GridDB と Cassandra のパフォーマンスとスケーラビリティ

Microsoft Azure 環境における *YCSB* パフォーマンス比較

2017 年 8 月 23 日

Revision 1.2.0J

目次

要約

フィックスターズは、東芝がオープンソースとして公開した NoSQL データベース「GridDB」の実行環境を Microsoft Azure 上に構築し、YCSB (Yahoo! Cloud Serving Benchmark) を使用したベンチマークを実施しました。また、性能比較のため、主要な NoSQL データベースの 1 つである Apache Cassandra のベンチマークも併せて実施しました。ベンチマークは、大小 2 つのサイズのデータベースと、1～32 ノードのクラスタシステムで行いました。これにより、様々なワークロードパラメータ間で異なる構成のデータベースの性能がどのように変化するかを測定しました。

このパフォーマンスベンチマークの結論として、GridDB がスループットとレイテンシの両方において Cassandra を上回るとともに、GridDB が真にスケーラブルであり、長期運用で一貫したパフォーマンスを提供可能であることが分かりました。

イントロダクション

NoSQL データベースは、リレーショナルデータベースの持ついくつかの問題を克服し、優れた拡張性、信頼性、柔軟性が提供できるように設計されています。クラウドコンピューティングやモバイルコンピューティング、Internet of Things などに代表されるように、より多くのデータを収集し、処理するような新しいテクノロジーが発展するに従って、NoSQL データベースは 最初に考慮すべき選択肢となっています。

GridDB は、東芝が開発した分散型 NoSQL データベースです。メモリを主、ストレージを従としたハイブリッド型インメモリデータベースであり、コンテナレベルで ACID (原子性、一貫性、独立性、永続性) を保証しており、豊富な機能を備えています。また、キーコンテナ型または時系列データベースとして使用することもできます。

Cassandra はオープンソースの分散型 NoSQL データベースです。Cassandra は、主要な NoSQL データベースの中で、高可用性と分散型設計を維持しながらも、高いパフォーマンスを持つことで知られています。

YCSB (Yahoo! Cloud Serving Benchmark) は、Java で書かれた NoSQL または Key-Value ストアデータベースに対するベンチマークツールです。

環境

Azure の構成

まず、アメリカ西海岸地域で 65 個の Standard D2 インスタンスを持つリソースグループを 3 つ作成しました。Standard D2 インスタンスは、2.40GHz、7GB のメモリ、1Gbps のネットワーキング、100GB のローカル SSD で動作する 2 つの Intel Xeon CPU E5-2673 コアで構成されています。GridDB と Cassandra のデータファイルをローカル SSD に保存し、OS ディスクは永続的なページブロブに保存しました。

各インスタンスは、OpenLogic Centos 6.5 Linux イメージをベースとしています。最初のインスタンスにはパブリック IP アドレスが含まれ、ヘッドノードとして動作します。最大 32 個のインスタンスが NoSQL データベースサーバとして使用され、同数のインスタンスによって YCSB クライアントを実行しました。

ヘッドノードは、サーバの起動、YCSB クライアントの実行、リソース使用率統計の収集、および結果の集計を行いました。

ソフトウェアのバージョン

東芝が提供する RPM パッケージを使用して、各 Azure ノードに GridDB バージョン 3.0 CE をインストールしました。

Cassandra は、Datastax のコミュニティ YUM リポジトリからバージョン 3.4 をインストールしました。

YCSB は 2016 年 7 月 5 日時点の GitHub リポジトリからのコピーを使いました。Cassandra2 データベースドライバは特に変更していません。YCSB GridDB ドライバは 2016 年 8 月に東芝より提供されたものを使用していますが、マルチキャストではなく固定リスト接続方式を使用するように変更しています。

ソフトウェアの構成

GridDB

GridDB では、ほとんどの場合で、デフォルトまたは推奨とされている設定値を使用しました。今回の検証結果が、これら推奨値がセットアップの際の理想的な値であるという証拠となっています。処理並列度 (concurrency) はコア数に合わせて 2 に設定し、チェックポイント用メモリバッファサイズ (checkpointMemoryLimit) は 512MB、メモリバッファサイズ (storeMemoryLimit) は 6,144MB に設定しました。この構成により、4GB のレコードをメモリに保持するための十分なスペースが得られ、他のシステムアクションのために約 512MB を確保しました。

GridDB サーバ間の接続方法は、 より一般的なマルチキャスト方式ではなく、 固定リスト方式を使用しました。これは、Azure や他の多くのクラウドプロバイダがインスタンス間のマルチキャストをサポートしていないためです。

デフォルト値からの唯一の変更点は、storeBlockSize を 64KB から 32KB に設定した点です。

Cassandra

Cassandra のベンチマークでは、ライトタイムアウトが発生しました。これは、他の多くのベンチマークでも報告されているようです。この問題を解決するために、YCSB ワークロードファイルで core_workload_insertion_retry_limit を 0 から 10 に増やし、write_request_timeout_in_ms、counter_write_request_timeout_in_ms、および range_request_timeout_in_ms をすべて 10 秒に増やし、read_request_timout_in_ms を 5 秒に増やしました。

クラスタの起動時間を短縮するために、n-1 個のシード・ノードを使用しました。n はノードの数です。 シード・ノードを 1 と 4 にした場合も検証しましたが、1 つのラック/データセンター環境ではパフォーマンスに影響は見られませんでした。

同時接続のリーダー、ライターの数は、検証によって確認されたようにすべて 32 に設定しました。

共通事項

両システムで、limits.conf を使用して開けるファイルの最大数を 64,000 に増やしました。

テスト方法

テスト設計

テストの目的は、他の NoSQL ベンチマークでよく使われるものと同様の評価軸を使いながら、各データベースがさまざまな条件下でどのように実行されたかを確認することでした。

事前に小規模なテストを行い、 32 と 192 の間のスレッド数において同様の結果が得られることを確認していますが、特に 128 スレッドの時に性能が最も安定していました。したがって、タイムアウト例外を防ぐために 32 スレッドを使用する Cassandra ロードを除いて、すべてのテストでスレッド数は 128 に設定しました。その後の調査により、Cassandra においては、スレッド数は可変なものの、少ないスレッド数のときに最高のパフォーマンスとなるのが一般的な傾向であると判明しました。

データセットとしては、ノードあたり 400 万レコードの小さなデータセットと、ノードあたり 1,200 万レコードの大きなデータセットの 2 種類を使用しました。各レコードは 10 個の 100 バイト文字列(1 レコードあたり 1K バイト) で構成されます。したがって、ノードごとのデータベースサイズはそれぞれ 4GB または 12GB になります。小さなデータセットはメモリ内に完全に収まることができますが、大きなデータセットの 50%はローカルのメモリからストレージにフラッシュする必要があります。

トランザクション・ワークロードは、クライアントごとに 1,000 万回の操作を実行し、データセット全体にアクセスするものです。Cassandra の場合、この設定により、JVM のウォーミングアップに十分な時間を確保することができ、行がキャッシュ内外に確実に流れるようにしました。

テストの実行方法

リソースが共有されたパブリッククラウド上でベンチマークを実行する場合、実行時毎のパフォーマンスにばらつきが出ます。このベンチマークの評価対象は Azure ではなく、GridDB と Cassandra です。Azure によるパフォーマンスの変動を最小限に抑えるため、各テストは異なるリソースグループで 3 回実行し、それらのうちの最高値を結果として採用しています。

すべてのインスタンスが「割り当て解除」されている状態から開始して、ヘッドノードが最初に必要な数のインスタンスを起動します。実行後、構成ファイルを配置し、ローカル SSD をマウントし、既存のデータベースデータファイルを削除し、最後に initscript を使用して GridDB または Cassandra を起動します。

サーバの起動が完了すると、GridDB の gs_stat または Cassandra の nodetool を介してサーバ統計情報を取得し、詳細な分析のために保存しました。

YCSB のロードは、適切な insertstart、insertcount、recordcount パラメータを使用して、すべてのクライアント・ノード上で並行して実行します。ロードが完了した後、ワークロードは、YCSB が推奨する次の順序で実行されます
（https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads を参照）。

- Workload A -- 頻繁に更新
- Workload B -- 主に読み取り

- Workload C -- 読み取り専用
- Workload F -- 読み取り、変更、書き込み
- Workload D -- 最新の読み取り

各ワークロードが終了すると、サーバの統計情報を再度取得します。

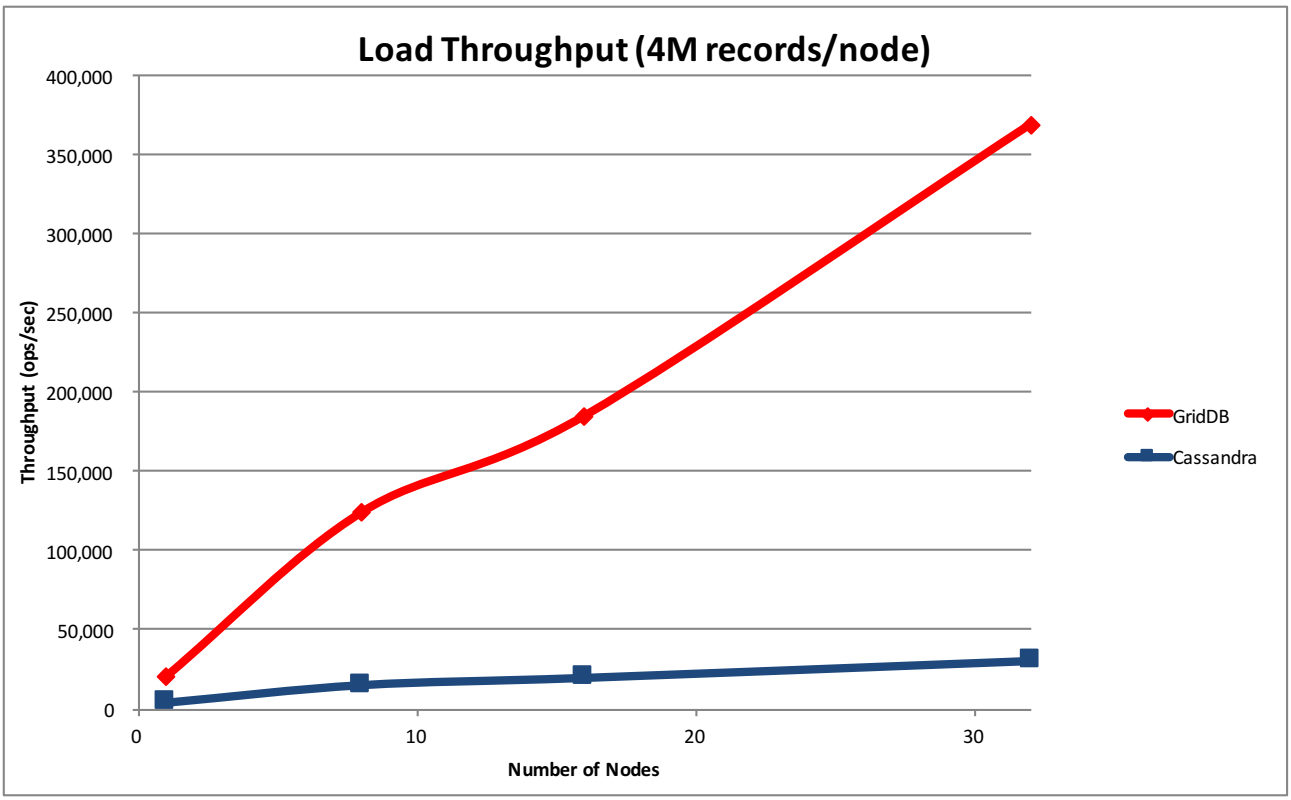
結果の収集と集計

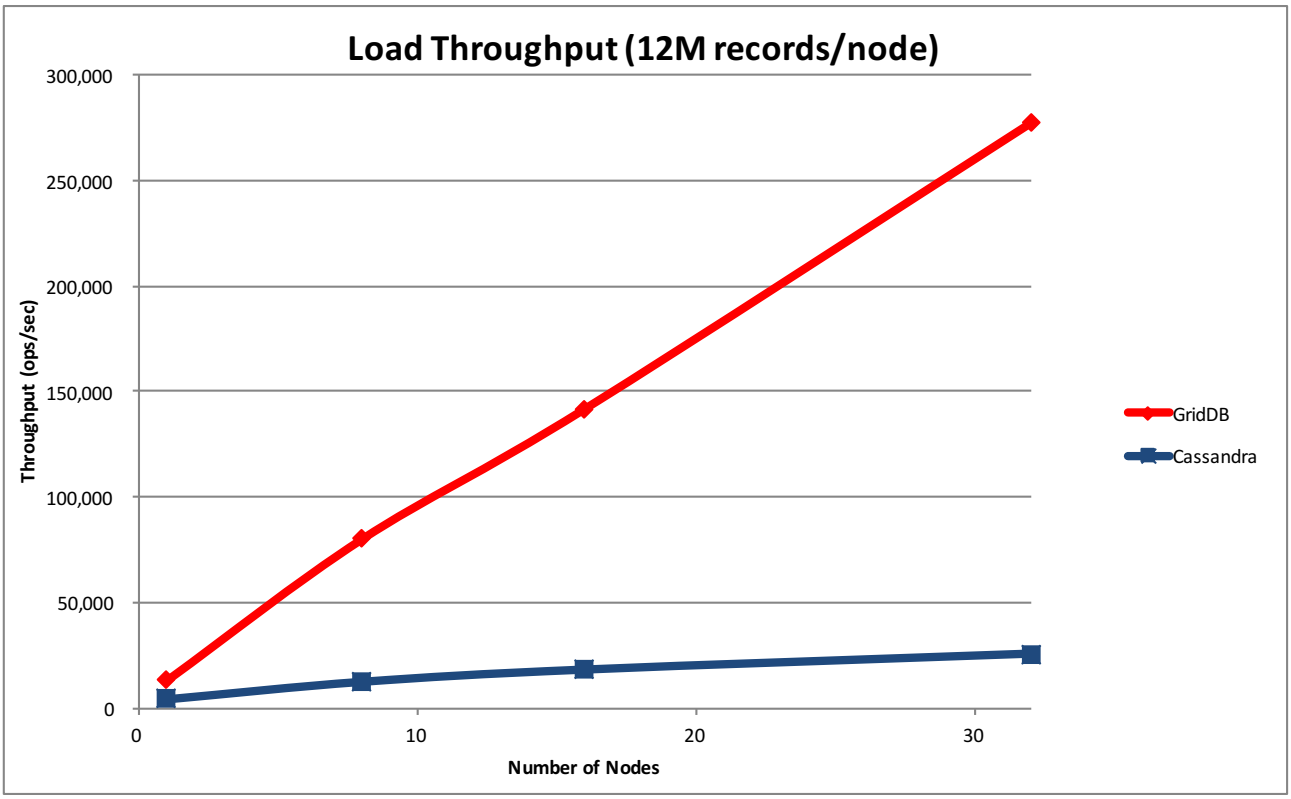
すべての YCSB 出力を後処理して結果を集計するために、awk と grep を使用した単純な bash スクリプトを使用して、CSV 形式の行ごとに 1 回のテスト実行を出力し、さらにスプレッドシートで処理しました。

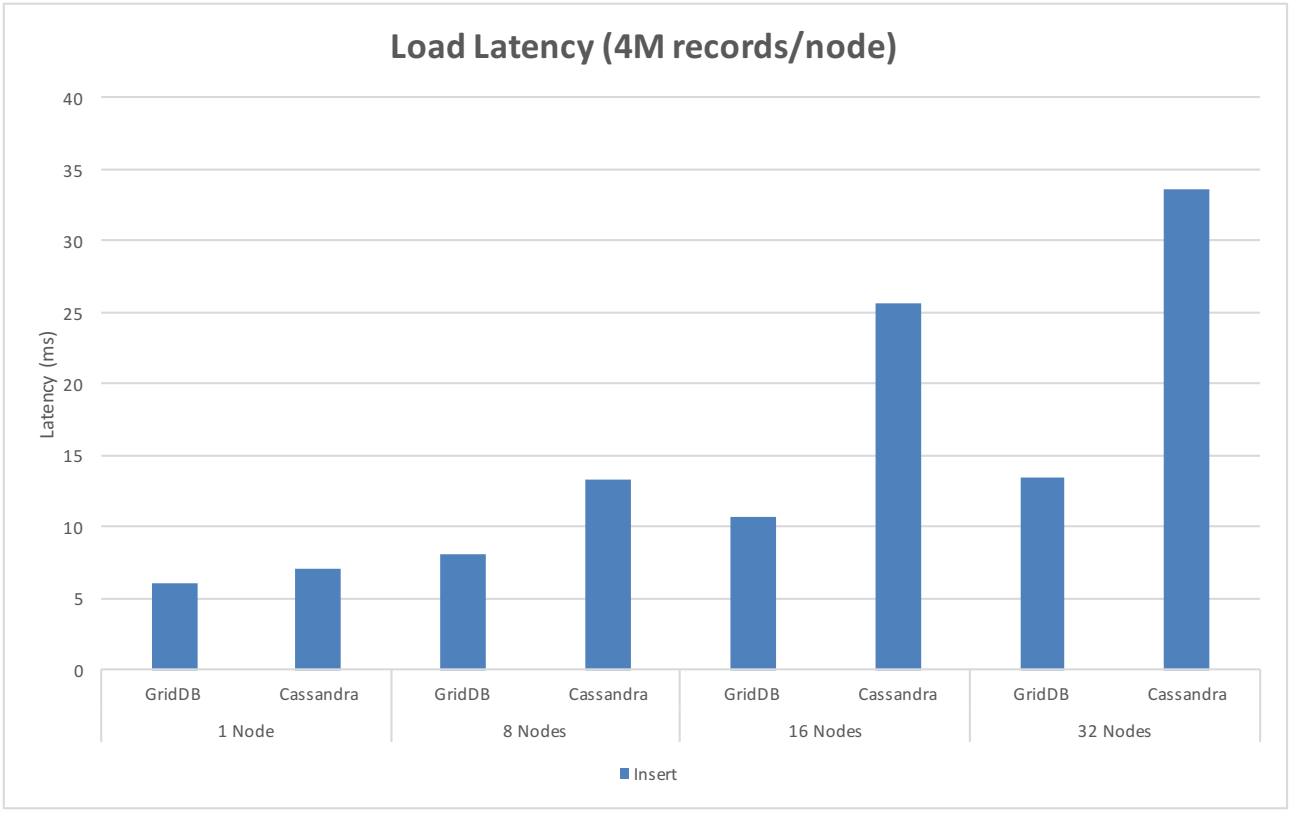
ベンチマークの結果

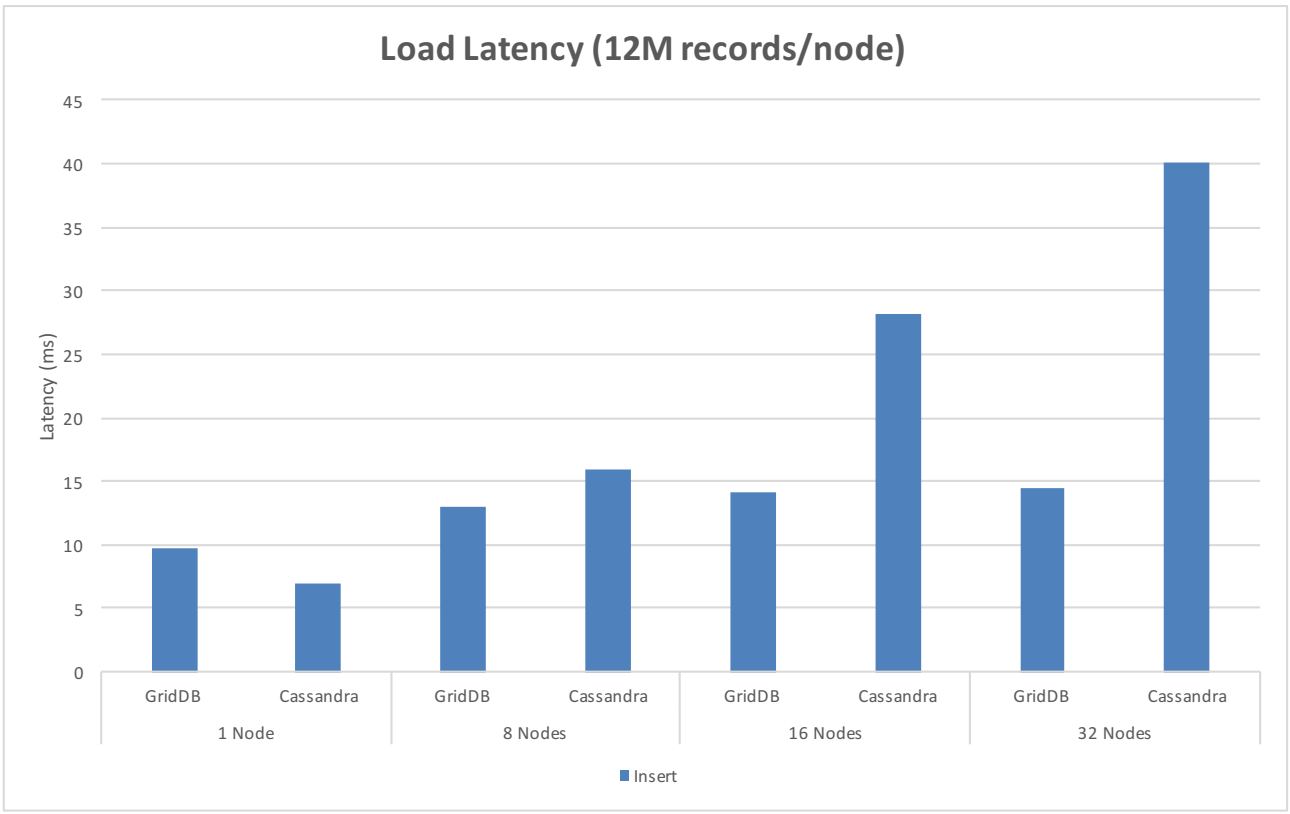
Load

他の Cassandra ベンチマークレポートにおいて Cassandra にデータを読み込む際に問題が発生すると報告されていますが、フィックスターズのベンチマークでも同様の問題が発生しました。同時書き込み数を推奨設定から増やし、タイムアウトを 10 倍に延長し、スレッド数を 32 に減らすと、すべての TimeoutExceptions が修正されました。スループットのグラフにおいては数値の高い方が、レイテンシのグラフにおいては数値の低い方が良い結果を意味します。


Load Throughput (4M records/node)
Throughput (ops/sec)
400,000
350,000
300,000
250,000
200,000
150,000
100,000
50,000
0
0
10
20
30
Number of Nodes
GridDB
Cassandra

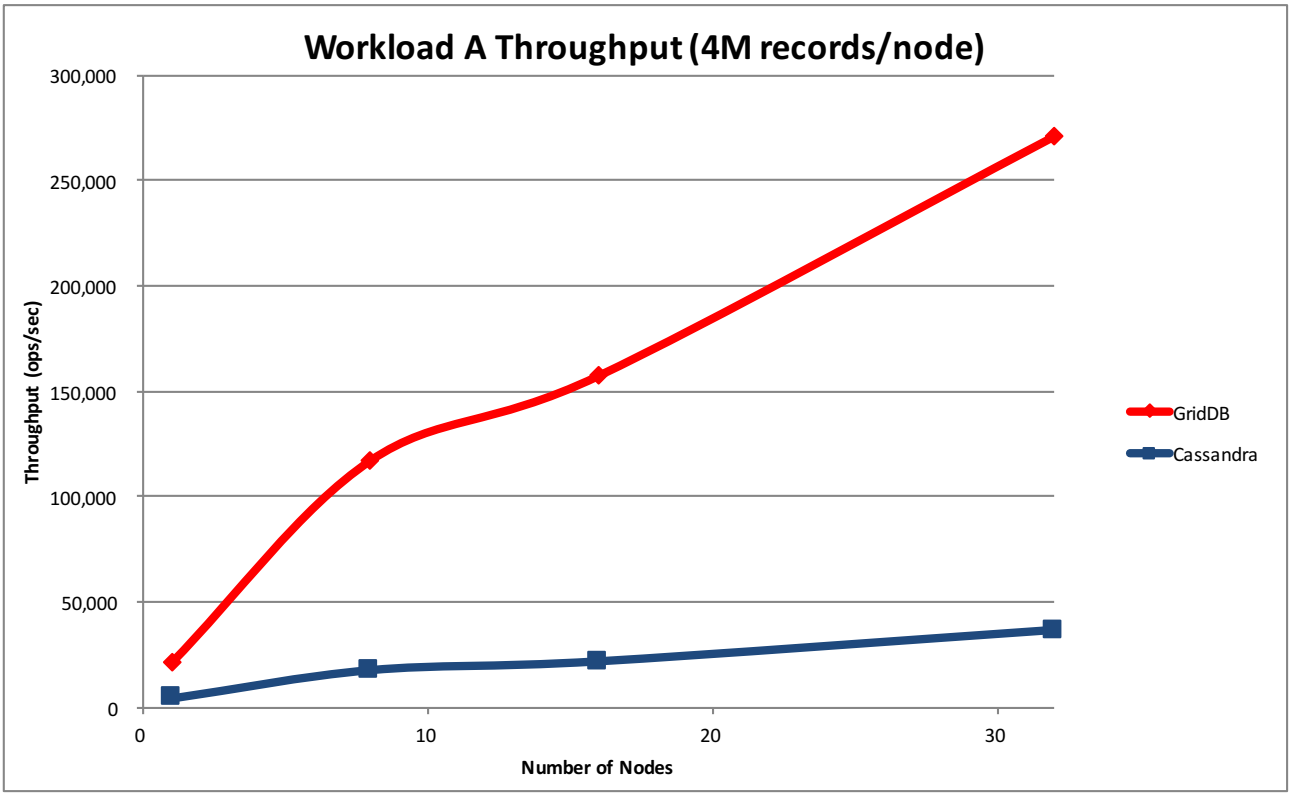

Load Throughput (12M records/node)
Throughput (ops/sec)
300,000
250,000
200,000
150,000
100,000
50,000
0
0
10
20
30
Number of Nodes
GridDB
Cassandra

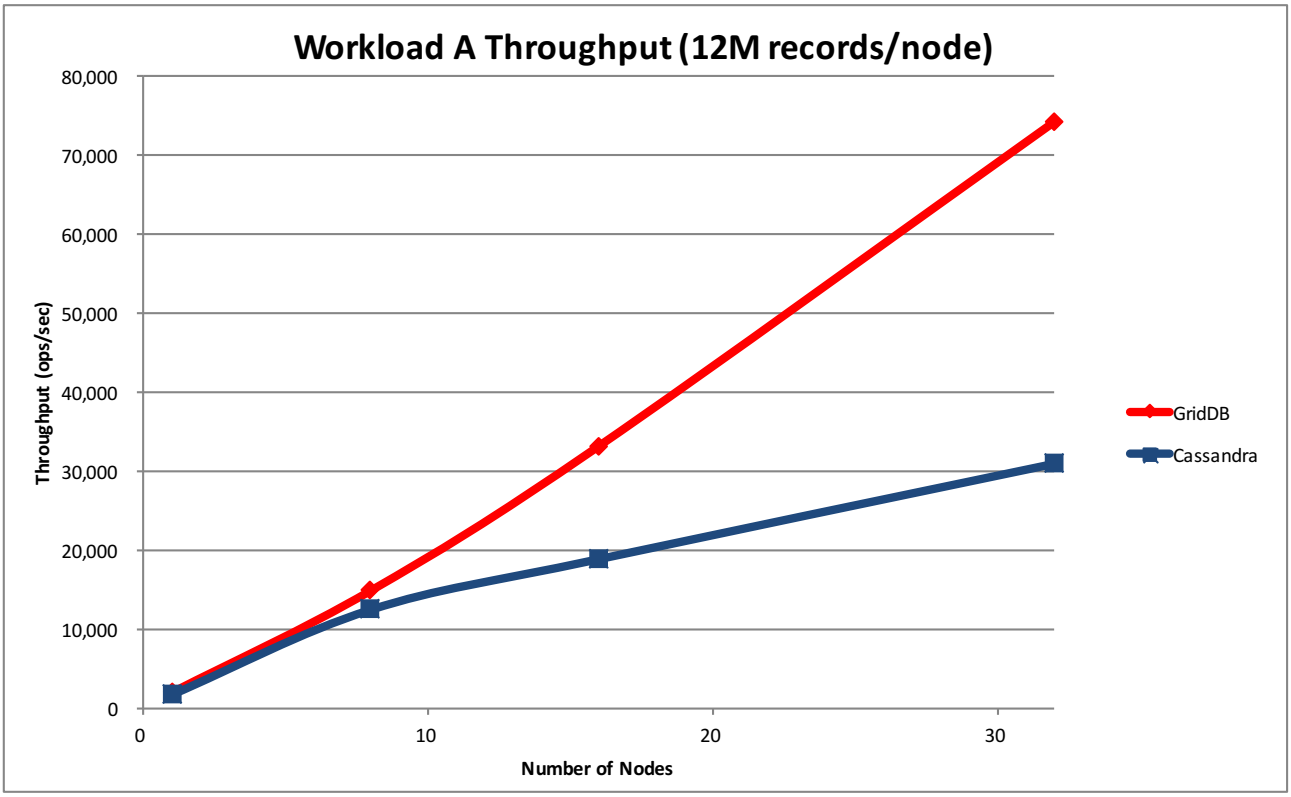

Load Latency (4M records/node)
Latency (ms)
40
35
30
25
20
15
10
5
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Insert

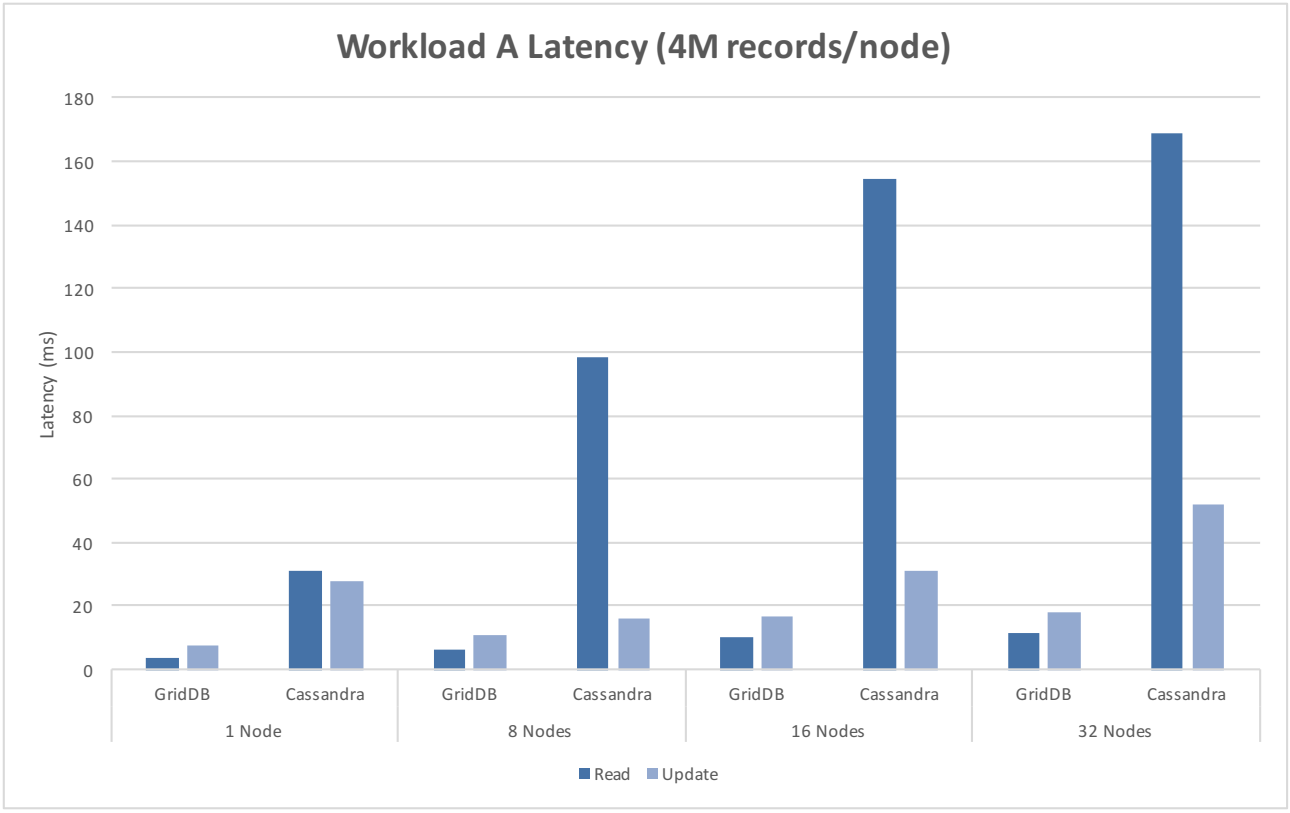

Load Latency (12M records/node)
Latency (ms)
45
40
35
30
25
20
15
10
5
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Insert

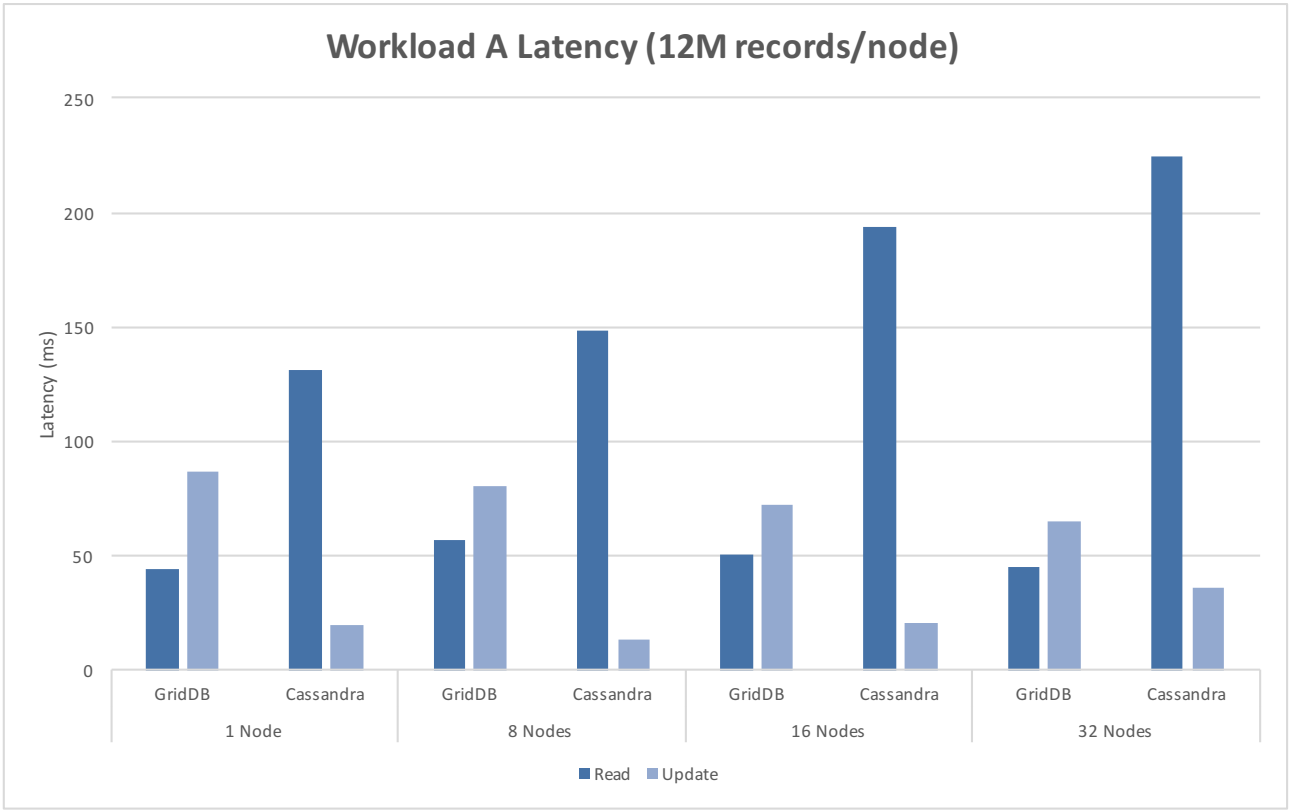
Workload A

Workload A は更新(update)が集中的に行われるワークロードです。スループットのグラフでは数値の高い方が、レイテンシのグラフにおいては数値の低い方が良い結果を意味します。



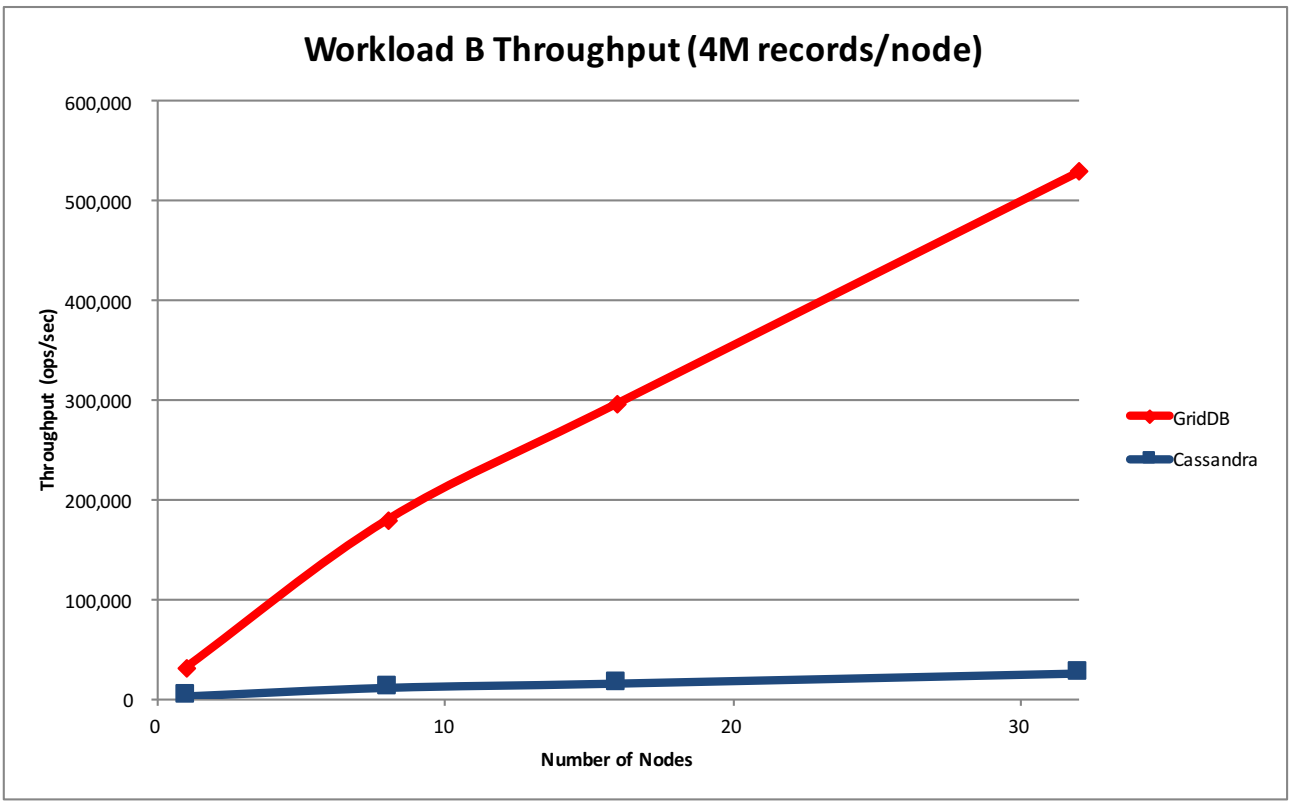


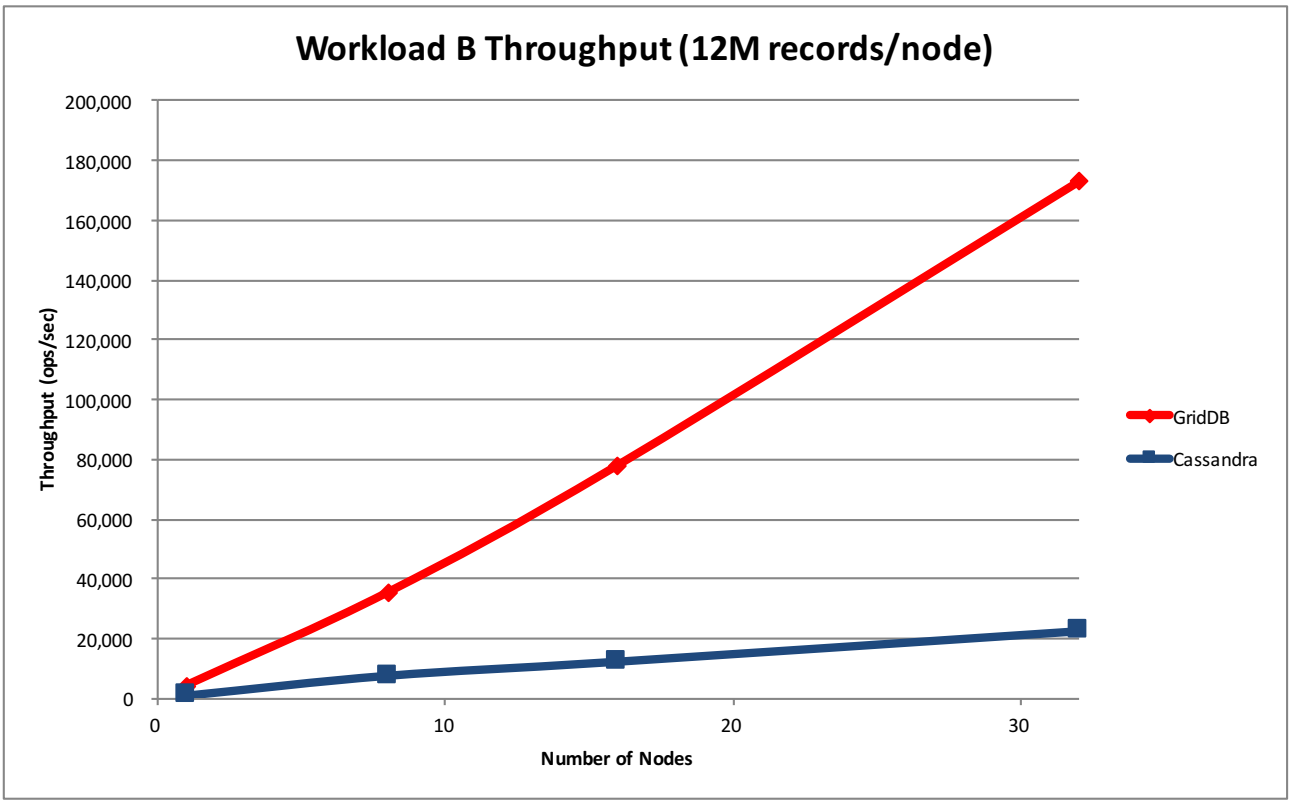

Workload A Latency (4M records/node)
Latency (ms)
180
160
140
120
100
80
60
40
20
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read
Update

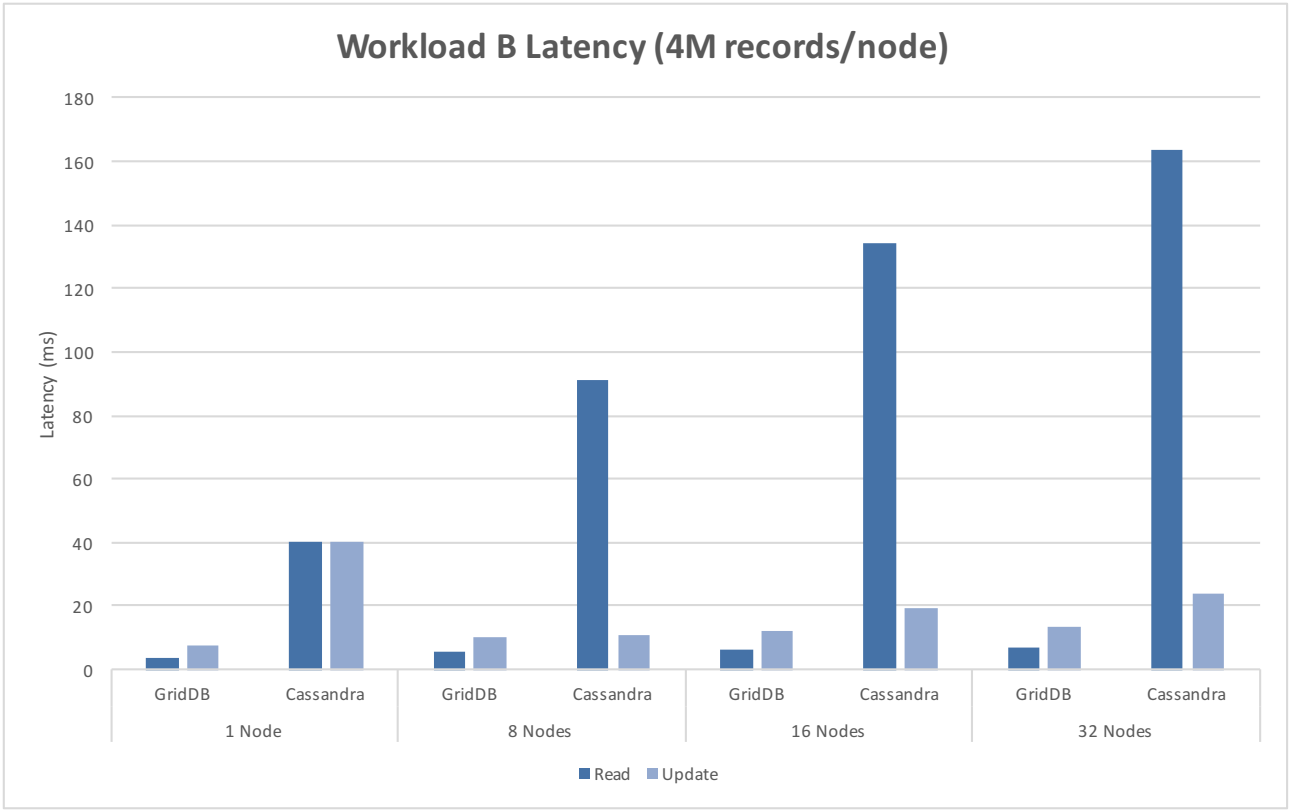

Workload A Latency (12M records/node)
Latency (ms)
250
200
150
100
50
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read
Update

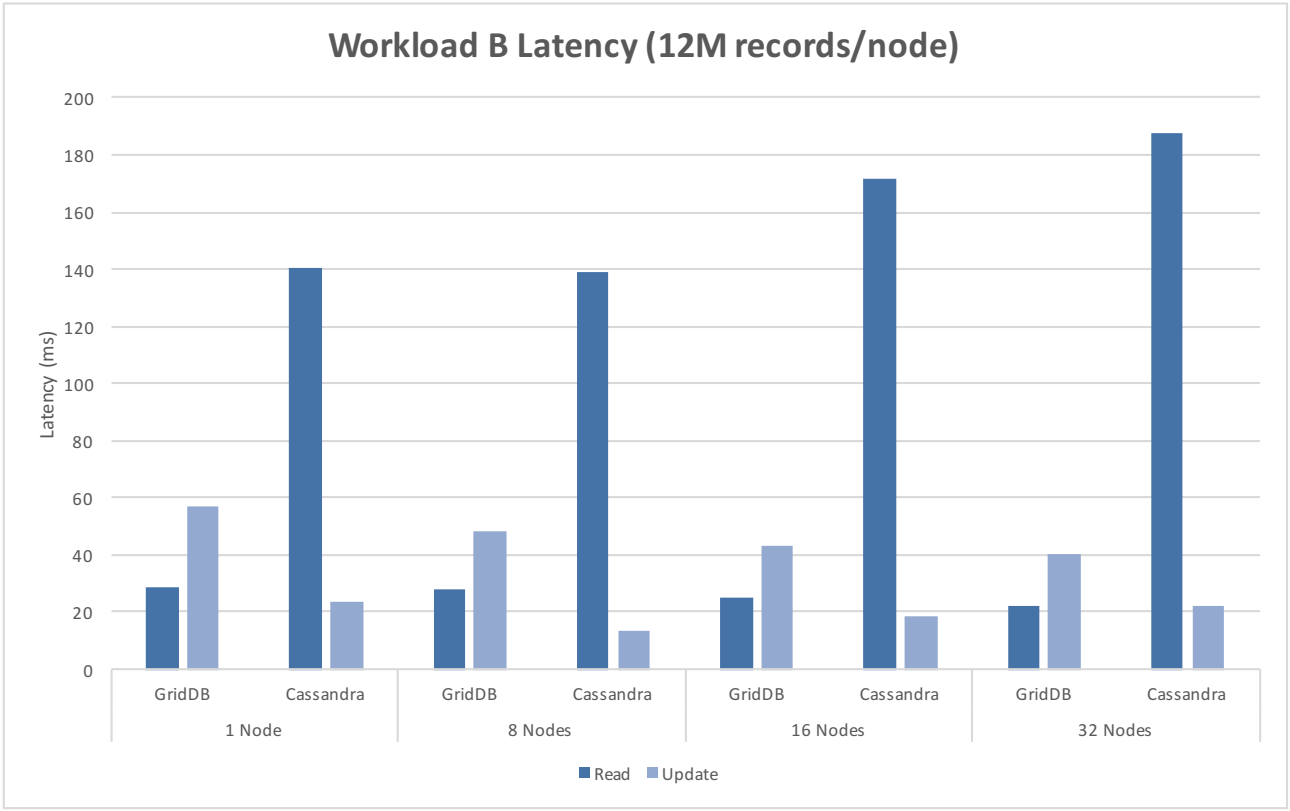
Workload B

Workload B では、95%の読み取り操作と 5%の書き込み操作を行います。スループットのグラフでは数値の高い方が、レイテンシのグラフにおいては数値の低い方が良い結果を意味します。



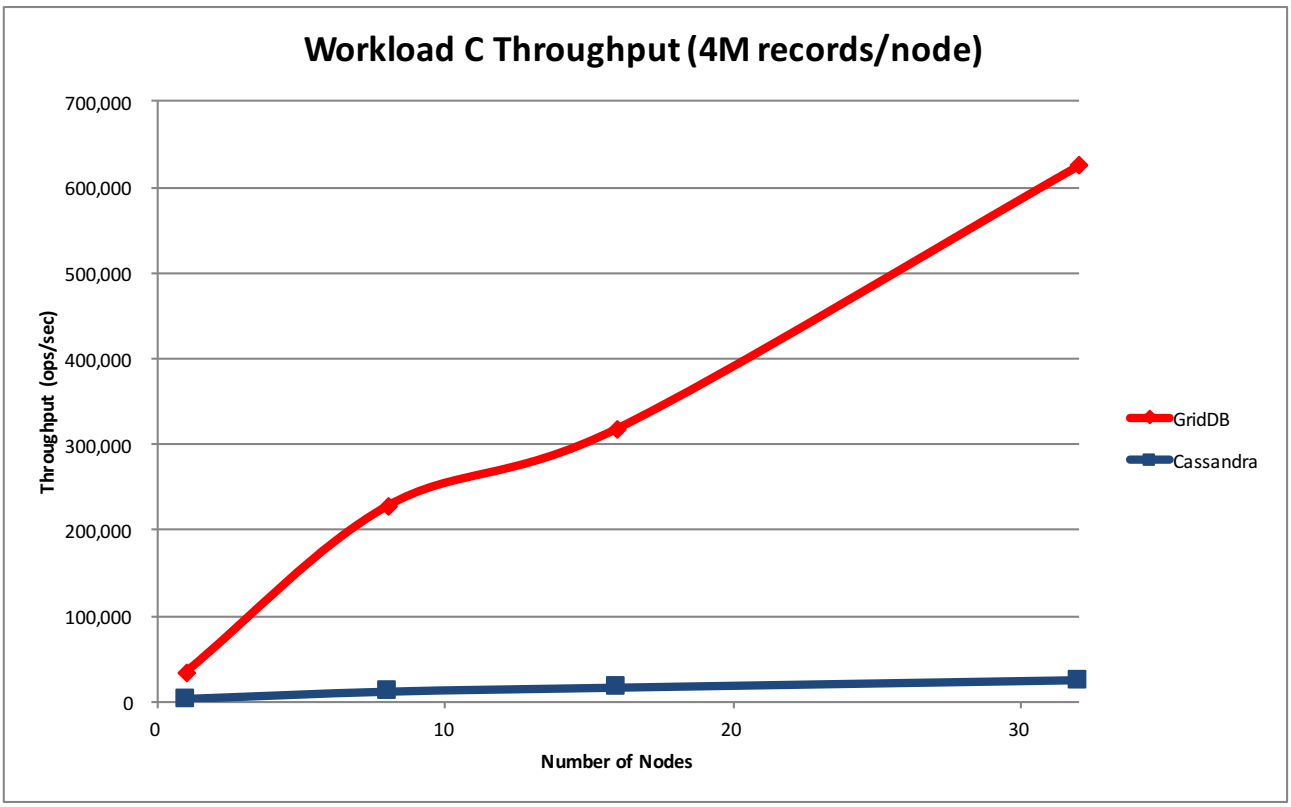


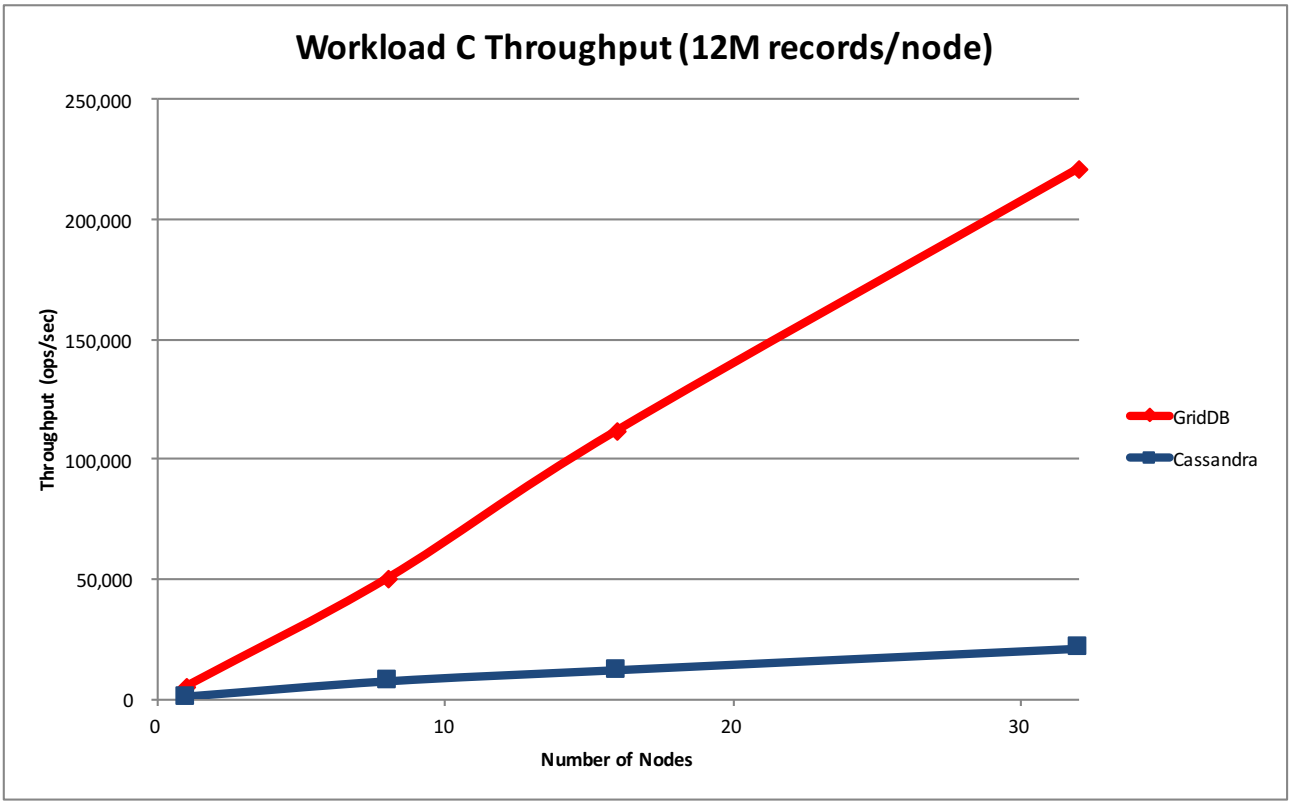

Workload B Latency (4M records/node)
Latency (ms)
180
160
140
120
100
80
60
40
20
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read
Update

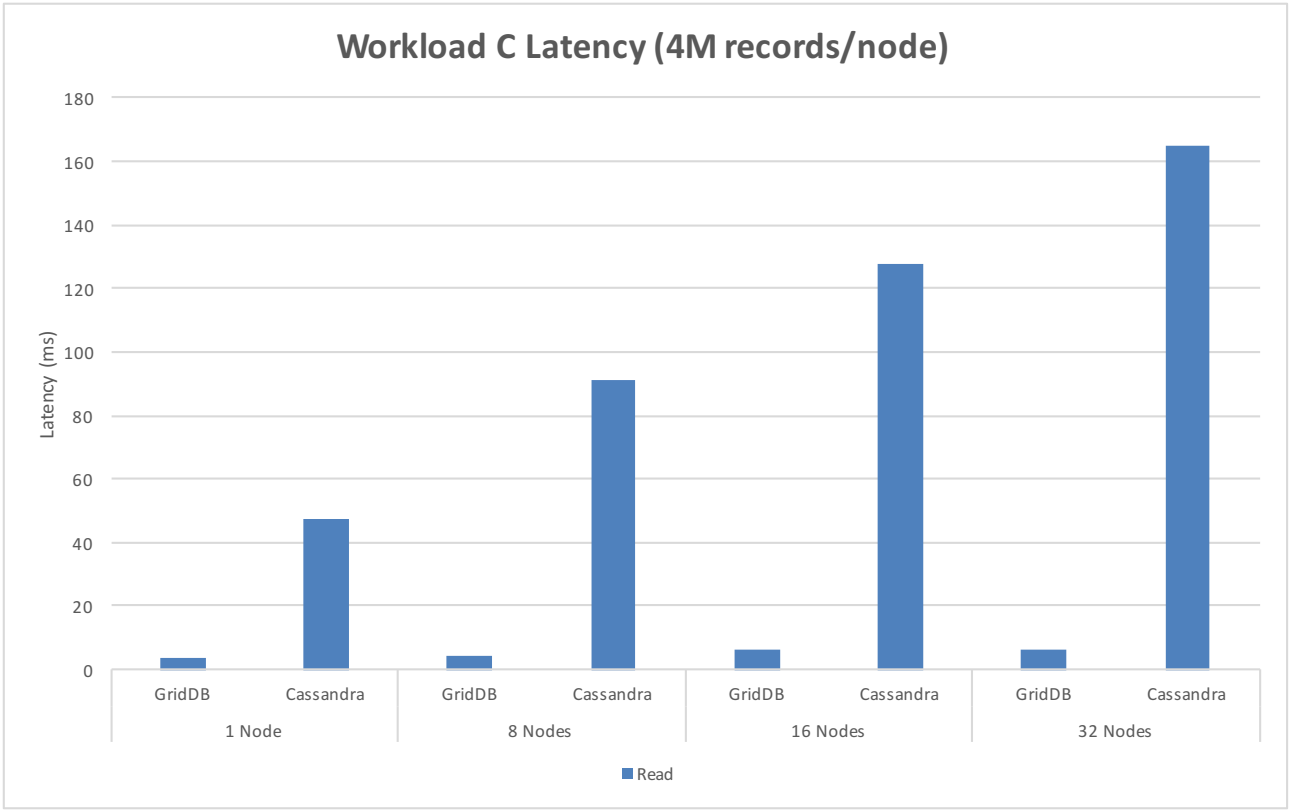

Workload B Latency (12M records/node)
Latency (ms)
200
180
160
140
120
100
80
60
40
20
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read
Update

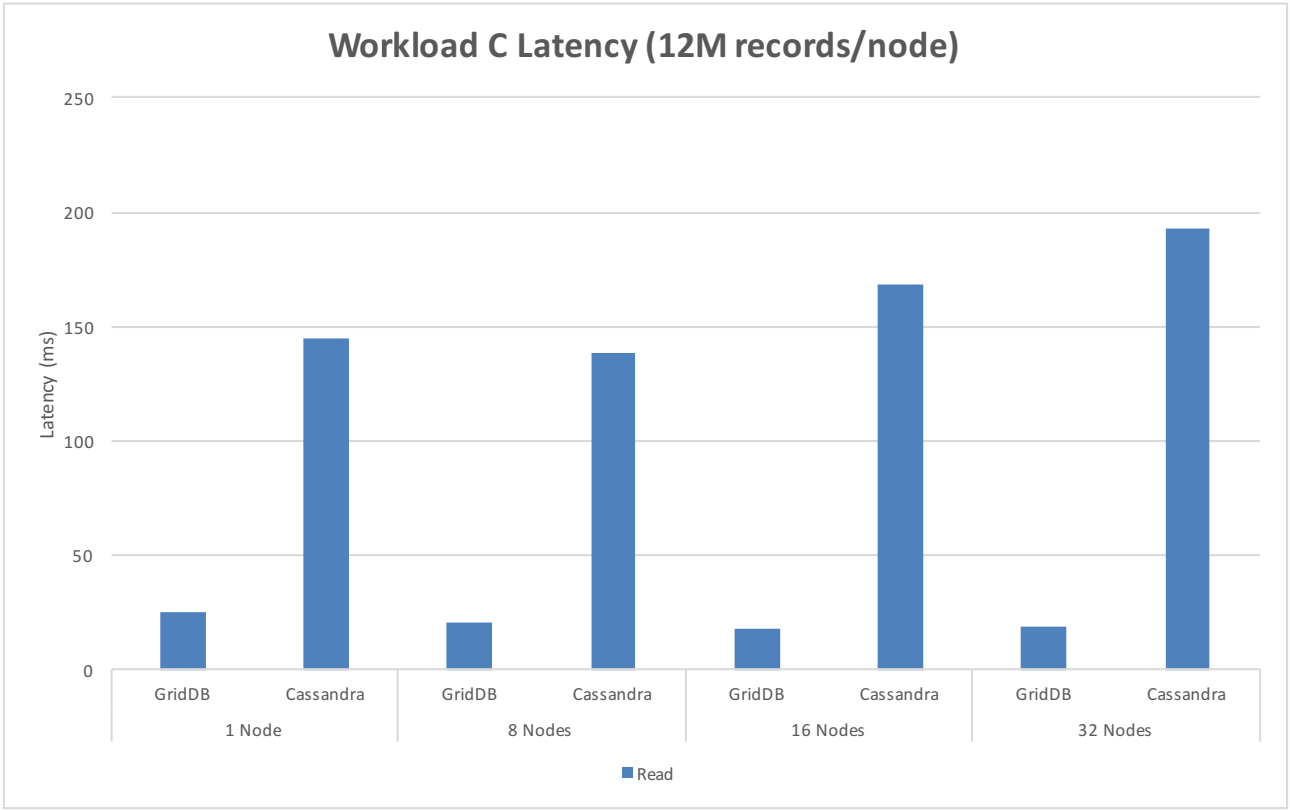
Workload C

Workload C は読み取り操作のみです。スループットのグラフでは数値の高い方が、レイテンシのグラフにおいては数値の低い方が良い結果を意味します。



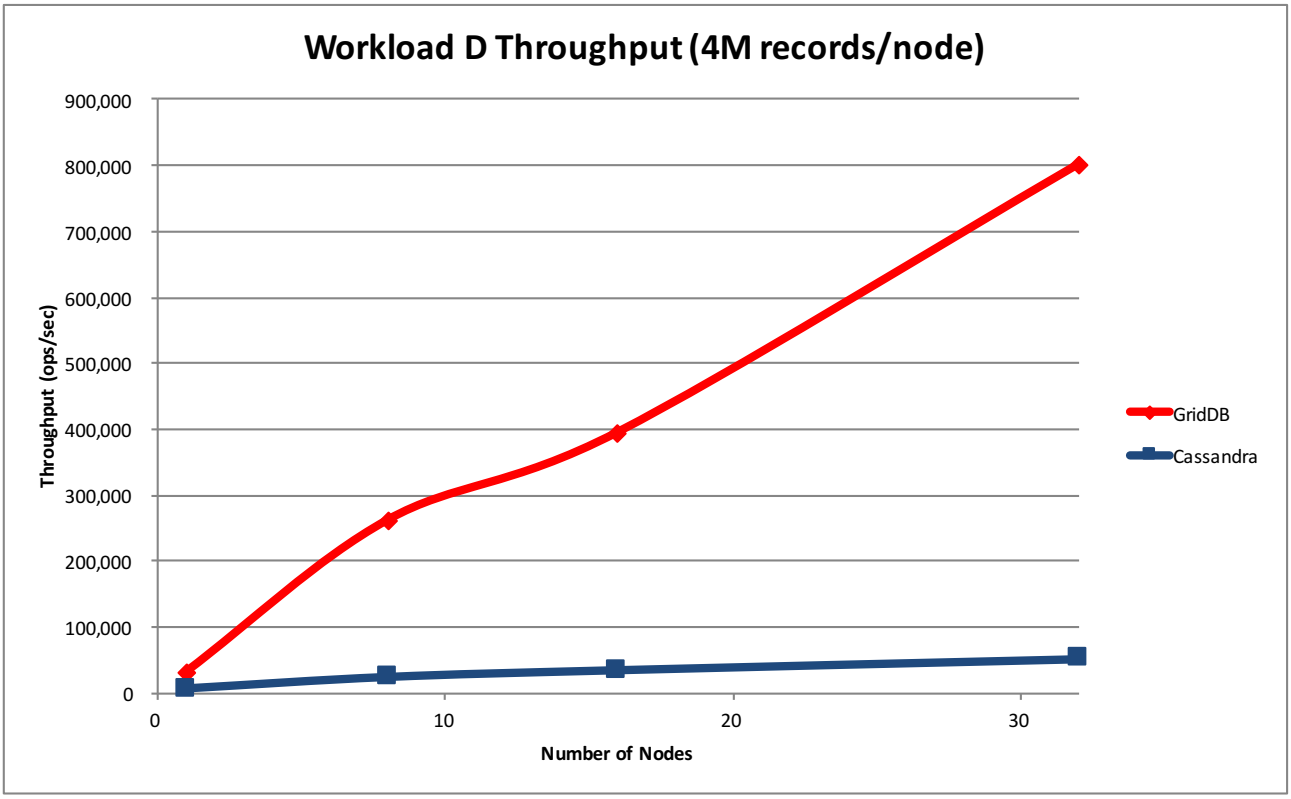


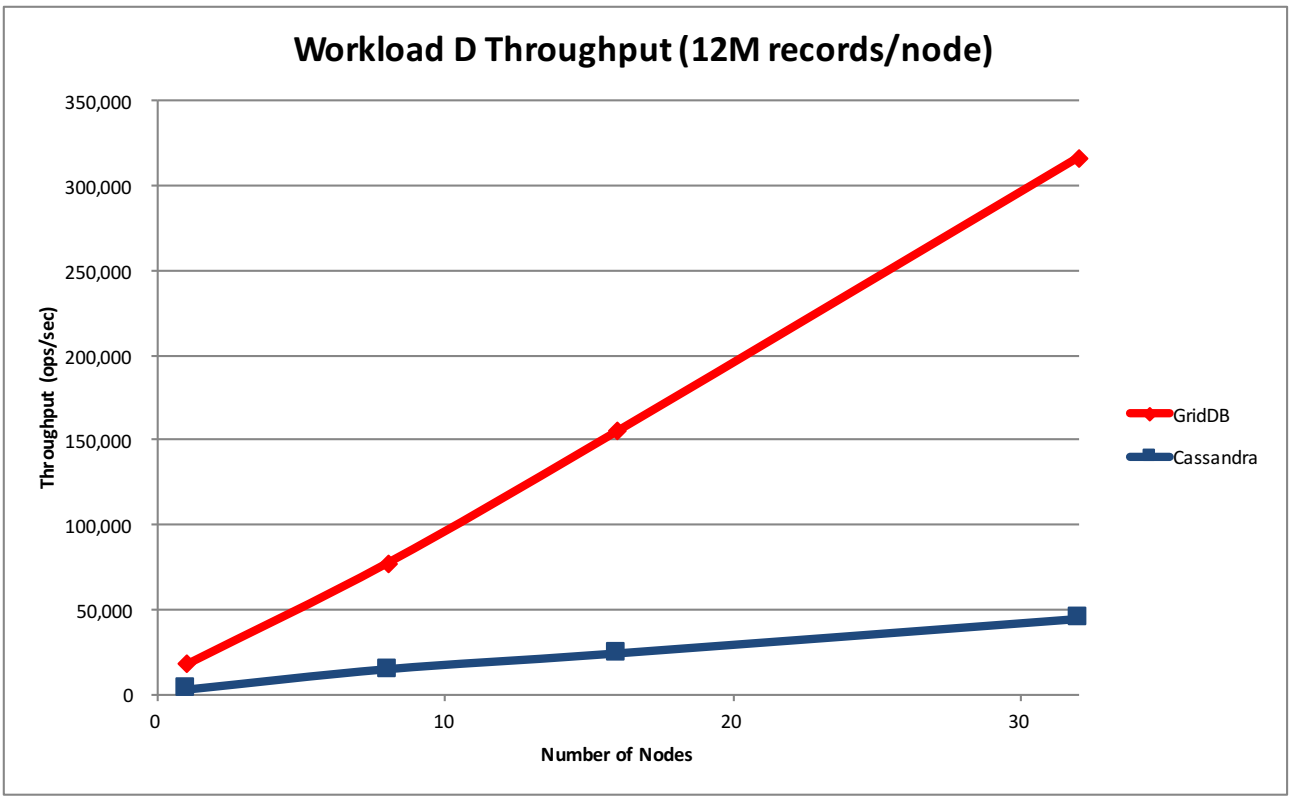

Workload C Latency (4M records/node)
Latency (ms)
180
160
140
120
100
80
60
40
20
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read

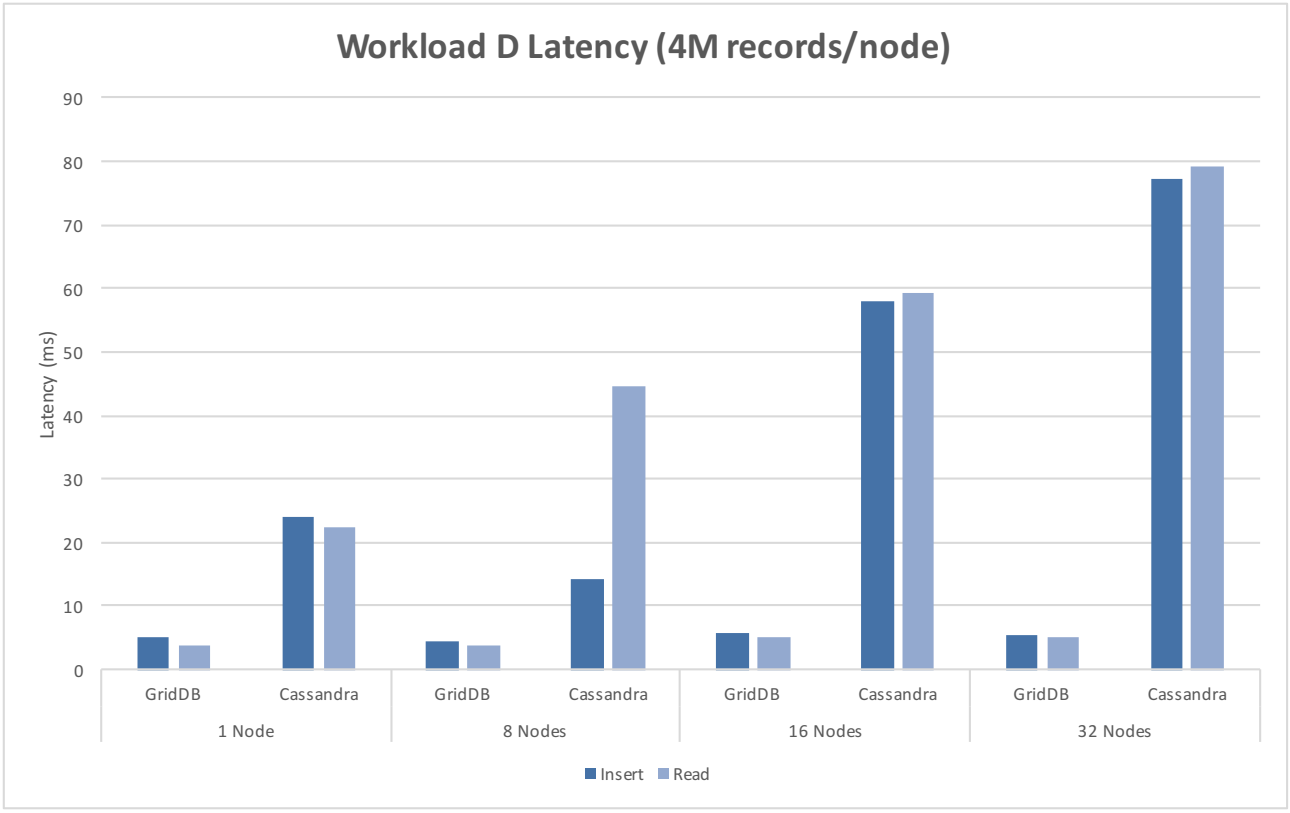

Workload C Latency (12M records/node)
Latency (ms)
250
200
150
100
50
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read

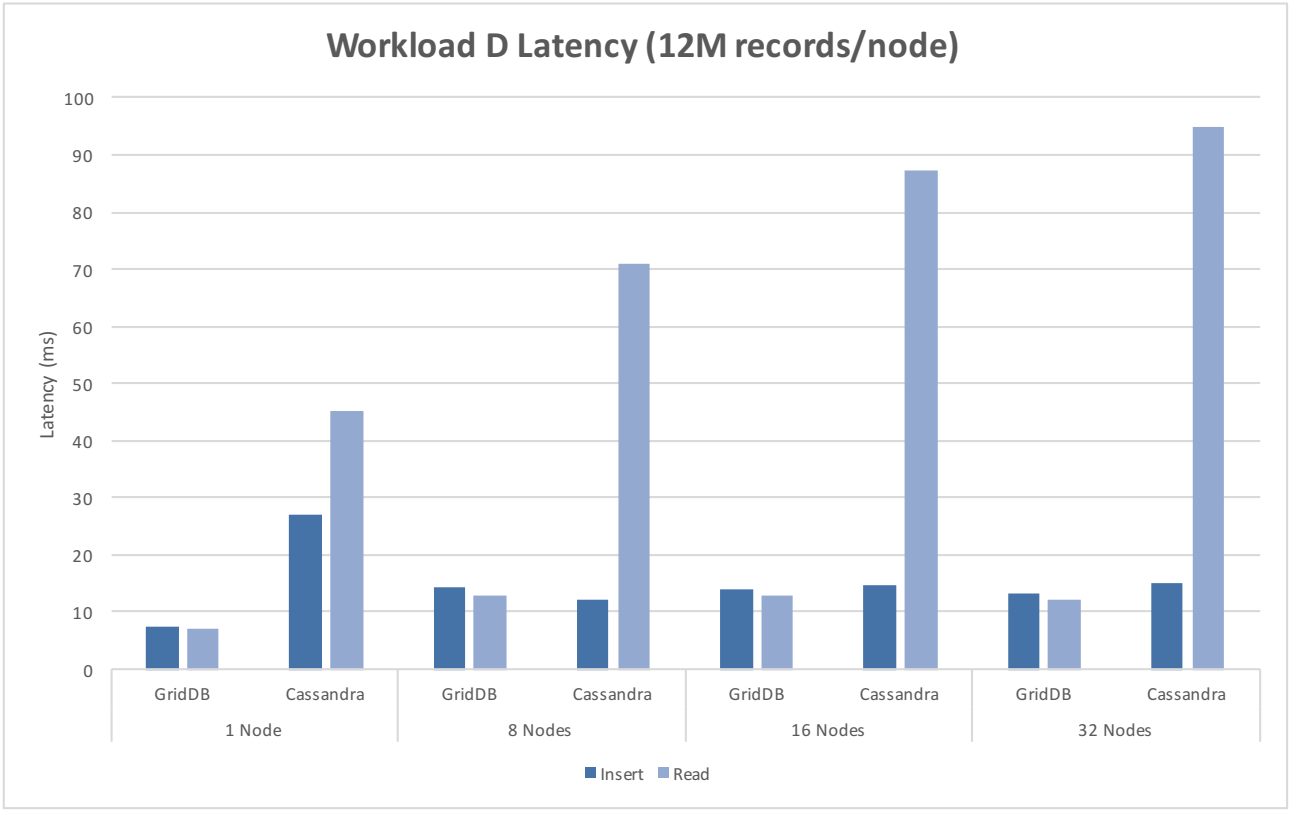
Workload D

Workload D では新しいレコードを挿入し、それらの新しいレコードを読み取ります。スループットのグラフでは数値の高い方が、レイテンシのグラフにおいては数値の低い方が良い結果を意味します。



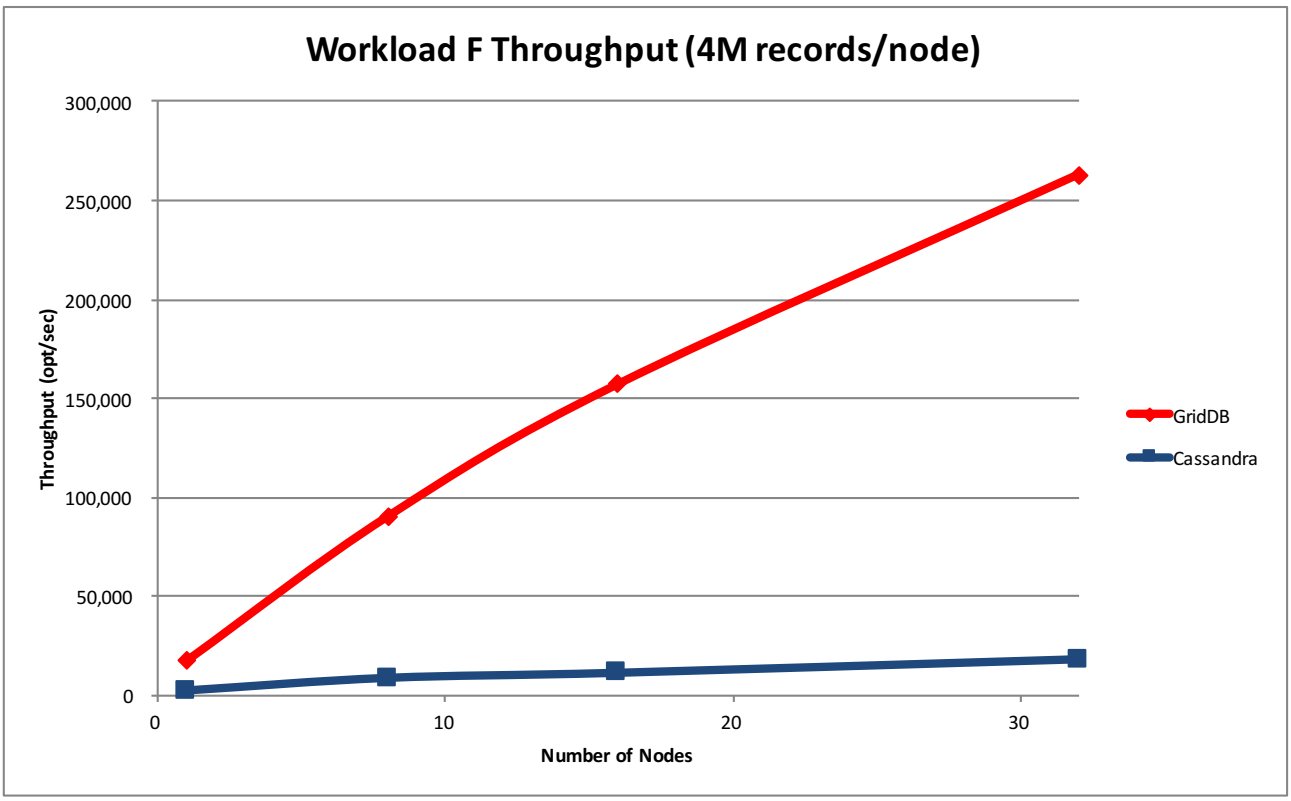


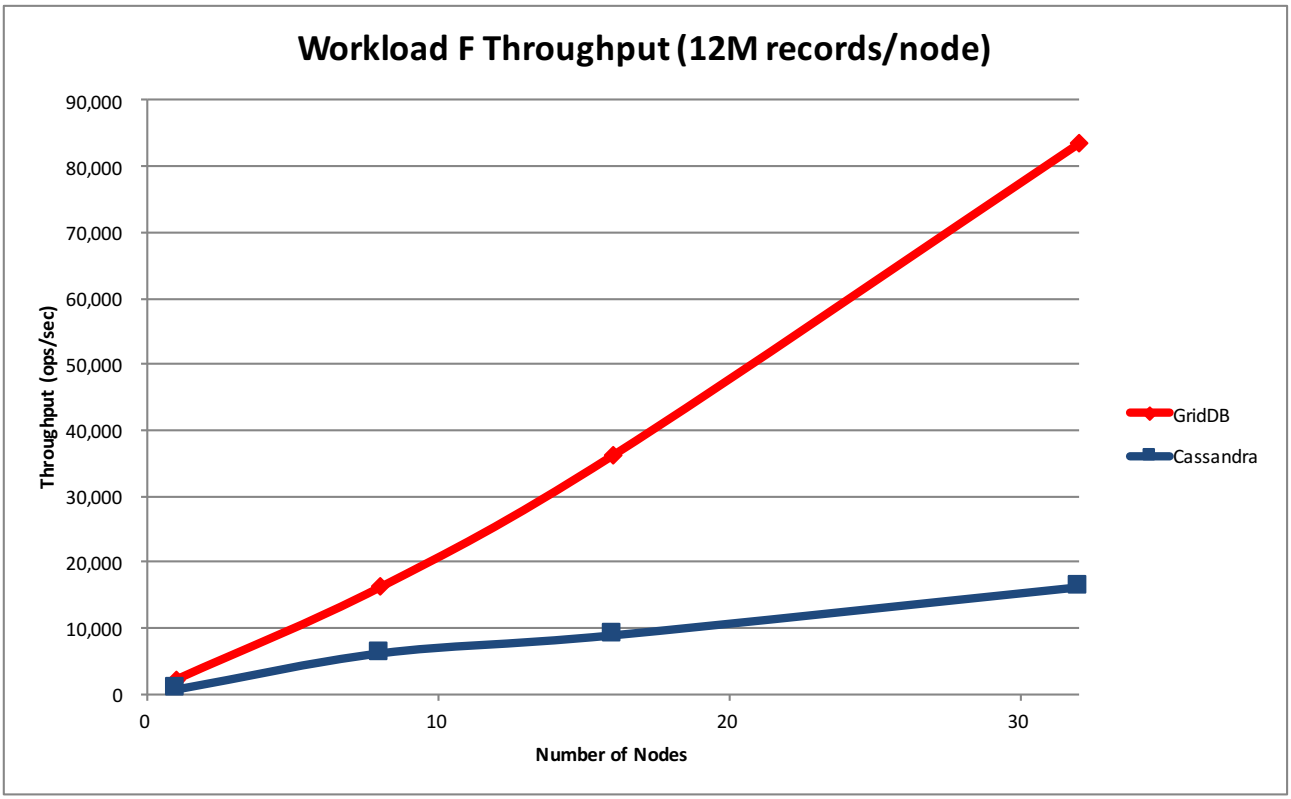

Workload D Latency (4M records/node)
Latency (ms)
90
80
70
60
50
40
30
20
10
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Insert
Read

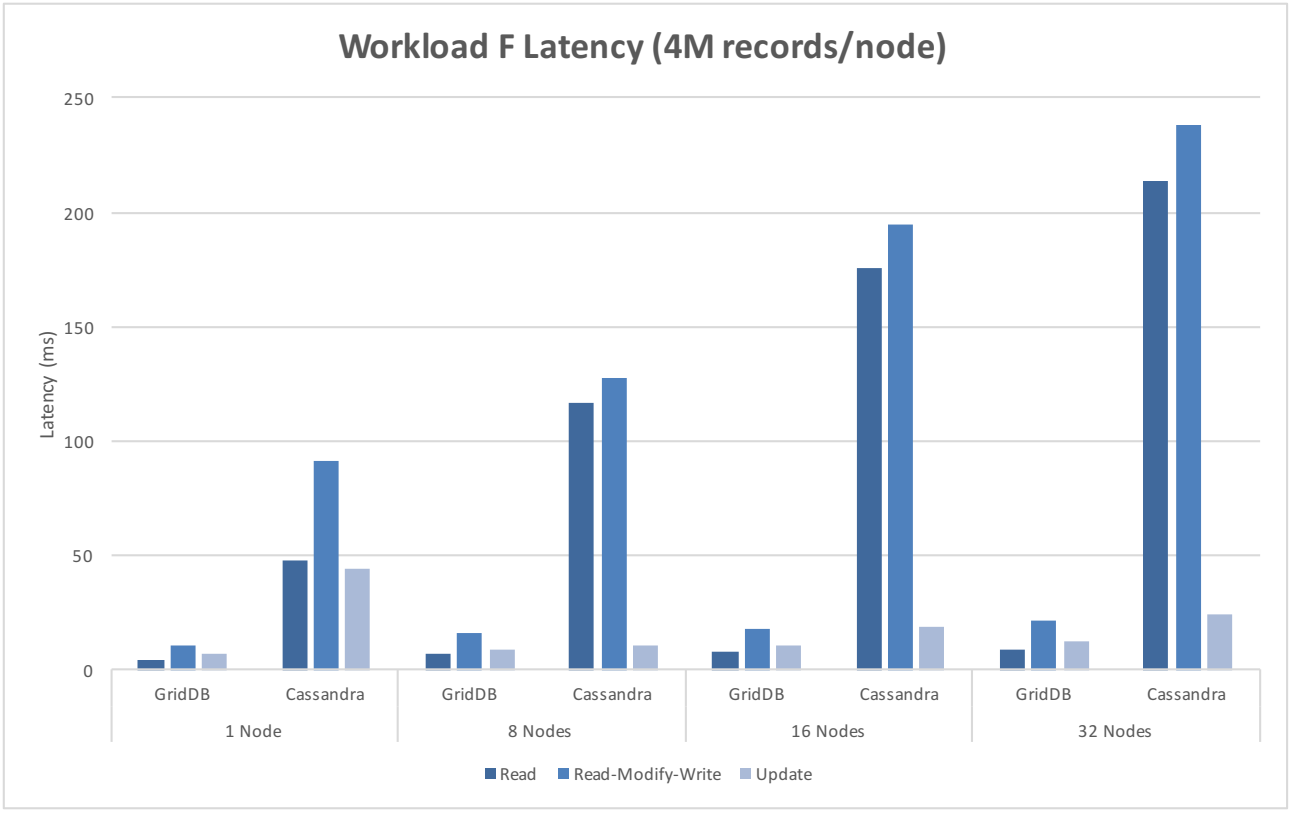

Workload D Latency (12M records/node)
Latency (ms)
100
90
80
70
60
50
40
30
20
10
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Insert
Read

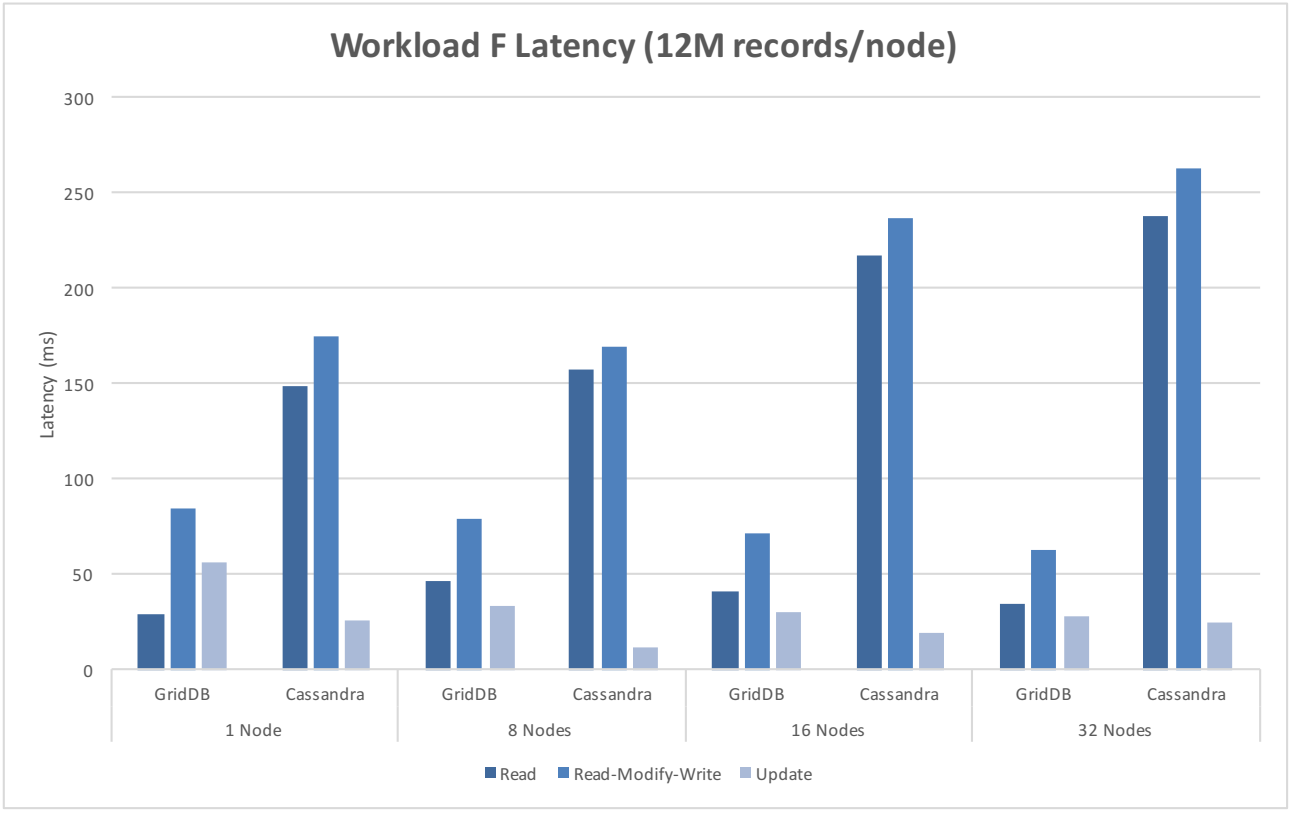
Workload F

Workload F ではレコードを読み取り、変更してから書き戻します。スループットのグラフでは数値の高い方が、レイテンシのグラフにおいては数値の低い方が良い結果を意味します。





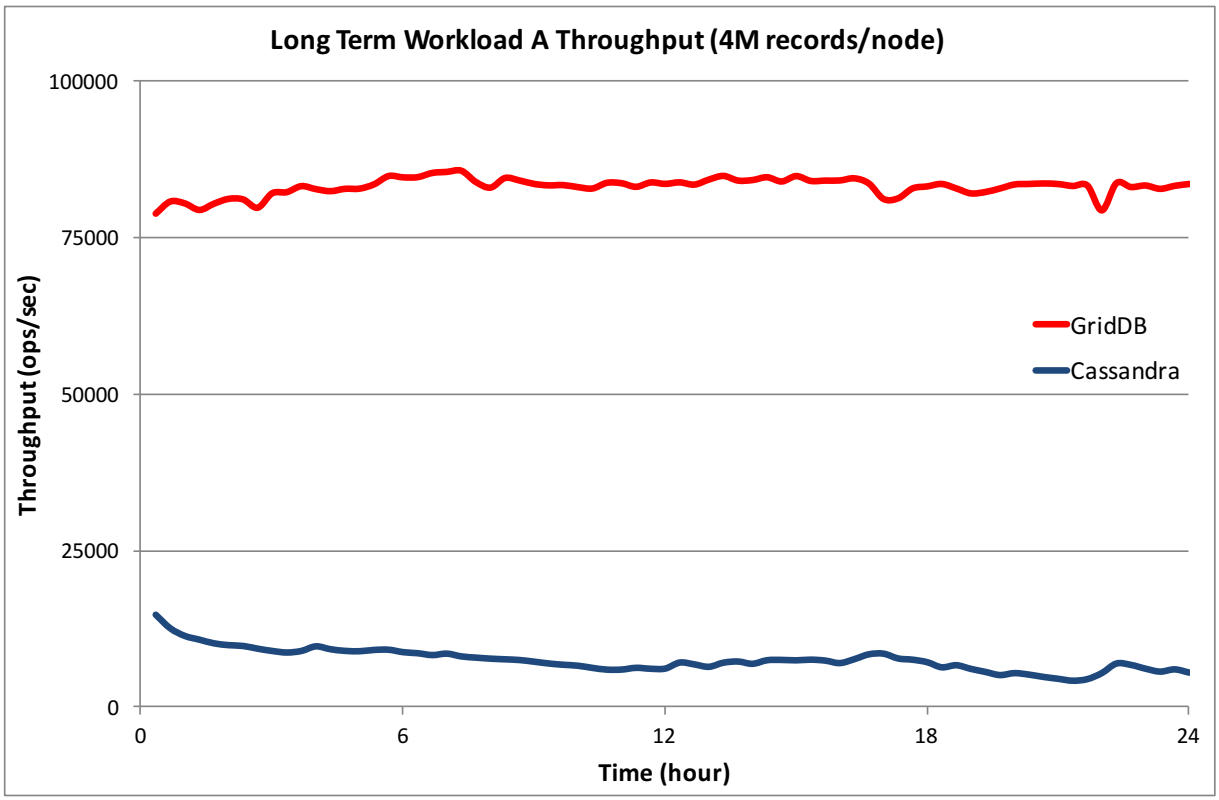

Workload F Latency (4M records/node)
Latency (ms)
250
200
150
100
50
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read
Read-Modify-Write
Update

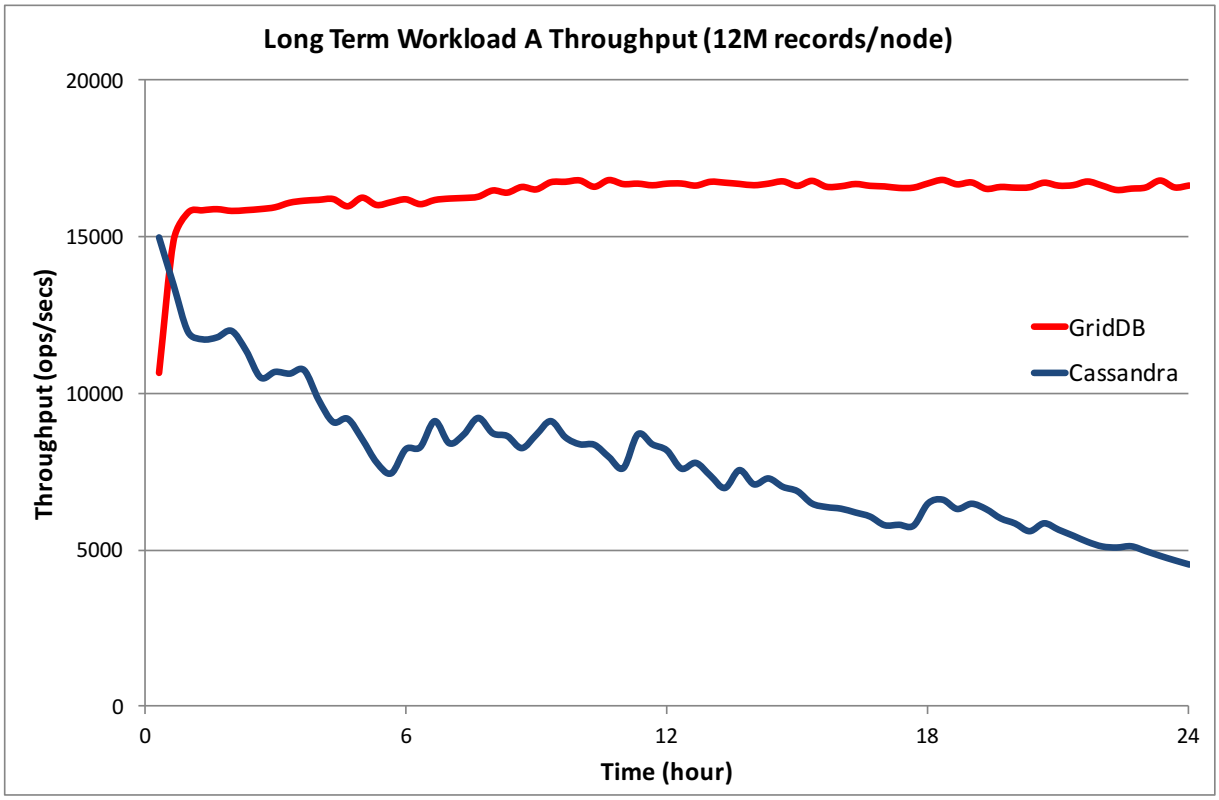

Workload F Latency (12M records/node)
Latency (ms)
300
250
200
150
100
50
0
GridDB
Cassandra
1 Node
8 Nodes
16 Nodes
32 Nodes
Read
Read-Modify-Write
Update

長期的な Workload A

Workload A などの更新集中型ワークロードでは、Cassandra はログベースのアーキテクチャで行をすばやく削除済みとしてマークし、ログの末尾に新しい値を追加できるため、Cassandra の結果は非常に良好でした。しかし我々は、Cassandra は時間が経つにつれてパフォーマンスが落ちることに気づきました。 そこで 8 ノードのクラスタを構成し、ノードごとに 400 万と 1,200 万のレコードをロードし、operationcount を 2^{32-1} に設定し、テストを 24 時間実行しました。

大きなデータセットでの結果の方がより傾向が明確に出ていますが、どちらのテストにおいても、Cassandra のスループットは、初期と比較して 24 時間後のスループットは 50% 未満となりました。 一方、GridDB のパフォーマンスは、データサイズの大小にかかわらず、非常に安定していました。


Long Term Workload A Throughput (4M records/node)
100000
75000
50000
25000
0
Throughput (ops/sec)
0
6
12
18
24
Time (hour)
GridDB
Cassandra


Long Term Workload A Throughput (12M records/node)
20000
15000
10000
5000
0
Throughput (ops/secs)
0
6
12
18
24
Time (hour)
GridDB
Cassandra

結果一覧

スループットの値は「1 秒あたりの操作数 (ops/sec)」単位、レイテンシの値は「ミリ秒 (msec)」単位です。

小さなデータセットでのスループット (ノードあたり 400 万レコード)

		1 Node	8 Nodes	16 Nodes	32 Nodes
Load	GridDB	20,425	123,859	184,836	369,046
	Cassandra	4,246	15,223	19,753	30,304
Workload A	GridDB	21,286	117,284	157,347	270,690
	Cassandra	4,330	17,656	21,781	36,496
Workload B	GridDB	31,449	179,842	296,967	529,091
	Cassandra	3,171	11,657	15,865	25,832
Workload C	GridDB	33,796	227,802	318,485	624,954
	Cassandra	2,707	11,174	15,886	24,623
Workload D	GridDB	31,010	261,624	395,112	801,982
	Cassandra	5,672	23,654	34,246	51,389
Workload F	GridDB	17,300	90,310	157,144	262,940
	Cassandra	1,837	8,351	10,971	17,942

小さなデータセットでのレイテンシ (ノードあたり 400 万レコード) -- 1 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	6.0			
	Cassandra	7.0			
Workload A	GridDB		3.9		7.7
	Cassandra		30.9		28.1
Workload B	GridDB		3.8		7.6
	Cassandra		40.2		40.0
Workload C	GridDB		3.6		
	Cassandra		47.2		
Workload D	GridDB	5.1	3.9		
	Cassandra	24.1	22.4		
Workload F	GridDB		3.7	10.7	6.9
	Cassandra		47.5	91.6	44.1

小さなデータセットでのレイテンシ (ノードあたり 400 万レコード) -- 8 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	8.0			
	Cassandra	13.3			
Workload A	GridDB		6.2		11.0
	Cassandra		98.6		16.3
Workload B	GridDB		5.3		10.3
	Cassandra		91.3		10.6
Workload C	GridDB		4.4		
	Cassandra		91.0		
Workload D	GridDB	4.5	3.8		
	Cassandra	14.2	44.5		
Workload F	GridDB		6.7	15.7	9.0
	Cassandra		47.5	91.6	44.1

小さなデータセットでのレイテンシ (ノードあたり 400 万レコード) -- 16 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	10.7			
	Cassandra	25.7			
Workload A	GridDB		10.2		16.5
	Cassandra		154.7		30.9
Workload B	GridDB		6.4		12.3
	Cassandra		134.0		19.3
Workload C	GridDB		6.3		
	Cassandra		128.0		
Workload D	GridDB	5.7	5.0		
	Cassandra	57.9	59.3		
Workload F	GridDB		7.5	18.2	10.6
	Cassandra		176.0	194.5	18.6

小さなデータセットでのレイテンシ (ノードあたり 400 万レコード) -- 32 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	13.4			
	Cassandra	33.6			
Workload A	GridDB		11.3		18.0
	Cassandra		168.7		52.2
Workload B	GridDB		6.8		13.2
	Cassandra		163.8		23.7
Workload C	GridDB		6.2		
	Cassandra		164.7		
Workload D	GridDB	5.5	4.9		
	Cassandra	77.2	79.1		
Workload F	GridDB		8.7	21.3	12.6
	Cassandra		213.8	237.8	24.0

大きなデータセットでのスループット (ノードあたり 1,200 万レコード)

		1 Node	8 Nodes	16 Nodes	32 Nodes
Load	GridDB	13,082	80,074	141,847	277,243
	Cassandra	4,325	12,405	18,063	25,412
Workload A	GridDB	1,945	14,847	33,078	74,053
	Cassandra	1,699	12,485	18,892	30,973
Workload B	GridDB	4,233	35,419	78,117	173,166
	Cassandra	951	7,674	12,431	22,684
Workload C	GridDB	5,149	50,211	111,996	220,950
	Cassandra	884	7,353	12,082	21,129
Workload D	GridDB	17,575	77,486	155,445	316,608
	Cassandra	2,881	15,003	24,349	44,677
Workload F	GridDB	2,242	16,209	36,188	83,399
	Cassandra	788	6,236	8,960	16,212

大きなデータセットでのレイテンシ (ノードあたり 1,200 万レコード) -- 1 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	9.7			
	Cassandra	7.0			
Workload A	GridDB		44.2		87.0
	Cassandra		130.8		19.5
Workload B	GridDB		28.7		56.9
	Cassandra		140.2		23.8
Workload C	GridDB		24.8		
	Cassandra		144.6		
Workload D	GridDB	7.3	7.2		
	Cassandra	27.0	45.3		
Workload F	GridDB		29.1	84.7	55.6
	Cassandra		149.1	175.2	26.1

大きなデータセットでのレイテンシ (ノードあたり 1,200 万レコード) -- 8 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	12.9			
	Cassandra	15.9			
Workload A	GridDB		56.4		80.6
	Cassandra		148.4		13.1
Workload B	GridDB		27.7		48.4
	Cassandra		139.0		13.2
Workload C	GridDB		20.2		
	Cassandra		138.6		
Workload D	GridDB	14.2	13.0		
	Cassandra	12.1	70.8		
Workload F	GridDB		46.1	79.3	33.2
	Cassandra		47.5	91.6	44.1

大きなデータセットでのレイテンシ (ノードあたり 1,200 万レコード) -- 16 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	14.2			
	Cassandra	28.2			
Workload A	GridDB		50.4		72.3
	Cassandra		194.0		20.2
Workload B	GridDB		25.1		43.4
	Cassandra		171.5		18.4
Workload C	GridDB		18.1		
	Cassandra		168.6		
Workload D	GridDB	13.9	13.0		
	Cassandra	14.6	87.1		
Workload F	GridDB		41.0	71.1	30.1
	Cassandra		217.7	236.6	18.9

大きなデータセットでのレイテンシ (ノードあたり 1,200 万レコード) -- 32 ノード

		Insert	Read	Read-Mod-Write	Update
Load	GridDB	14.5			
	Cassandra	40.1			
Workload A	GridDB		44.6		65.1
	Cassandra		224.8		35.7
Workload B	GridDB		22.4		40.6
	Cassandra		187.4		22.3
Workload C	GridDB		18.2		
	Cassandra		192.4		
Workload D	GridDB	13.4	12.3		
	Cassandra	14.9	94.8		
Workload F	GridDB		34.7	62.3	27.6
	Cassandra		238.3	262.6	24.3

結論

GridDB のメモリを中心に考えられたハイブリッド型インメモリ・アーキテクチャは、メモリ内に収まるデータセットと、メモリ内に収まらずストレージを必要とするデータセットに対するオペレーションの両方で、Cassandra を上回ることが分かりました。さらに GridDB は 24 時間の長時間動作においても、安定した性能を維持しました。

GridDB のノード間通信は、少なくとも 32 ノードまで、Cassandra の分散型ピアツーピアシステムよりも大幅に優れていると言えます。Cassandra がある特定の Azure インスタンスタイプで、ノード数の増加に対しパフォーマンスが 50%程度でしかスケールできないのに対し、GridDB のパフォーマンスは追加されるノードの数にほぼ比例して増加しました。

付録

設定ファイル例

gs_node.json

```
{
    "dataStore":{
        "dbPath":"data",
        "storeMemoryLimit":"6144MB",
        "storeWarmStart":true,
        "concurrency":2,
        "logWriteMode":1,
        "persistencyMode":"NORMAL",
        "affinityGroupSize":4
    },
    "checkpoint":{
        "checkpointInterval":"1200s",
        "checkpointMemoryLimit":"512MB",
        "useParallelMode":false
    },
    "cluster":{
        "servicePort":10010
    },
    "sync":{
        "servicePort":10020
    },
    "system":{
        "servicePort":10040,
        "eventLogPath":"log"
    },
    "transaction":{
        "servicePort":10001,
        "connectionLimit":10000
    },
    "trace":{
        "default":"LEVEL_ERROR",
        "dataStore":"LEVEL_ERROR",
        "collection":"LEVEL_ERROR",
        "timeSeries":"LEVEL_ERROR",
        "chunkManager":"LEVEL_ERROR",
        "objectManager":"LEVEL_ERROR",
        "checkpointFile":"LEVEL_ERROR",
        "checkpointService":"LEVEL_INFO",
        "logManager":"LEVEL_WARNING",
        "clusterService":"LEVEL_ERROR",
        "syncService":"LEVEL_ERROR",
        "systemService":"LEVEL_INFO",
        "transactionManager":"LEVEL_ERROR",
        "transactionService":"LEVEL_ERROR",
        "transactionTimeout":"LEVEL_WARNING",
        "triggerService":"LEVEL_ERROR",
        "sessionTimeout":"LEVEL_WARNING",
        "replicationTimeout":"LEVEL_WARNING",
        "recoveryManager":"LEVEL_INFO",
        "eventEngine":"LEVEL_WARNING",
```

```
        "clusterOperation":"LEVEL_INFO",
        "ioMonitor":"LEVEL_WARNING"
    }
}
```

gs_cluster.json

```
{
    "dataStore":{
        "partitionNum":128,
        "storeBlockSize":"32KB"
    },
        "cluster":{
            "clusterName":"defaultCluster",
            "replicationNum":1,
            "heartbeatInterval":"5s",
            "loadbalanceCheckInterval":"180s",
            "notificationMember": [
            {
                "cluster": {"address":"10.0.0.13", "port":10010},
                "sync": {"address":"10.0.0.13", "port":10020},
                "system": {"address":"10.0.0.13", "port":10040},
                "transaction": {"address":"10.0.0.13", "port":10001},
            }
            ]
        },
        "sync":{
            "timeoutInterval":"30s"
        }
}
```

cassandra.yaml

```
cluster_name: 'Test Cluster'
num_tokens: 256
hinted_handoff_enabled: true
hinted_handoff_throttle_in_kb: 1024
max_hints_delivery_threads: 2
hints_directory: /var/lib/cassandra/hints
hints_flush_period_in_ms: 10000
max_hints_file_size_in_mb: 128
batchlog_replay_throttle_in_kb: 1024
authenticator: AllowAllAuthenticator
authorizer: AllowAllAuthorizer
role_manager: CassandraRoleManager
roles_validity_in_ms: 2000
permissions_validity_in_ms: 2000
credentials_validity_in_ms: 2000
partitioner: org.apache.cassandra.dht.Murmur3Partitioner
data_file_directories:
    - /var/lib/cassandra/data
commitlog_directory: /var/lib/cassandra/commitlog
disk_failure_policy: stop
commit_failure_policy: stop
key_cache_size_in_mb:
key_cache_save_period: 14400
```

```
row_cache_size_in_mb: 0
row_cache_save_period: 0
counter_cache_size_in_mb:
counter_cache_save_period: 7200
saved_caches_directory: /var/lib/cassandra/saved_caches
commitlog_sync: periodic
commitlog_sync_period_in_ms: 10000
commitlog_segment_size_in_mb: 32
seed_provider:
    - class_name: org.apache.cassandra.locator.SimpleSeedProvider
      parameters:
          - seeds:  ${SEEDS}
concurrent_reads: 32
concurrent_writes: 32
concurrent_counter_writes: 32
concurrent_materialized_view_writes: 32
memtable_allocation_type: heap_buffers
index_summary_capacity_in_mb:
index_summary_resize_interval_in_minutes: 60
trickle_fsync: false
trickle_fsync_interval_in_kb: 10240
storage_port: 7000
ssl_storage_port: 7001
start_native_transport: true
native_transport_port: 9042
start_rpc: false
rpc_port: 9160
rpc_keepalive: true
rpc_server_type: sync
thrift_framed_transport_size_in_mb: 15
incremental_backups: false
snapshot_before_compaction: false
auto_snapshot: true
tombstone_warn_threshold: 1000
tombstone_failure_threshold: 100000
column_index_size_in_kb: 64
batch_size_warn_threshold_in_kb: 5
batch_size_fail_threshold_in_kb: 50
compaction_throughput_mb_per_sec: 16
compaction_large_partition_warning_threshold_mb: 100
sstable_preemptive_open_interval_in_mb: 50
read_request_timeout_in_ms: 50000
range_request_timeout_in_ms: 100000
write_request_timeout_in_ms: 100000
counter_write_request_timeout_in_ms: 100000
cas_contention_timeout_in_ms: 10000
truncate_request_timeout_in_ms: 600000
request_timeout_in_ms: 900000
cross_node_timeout: false
endpoint_snitch: SimpleSnitch
dynamic_snitch_update_interval_in_ms: 100
dynamic_snitch_reset_interval_in_ms: 600000
dynamic_snitch_badness_threshold: 0.1
request_scheduler: org.apache.cassandra.scheduler.NoScheduler
server_encryption_options:
    internode_encryption: none
    keystore: conf/.keystore
```

```
    keystore_password: cassandra
    truststore: conf/.truststore
    truststore_password: cassandra
client_encryption_options:
    enabled: false
    optional: false
    keystore: conf/.keystore
    keystore_password: cassandra
internode_compression: all
inter_dc_tcp_nodelay: false
tracetype_query_ttl: 86400
tracetype_repair_ttl: 604800
gc_warn_threshold_in_ms: 1000
enable_user_defined_functions: false
enable_scripted_user_defined_functions: false
windows_timer_interval: 1
transparent_data_encryption_options:
    enabled: false
    chunk_length_kb: 64
    cipher: AES/CBC/PKCS5Padding
    key_alias: testing:1
    key_provider:
      - class_name: org.apache.cassandra.security.JKSKeyProvider
        parameters:
          - keystore: conf/.keystore
            keystore_password: cassandra
            store_type: JCEKS
            key_password: cassandra
```

Cassandra Schema

```
create keyspace ycsb  WITH REPLICATION = {'class' : 'SimpleStrategy',
'replication_factor': 1 };"

create table ycsb.usertable ( y_id varchar primary key, field0 varchar, field1
varchar, field2 varchar, field3 varchar, field4 varchar, field5 varchar, field6
varchar, field7 varchar, field8 varchar, field9 varchar, field10 varchar );"
```