と同様に実行したワークロードとなっている．

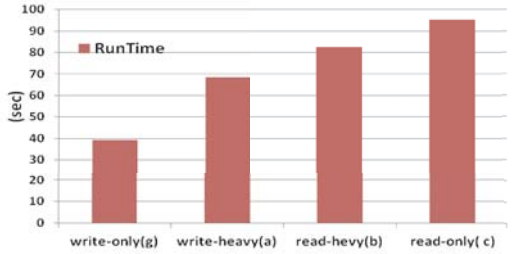

図 3　各種ワークロードの実行時間

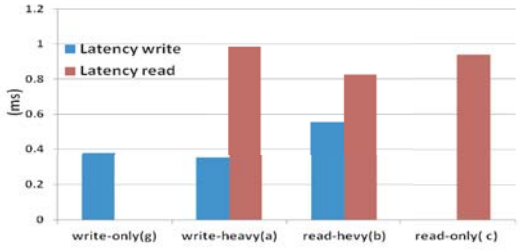

図 4　各種ワークロードの読み出し，書き込み処理の際に生じる遅延

4.3　評　　価

図 3 のグラフから，読み出しの比率が大きくなるにつれて，実行にかかる時間も大きくなっていることが読み取れる．図 4 では，読み出し時に生じる遅延は，書き込み時の遅延よりも大きいことが分かる．これらのことから，読み出し時の遅延は書き込み時の遅延より，全体の実行時間に対して大きな影響を与えていると考えられる．これらの測定結果は，Cassandra は元々書き込み性能を重視した NoSQL として開発されているため，妥当な結果だと言える．

5.　一貫性を考慮した性能測定，評価

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 3，一貫性レベルは ONE，TWO，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レベルを指定した．それ

以外の設定は表 3 の (5. 一貫性) に示したとおりである．ただし 2 章で述べたように，この他に Cassandra が設定可能な一貫性レベルとして QUORUM があるが，今回の設定条件では，QUORUM が返答を必要とする過半数のレプリカの数が 2 となり，一貫性レベル TWO を指定した場合と同じになるため，QUORUM は使用せず TWO を使用した．

5.2 性能測定結果

図 5 に各種ワークロードを行った際の実行時間を示す．縦軸が実行時間 (sec) で，横軸が実行時に設定した一貫性レベル (読み出しの際の一貫性レベル__書き込みの際の一貫性レベル) となっている．図 6 には，ワークロード c，g を用いた際の読み出し，書き込み処理時に生じる遅延の平均値を示し，縦軸が各処理の際に生じる遅延 (ms)，横軸が設定した一貫性レベル (読み出しの際の一貫性レベル__書き込みの際の一貫性レベル) となっている．図 5 と図 6 の測定では，書き込みと読み出し共に同一な一貫性レベルを指定した．

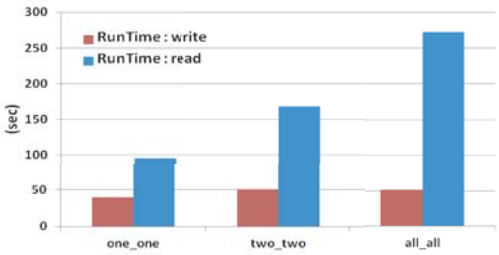

図 5　ワークロード c(読み出しオンリー),g(書き込みオンリー) の実行時間

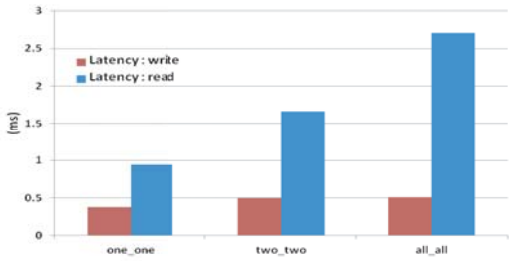

図 6　ワークロード c(読み出しオンリー),g(書き込みオンリー) の読み出し，書き込み処理の際に生じる遅延

図 7 には，書き込みの一貫性レベルのみを変化させてワークロード c，g を実行した際に生じる遅延の平均値を示す．縦軸が遅延の平均値 (ms) で，横軸が実行したと設定した一貫性レベル (読み出しの際の一貫性レベル__書き込みの際の一貫性レベル) なっている．図 8 には，先ほどとは逆に，読み出しの一貫性レベルのみを変化させてワークロード c，g を実行した際に生じる遅延の平均値を示す．こちらも図 7 同様に縦軸が遅延の平均値 (ms) で，横軸が実行したと設定した一貫性レベル (読み出しの際の一貫性レベル__書き込みの際の一貫性レベル) なっている．

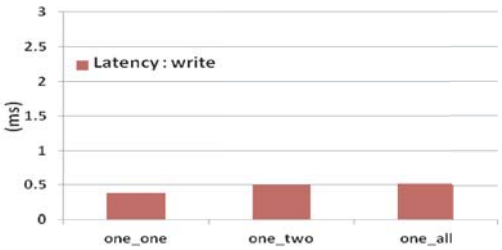

図 7　書き込みの一貫性レベルのみを変化させた場合のワークロード g(書き込みオンリー) の書き込み処理時の遅延

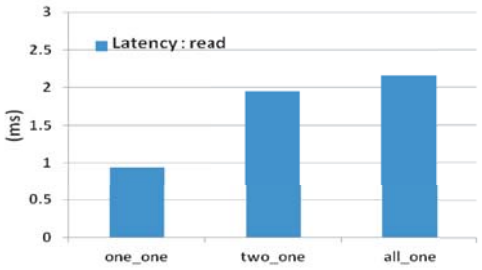

図 8　読み出しの一貫性レベルのみを変化させた場合のワークロード c(読み出しオンリー) の読み出し処理時の遅延

5.3 評　　価

図 6 では，基本性能測定と同様に読み出し時の遅延は，書き込み時の遅延よりも大きく，読み出し時の遅延は全体の実行時間に対して大きな影響を与えていることが分かる．また，図 6 で一貫性のレベルを変更した場合も，読み込み時の遅延に比べ，書き込み時の遅延にはあまり大きな変化は見られない．図 7，8 からは，書き込みの一貫性レベルを変化させるより，読み出しの一貫性レベルを変化させる方が生じる遅延に大きな変化が見られる．以上より，一貫性レベルを変化させた比較においても，読み出し処理に比べて書き込み処理の方が効率的に行われていることが分かる．

また，図 6 から一貫性レベルを ALL に指定すると，読み出し時の遅延に関しては一貫性レベル ONE の約 1.3 倍，書き込み時の遅延に関しては一貫性レベル ONE の約 1.03 倍になっていることが読み取れる．これは，一貫性レベルを ALL にすると，レプリケーション数で指定した数と同じ数のレプリカから返答を待つのでこのような結果になったと考えられ，レプリケーション数を増加させると実行時間や遅延の増加が予想できる．読み出し処理の遅延の増加率が大きい理由としては，一貫性レベルを ALL に指定すると，読み出し処理の際は全ノードに検索を行うのに対し，書き込み処理の際はレプリケーション数で指定した数のノードが書き込みを受け付ければ処理完了とみなすためであると考えられる．

一貫性の強さに着目して評価を行うと，2 章で述べた方程式から一貫性レベル ONE を使用した場合は弱一貫性を持ち，一貫性レベル TWO と ALL を使用した場合は強一貫性を持つことになる．このことを踏まえて測定結果をみると，今回の測定条件では一貫性レベルと処理時間がトレードオフの関係にあるものの，一貫性レベルを TWO にした場合が一番，速さと正確さを兼ね備えていると言える．

6. レプリケーション数を考慮した測定，評価

6.1 測 定 概 要

次にレプリケーション数を考慮した性能測定を行う．測定では，Cassandra 側の設定がレプリケーション数 3 と 4，一貫性レベルは ONE，TWO，ALL と変化させる．レプリケーション数が 3 の場合には一貫性レベル ALL を指定すると，3 ノードからの返答を必要とし，レプリケーション数が 4 の場合は，4 ノードからの返答が必要となる．一貫性レベルは読み出しと書き込みで同一なレベルを指定した．YCSB 側の設定としては，表 3 の (6．レプリカ) に示した通りである．

6.2 性能測定結果

図 9，10 にレプリケーション数を 3 と 4 に指定した場合の，ワークロード c(読み出しオンリー) と g(書き込みオンリー) の実行時間 (sec) を示す．

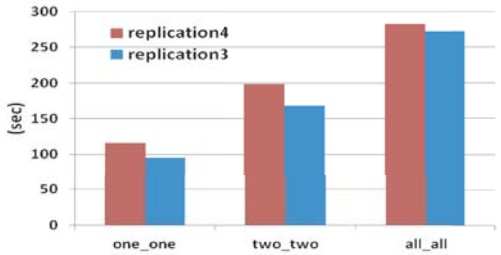

図 9　レプリケーション数を変化させてワークロード c(読み出しオンリー) を実行した際の実行時間

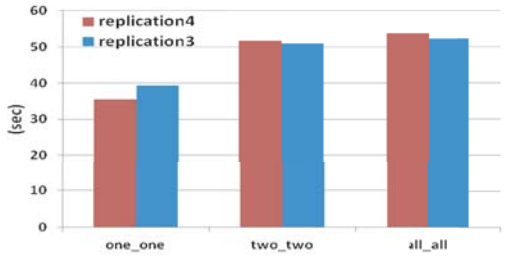

図 10　レプリケーション数を変化させてワークロード g(書き込みオンリー) を実行した際の実行時間

図 9 では，どの一貫性レベルを指定してもレプリケーション数 4 を指定した場合の方が実行時間が長くなっていることが確認できる．本来ならば一貫性レベル ONE と TWO を指定した場合はそれぞれ，1 ノード，2 ノードからの返答があれば処理の完了とみなすため，実行時間がレプリケーション数の違いによる影響を受けないと考えられるが，読み出し処理時はどの一貫性レベルを指定しても，古くなったデータを持つレプリカを最新の値に更新するリードリペアと呼ばれる処理がバックグラウンドで実行されるため，実行時間が増加している結果になったと言える．

図 10 では、一貫性レベル ONE を指定した場合のみ，レプリケーション数が 3 の場合の方が実行時間が長くなっているが，TWO ではほぼ同じくらいになり，ALL ではレプリケーション数 4 の場合の方が実行時間が長いという結果になった．一貫性レベル ALL を指定した場合は，レプリケーション数が 4 の場合の方が返答を必要とするノードの数が多いので，妥当な結果だと言える．しかしその実行時間の差はわずかであった．

7. スレッド数を変化させた測定，評価

7.1 測 定 概 要

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．測定では，Cassandra 側，YCSB 側の設定はそれぞれ表 3 の (7．スレッド) に示した通りで，YCSB のスレッド数を 1〜10 と増加させていく．

7.2 性能測定結果

図 11，12 にワークロード c(読み出しオンリー) と g(書き込みオンリー) 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を示す．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．

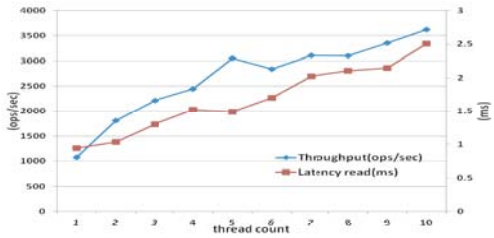

図 11　スレッド数を変化させてワークロード c(読み出しオンリー) を実行した際のスループットと処理の際に生じる遅延の平均値

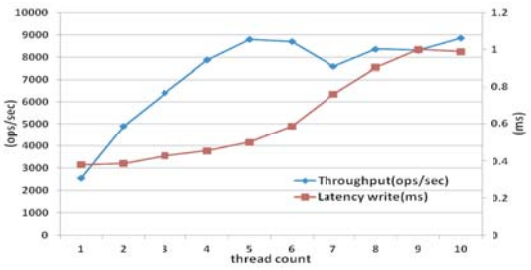

図 12　スレッド数を変化させてワークロード g(書き込みオンリー) を実行した際のスループットと処理の際に生じる遅延の平均値

7.3 評　　価

図 11，12 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 11 では，ワークロード c を実行するスレッド数が増加すればするほどスループットも増加する傾向が見られる．図 12 では，ワークロード g を実行するスレッド数が 1〜5 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 8000〜9000(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 はある一定の値までは負荷を大きくしても書き込み読み出し共に効率よく処理できると言える．読み出し処理については，今回の測定ではスレッド数の増加に伴いスループットも増加していて，この傾向はスレッド数が 10 以上になっても続くと予想される．したがって，まだ性能向上の余地があると考えられる．

また，スレッド数が 2 の場合は書き込み読み出し共にスレッド数が 1 の場合と比べてスループットが約 2 倍になっていることがグラフから読み取れる．このことより，Cassndra はノード数が増加しても，スループット性能が向上していくことが確認できた．

8. 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．

まず，Cassandra の基本性能を確認するために性能測定を行った結果，読み出しの際に生じる遅延の方が，書き込みの際に生じる遅延に対して約 3 倍の大きさであることや，全体に与える影響が大きかったことなどから，Cassandra は書き込み性能を重視した NoSQL であることが確認できた．

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結果，一貫性のレベルをより強いものに指定すると，実行時間と各遅延の増加，スループットの低下がみられた．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．また今回の測定を行った条件では，一貫性レベルを TWO にした場合が、一貫性と各処理に対する性能どちらも高いレベルで保てることが分かった．

レプリケーション数を考慮した測定を行った際には，読み出しの際はレプリケーション数が全体の実行結果に影響を与えていることが確認でき，書き込みの際はあまり大きな影響を与えていないことが確認できた．また，今回指定したレプリケーション数が 3 と 4 という連続した値だったため，差があまりはっきりと見られなかった．

最後にスレッド数を変化させて測定を行った結果，ある一定の値までは，負荷をかけても Cassandra は効率良く処理が行えていることが分かり，Cassandra はノード数が増加してもスループット性能が向上していくことが確認できた．

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．また，Cassandra は元々大規模なデータを扱うことを前提として開発されていることを踏まえ、より大きなクラスタを構築する．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，評価を行い，Cassandra の性能改善の手法を提案する．

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] Eben Hewitt(著), 大谷晋平, 小林隆 (訳):Cassandra, オライリー・ジャパン,2011

[4] 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．

[5] 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．