1A1-OS-17a-3

OS-17 「ビッグデータと AI 技術」招待講演
Invited talk on OS-17 “Big Data and AI Technology”

ビッグデータインメモリ

分散 Key Value Store「okuyama」を用いた分散メモリーストレージの構築と検証

Big data in memory

岩瀬 高博[*1,2] 藤井 秀明[*2] 泥谷 誠[*2] 岩爪 道昭[*2]
Takahiro Iwase Hideaki Fujii Makoto Hijiya Michiaki Iwazume

[*1] 株式会社 神戸デジタル・ラボ [*2]独立行政法人 情報通信研究機構
Kobe Digital Labo, inc National Institute of Information and communications Technology

This paper proves validity of using memory to manage huge volume of data. To prove that, we use NoSQL database "okuyama" developed by Iwase and conduct performance comparison of "okuyama"'s two kinds of data storing mechanism: the one is storing in harddisk, and the other is storing in memory.

1. はじめに

近年、インターネットの普及により誰もがネットワーク上の情報を閲覧し公開することが容易となってきている。さらにソーシャルネットワークサービスやセンサーデータなどの普及も重なり大量の非定型データがリアルタイムに生成されるようになってきた。

これらのデータは短時間のうちに数億、数十億の件数となりそのサイズはテラバイトに収まらずペタバイトにもおよぶ。この短時間かつ大容量のデータ生成のサイクルを総称して「ビッグデータ(big data)」と呼ばれる。ビッグデータを解析、活用することでいままでにないビジネスサービスの創出や、現在のサービスの改善に役立つ可能性が高いため多くの分野から高い関心が寄せられている。

またビッグデータを高速に効率よく処理するためのデータベースとして NoSQL と言われるデータベースも登場し始めた。

本論文はデータベース技術の背景整理から、現在 NICT ユニバーサルコミュニケーション研究所にて進めている数十億ページ規模の Web アーカイブを含む大規模情報基盤の構築に向けた研究開発[岩爪 2012]の一環として実施している NoSQL データベース okuyama を利用した大規模データ処理実験の結果を基に考察と今後の取り組みについて説明する。

2. データベース技術について

本章ではデータを蓄積活用するために必要不可欠となるデータベース技術について整理をおこなう。

データベースは管理するデータの形式と処理方法に応じて最適な種別が存在する。ここでは代表的の 2 つのデータベースについて説明をおこなう。

(1) リレーショナルデータベース

データが定型的であり、データ間の関連性を重要視する場合にもっとも最適なデータベースはリレーショナルデータベースといえる。リレーショナルデータベース(RDBMS)は現在広く用いられているデータベースであり、関係性の構築と取得するデータの条件を SQL といわれる言語で記述し指定する。複数のデータに対する複雑な条件定義による処理を得意としている。またトランザクションといわれる一連の情報処理の単位を指定でき、その単位内で実行される複数の処理は他の処理から独立させることが可能である。そのため複数のデータへの排他的更新を行う場合などに活用される。ただし、これらの特性はデータ数が大規模になった際、参照時に大量のデータの読み込む必要があることや、大量の同時更新が発生した際に複数の処理を同期化する必要があるため性能の低下につながる。

(2) NoSQL データベース

管理するデータが非定型であり大規模、大容量のデータを管理しなければいけない場合は、NoSQL といわれるデータベースが活用されている。NoSQL は複雑な条件指定による検索やトランザクション処理のような処理単位の独立性を保障することは苦手とするが、これらの処理を省くことで 1 データへの参照性、更新性を極限まで高めている。
また複数の計算機資源を束ねて 1 つのデータベースを構築することが可能な場合も多く、この特性を利用することで 1 台では管理することが出来ないような大規模なデータ管理も可能である。また水平スケーラビリティも高くデータベース起動後も動的に計算機の追加を行い処理能力を向上させる機能を備えるものも多く存在し、大規模データ処理を強く意識していることがわかる。NoSQL は、Not Only SQL と呼ばれるデータベース群の総称であり、実際には、キーバリュー型、コラム型、ドキュメント型、グラフ型が 4 種類のデータベースが存在する。

以下本稿では、非定型なネット上の収集、蓄積により適した、キー・バリュー型 NoSQL の一種である分散キー・バリュー・ストアに焦点を当て、膨大なデータをメモリ上で高速に蓄積、アクセス可能とするアプローチについて概説する。

3. 分散キー・バリュー・ストア okuyama

(1) 全体アーキテクチャ

連絡先：岩瀬 高博，株式会社神戸デジタル・ラボ，
iwase@kdl.co.jp

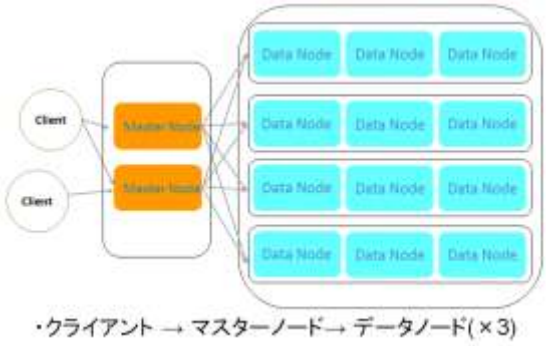

図 1 okuyama 全体アーキテクチャ

okuyama は Java 言語で実装されておりマルチプラットフォームで稼動可能なデータベースである。

データノード/マスターノードの 2 つの要素で構成され、データノードはデータの管理を行う。マスターノードはデータベース利用者とデータノード間の接続の中継を行い複数のデータノードを束ねて管理し利用者には全てのデータノードを 1 つの空間として提供する。

また両方の要素とも冗長構成を構築することが可能であり、障害時は自動的にフェイルオーバーが行われる。

それぞれのノードは[図 1]のように配置され TCP/IP ネットワークにて結ばれている。

(2) データ構造

キーバリュー型のデータ構造を採用しており、キーによる検索、更新が可能である。トランザクションはサポートされていない。

取得処理に関してはデータ一貫性のレベルを変更することが可能となっており、3 段階に一貫性レベルを調整可能である。最大のレベルを選択した場合、各データのバージョン情報を利用しレプリケーション環境下でも高い一貫性を維持することが可能である。またこの他にもデータの新規性を保証した登録機能や、あらかじめデータに有効期限を設けた登録が可能である。

検索に関しては Tag 機能を有しており、データ登録時に Tag を登録することで、取得時に同じ Tag を付加されているデータをグループとして一括取得が可能である。Tag に関しては 1 つのキーに無制限に付加することが可能となっており、後から削除することも可能である。また、複数の Tag を一度に指定し、AND 条件や OR 条件といったデータのマージ処理を行いながら取得することも可能である。

(3) プロトコル

アクセスプロトコルには専用のオリジナルプロトコルと Memcached プロトコルを採用している。オリジナルプロトコルはアスキープロトコルを採用しており、また仕様をオープンに公開しているため、昨今有志によるサードパーティー製のクライアントも登場している。

(4) ストレージ

オンメモリ型ストレージ、ディスク型ストレージ両方を備えており、高速な応答が必要な場合や、大容量を管理する場合などで使い分けることが可能である。また、キーバリューのキーのみをメモリに保存するなども可能であり多様なユースケースに対応可能である。また一度保存したデータは全てのストレージタイプ間で互換性を持つためデータの損失なくタイプを変更可能である。

(5) スケーラビリティ

データノード/マスターノード共に無停止にて動的に追加可能である。追加を行うことで全体の処理能力を向上させることが可能である。また追加時も全ての利用者からの操作に応答することが可能であり、データノードに関してはデータの再配置も自動的に行われる。

(6) 運用

複数のユーザが同時に異なるユースケースで利用することを考慮し、パーティション機能を有している。この機能を利用することで安全に空間を分割することが可能である。また、パーティション単位でのデータフォーマットや、サイズ取得などが可能である。

バックアップ機能を有しており、運用中に無停止でデータのスナップショットバックアップが取得可能である。このバックアップデータは全てのストレージタイプで互換性がある。

4. 大規模インメモリ DB に向けた実証実験

4.1 研究背景と実験目的

大規模情報基盤の研究において Web クローラによる大規模データ収集の研究がおこなわれている。クローラが収集するページは 40 億にも及び、大規模なコンテンツの管理と同時に Web ページの住所となる大量の URL の管理が必要となる[藤井 2012]。

そこで okuyama が URL 管理を行うデータベース基盤として活用できないか実際にデータを投入し実験をおこなうこととした。管理を行うデータの性質は大量というだけではなく、1 つのデータ当たり 1000byte を超えるサイズとなるため、総容量としても大きなものとなる。

データへのアクセス要求に関しは参照、更新の両方が同時に複数発生することを想定し同時処理時の能力も確認するものとした。

4.2 実験内容

並列アクセス時の処理能力と、大容量データの管理という 2 つの側面の能力を検証するために、okuyama のデータノードの持つストレージの特性を変化させ実験を行い結果を検証することとした。

(1) 保存データのフォーマット

- キーデータ:30byte 固定長でのユニークな値
- バリューデータ :1000byte 固定長によるユニークな値

(2) ストレージ特性

- インメモリ型
  保存するキー、バリュー共にメモリ上に格納する設定。
  保存前に圧縮を行う設定とした。ただし圧縮を行う際に 1 データ単位で行うのではなく、ある程度のデータを束ねてバイナリ情報としてから圧縮を行う設定とした。この設定を okuyama ではシリアライズマップと呼ぶ。
  圧縮アルゴリズムは Zip を利用。
- ディスク型
  保存するキーをメモリ上に保存し、バリューをディスクに保存する設定。
  キーに関してはインメモリ型設定と同様とし、バリューを格納するディスクに関しては 1 データを 4096byte の固定長データとしてディスク上の 1 つのデータファイル上で全てのデータを管理する設定とした。

(3) 冗長構成

データノードは全て冗長化され格納された 1 つのデータは必ず複製が 1 つ作成されて保存されるレプリケーション方式とした。そのため okuyama 上に保存されているデータの数は登録されたデータ数の 2 倍になる設定である。

(4) 実験方法

あらかじめユニークなキーバリューのデータを 40 億件登録しておきテストパターンを実行

- 新規書き込みテストパターン
  0 件の状態から 40 億件に到達するまでの処理時間を計測
- ランダム書き込みテストパターン
  200 の独立したクライアントスレッドよりランダムな値を登録し、5 分間で登録出来た件数から性能を測定
- ランダム読み込みテストパターン
  200 の独立したクライアントスレッドよりランダムに登録済みの値を取得し、5 分間で取得出来た件数から性能を測定
- ランダム書き込み、ランダム読み込み同時テストパターン
  200 の独立したクライアントスレッドよりランダムに読み書きを行い、5 分間に処理出来た件数から性能を測定

(5) 実験環境

計算機は以下のスペックの 88 台の PC クラスタ利用した。

- CPU:Intel Xeon X5650 2.67GHz × 2 (12Core)
- Memory:74GB
- HDD:44TB(RAID6)
- Network:1Gbps

また、各計算機の実行環境は以下のものを用い、Java VM 内部のガーベージ・コレクションアルゴリズムとして、G1GC を採用した。

(6) okuyama の構成

- データノード:80 台/200 ノード
- マスターノード:5 台/5 ノード
- クライアント:2 台/200 クライアントスレッド

4.3 結果と考察

テスト結果の考察前に保存されるデータ量を計算し整理する。1 データ当たりキーが 30byte、バリューが 1000byte であるため、1 データ当たりのデータサイズは 1030byte となり、合計 40 億件のデータを保存するためには、3.837TByte が必要となる。全てのデータのレプリケーションを持つため、全体では 7.674TByte となる。

(1) 新規書き込みテスト結果からの考察

- インメモリ型でのテスト結果

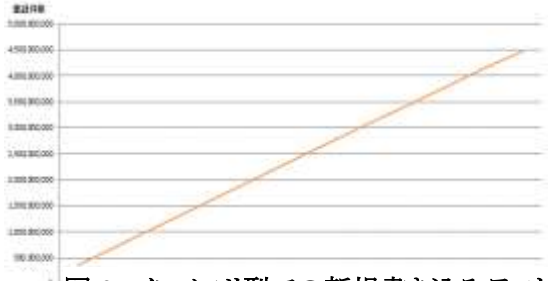

図 2 インメモリ型での新規書き込みテスト結果

[図 2]のテスト結果より約 5 時間 30 分で 40 億件のデータ登録が完了していることがわかる。また 30 分で約 3.5 億件ずつデータ登録件数が増えており、その処理能力を最後まで維持している。

またデータ総量の計算から 40 億件のデータではレプリケーションデータも含めると 7.6TByte のデータサイズとなり、データノードが利用している 80 台の計算機のメモリ搭載量の合計である 5.9TByte を上回るサイズを収容出来ていることがわかる。これは圧縮処理による効果だと推測できる。

- ディスク型でのテスト結果

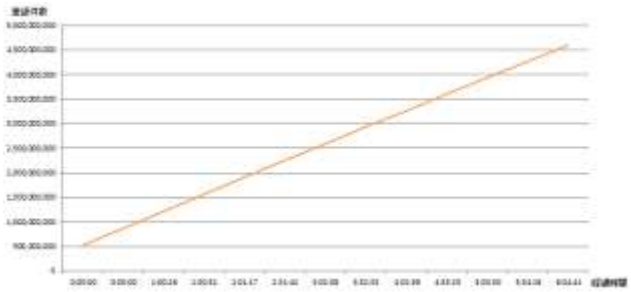

図 3 ディスク型での新規書き込みテスト結果

[図 3]のテスト結果より 5 時間 30 分で 40 億件のデータ登録が完了している。また 30 分で約 3.5 億件ずつデータ登録件数が増えており、その処理能力を最後まで維持している。

この結果により、インメモリ型とほぼ同等の性能であることがわかる。ディスクを利用しているにも関わらずインメモリ型と同様の処理能力があることから、ディスクへのアクセスを行う前に利用 OS のディスク書き出し前のバッファリング機構や、RAID コントローラによるキャッシングが考えられるが、正確な調査は未実施である。

(2) 5 分間のテスト結果からの考察

表 1.テストパターン別処理件数

	インメモリ型	ディスク型
ランダム書き込み	55,747,775	2,255,629
ランダム読み込み	25,039,099	2,274,220
ランダム読み/書き	36,833,982	2,140,052

インメモリ型の場合、書き込みテストでの処理回数が読み込み回数の 2 倍以上となっている。このことから圧縮を行う書き込みに処理に対して、復元をおこなう読み込み処理が低速なことが読み取れる。また 5 分間で約 5500 万件の処理が行えていることから、30 分換算にすると約、3.3 億件(毎秒約 18 万 3 千件)の登録が出来ることがわかる。このことから新規書き込みテスト時の 30 分あたり約 3.5 億件(毎秒約 19 万 4 千件)という処理数と遜色ない処理が出来ていることがわかった。

ディスク型の場合、新規登録時はインメモリ型と同様の処理能力だったが、ランダムアクセスを行った場合インメモリ型の 10%以下の処理能力であることがわかった。これはディスクへのアクセスが発生した際に、常にディスクの参照を行う必要があるため性能が劣化しているものと思われる。

このことからインメモリ型を使い圧縮と組み合わせることでテラバイト級のデータをメモリ上で管理できることが分かった。また上書き処理やランダムアクセスが発生する状況下でもディスク環境下に比べて高い処理能力を発揮できることが分かった。

5. 今後の取り組み

本検証では単純なデータアクセスを中心に検証を実施した。今後は Tag 機能を利用したデータのグルーピングを行った場合の検証を行い、効率的なデータ取得を行える基盤の整備に活かせるように研究を継続する予定である。

また実運用を想定し障害発生後のデータ復旧時などの効率的な復旧技術に関しても研究を行う予定である。

6. まとめ

本論文ではビッグデータと昨今のデータベース技術の関係の整理を行い。NoSQL データベース okuyama を利用した大規模データ処理の検証結果を報告した。

さらに okuyama の持つストレージの特徴であるインメモリ設定とディスク設定の比較を行いメモリ型データベース環境下での大規模データへのアクセス性能の比較を実施し、メモリ活用による大規模データの高速処理について提案を行った。

参考文献

[岩爪 2012] 岩爪他: アジア情報 HUB プロジェクト(第一報), 2012 年度人工知能学会全国大会, 1A1-OS-17a-1, 2012.

[藤井 2012] 藤井他: ハイパフォーマンス・エラスティック・クローリング, 2012 年度人工知能学会全国大会, 1A1-OS-17a-1, 2012.