分散データベースの性能によるデータ入出力の最適化

郭崇[†1]　久永忠範[†1]　能登大輔[†1]　渕田孝康[†1]

概要： 近年，ネットワークの発展の同時に，ビッグデータの出現およびその分析と利用が進んでいる．またデータベースの可用性，安全性，スケーラビリディーについてさらなる要求も増えてきている．伝統的なリレーショナル型データベース管理システム（RDBMS）の代わりに，非リレーショナル型分散データベース（NoSQL）も開発されている．しかし，各分散データベースはそれぞれ異なる性能，言語，構造を持っている．この問題を解決するため，我々は一つのシステムを設計する．このシステムでは，各データベースはバックエンドでお互い繋がって，テータを入出力する時，自動的に最適なデータベースを選択する．今回の研究では，データの入出力の最適化を行う．

キーワード：分散データベース，NoSQL

Optimization of Data Input and Output According to The Characteristics of Distributed Database

CHONG GUO[†1] TADANORI HISANAGA[†1] DAISUKE NOTO[†1]
TAKAYASU FUCHIDA[†1]

Abstract: In recent years, with the develop of Internet, the appearance of big data and its analysis and use are progressing. The request of database's availability, safety and scalability are increasing. So, instead of a traditional relational database management system (RDBMS), a non-relational distributed database(NoSQL) was developed. But, every distributed database has its unique characteristics, language and architecture. To solve this problem, we design a system. In this system, databases can connect each other in backend, and when input or output data its can choose a best database to make it. So, in this paper, we found a optimization of data input and output.

Keywords: Distributed database, NoSQL

1. はじめに

近年，ネットワーク上に存在するデータは莫大な量となっている．これまでのデータベース管理システムの主流であるリレーショナル型データベース管理システム（RDBMS）では，ネットワーク上に偏在する情報を集中管理することはもはや困難となってきている．そこで，新しいデータベース「NoSQL」が注目され始めている．

しかし，各分散データベースはそれぞれ異なる性能，言語を持っており，分散データベースの特徴に合った利用をすることが必要となる．そこで，我々は一つのシステムを設計する．このシステムでは，各データベースはバックエンドでにお互い繋がる．データを入力する時，システムは自動的に最適な書き込み速いデータベースを選択する．そして，データを検索する時，システムは自動的に最適な読み出し速いデータベースを選択する．

そのため，先ず各データベースの性能を知ることが必要である．今回の研究では，各分散データベースの書き込み方式の違いに基づき，最適な入力選択方法を提案する．

†1 鹿児島大学
Kagoshima University

2. データベース

NoSQL データベースはいくつの特徴を持っている．

- 非リレーショナル
- 分散
- オープンソース
- 水平スケーラブル

以下，主流なデータベースを紹介する．

2.1 Apache Cassandra

Cassandra は，Facebook が開発した分散データベース管理システムである，2009 年から Apache の Top プロジェクトとなった．NoSQL のカラム・ファミリーを実装したものであり Amazon Dynamo が導入したアーキテクチャーの側面を利用して Big Table データ･モデルをサポートする[1]. そして，Cassandra の特徴は以下の通りである

- 高拡張性
- 一貫性
- 分散
- key-value 形構造

Cassandra のデータ・モデルを図 1 に示す．Cassandra はカラム，行，カラム・ファミリー，キー・スペースからなる．カラムは Cassandra の基本単位である，その実体はタプル(triplet)で，名前，値，タイムスタンプを持つ．行はラベル

として名前が付けられたカラムの集合である，カラム・ファミリーはカラムのコンテナであり，リレーショナル・モデルでのテーブルのようなものといえる．キー・スペースはカラム・ファミリーのコンテナである[2].

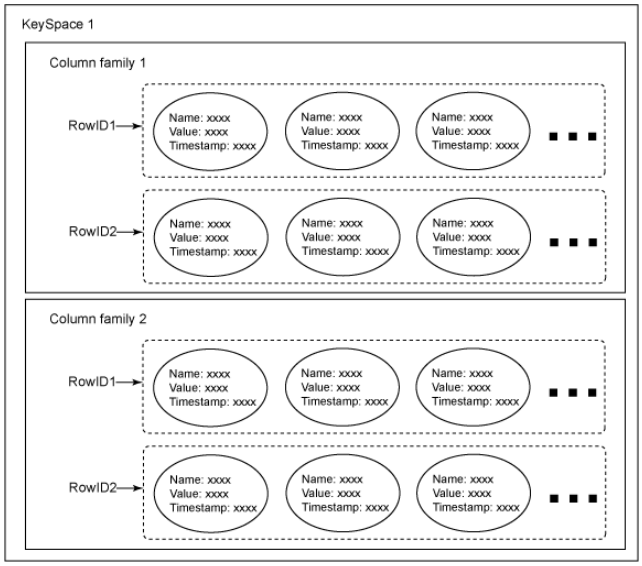

図 1　Cassandra のデータ・モデル
Figure 1　Data Model of Cassandra

2.2 MongoDB

MongoDB は 2007 年から 10gen 社が C++言語を用いて開発したドキュメント指向データベースである．高パフォーマンス，高可用性，容易なスケーラビリティを保持する．[3]

```
{
  name: "sue",                          ← field: value
  age: 26,                              ← field: value
  status: "A",                          ← field: value
  groups: [ "news", "sports" ]          ← field: value
}
```

図 2　MongoDB のドキュメント
Figure 2　Document of MongoDB

MongoDB 中のレコードはドキュメントで，図 2 は MongoDB のドキュメントを示す．MongoDB のドキュメントは field と value 二つの部分が含まれている．value 部分は他のドキュメントやアレイも格納できる．

2.3 HBase

HBase はオープンソース，分散，バージョン管理，非リレーショナルなデータベースで，Google の Bigtable に基づいて開発されている．HBase の目標は巨大なテーブルを格納することである，何十億行と何百万カラムのテーブルのよる[4].

HBase のデータ・モデルを図 3 に示す．図 3 のテーブルでは，一行と三つのカラム・ファミリーがある．行の名前は com.cnn.www で，カラム・ファミリーContents のバリューは三つ，カラム・ファミリーanchor のバリューは二つ，そして，カラム・ファミリーpeople はバリューがない．

Row Key	Time Stamp	Column Family contents	Column Family anchors	Column Family people
com.cnn.www	t9		anchor:cnnsi.com = "CNN	
com.cnn.www	t8		anchor:my.look.ca = "CNN.com	
com.cnn.www	t6	contents:html: "<html>...		
com.cnn.www	t5	contents:html: "<html>...		
com.cnn.www	t3	contents:html: "<html>...		

図 3　HBase のデータ・モデル
Figure 3　Data Model of HBase

一行の形はコードとして以下に示す.

```
{
   "com.cnn.www": {
    contents: {
      t6: contents:html: "<html>..."
      t5: contents:html: "<html>..."
      t3: contents:html: "<html>..."
    }
    anchor: {
      t9: anchor:cnnsi.com = "CNN"
      t8: anchor:my.look.ca = "CNN.com"
    }
    people: {}
  }
}
```

2.4 MySQL

MySQL は，NoSQL ではないが、世界で最も人気がある RDBMS である．高速、使いやすい、マルチユーザ対応はその特徴である[5]．現在，大部分の会社や組織は MySQL を使ってウェブサイトを構築している．

3. 実験と結果

3.1 実験環境

表 1 に開発環境を，表 2 に実験環境を，表 3 に各データベースのバージョンを示す.

表 1　開発環境
Table 1　Develop Environment

CPU	Intel core i3 3.20GHz
RAM	4GB
HDD	300GB
OS	Windows 10 Pro

表 2　実験環境

Table 2　Experiment Computer

CPU	Intel core i3 3.20GHz X 2	Intel core i7 3.40GHz	Intel core 2 2.66GHz X 2
RAM	4GB		
HDD	300GB X 2	500GB	250GB X 2
OS	Windows 10 Pro X 4		Windows 7 Pro

表 3　データベースのバージョン

Table 3　Database's Version

Database	Version
MySQL	5.7.13
Cassandra	3.7.0
MongDB	3.5
HBase	0.98.23

3.2 書き込み実験

今回の実験では，実際のビッグデータを入力データとして実験する．ビッグデータを利用するため，我々はテスト用のプログラムを開発した．

このプログラムでは，一つの大きなデータファイルから各サイズのデータファイルを生成する．具体例を図 1 に示す．プログラムは，先ず 1 行 1 カラムのファイルを生成して，次は 2 行 1 カラムのファイルを生成する，といった順番で 1 カラムの最後までファイル生成する．次に 1 行 2 カラムのファイルを生成して，その後，2 行 2 カラムのファイルを生成する．というようにこのカラムの末までファイルを生成する．全体的にはこの順番でファイルを生成し，データベースに入力する．そして，入力に要する時間を記録する，最後は収集した各データベースの入力時間を比較する．

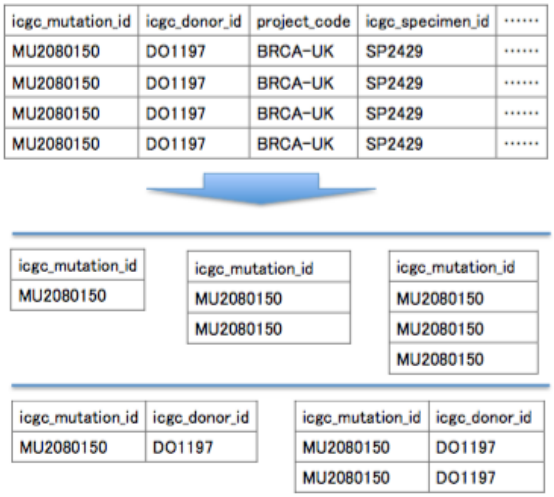

図 4 プログラムの流れ

Figure 4　Program Flow

3.3 書き込み実験の結果

図 5 は行により 1 ノードの各データベースの書き込み結果を示す．この図では，横軸は行の数，縦軸は実行時間である．この図より，Cassandra の実行時間が一番短い，次は MongoDB で，MySQL の実行時間は他のデータベースより長い．そして，Cassandra と MongoDB の実行時間はほぼ直線に増加していることが分かる．

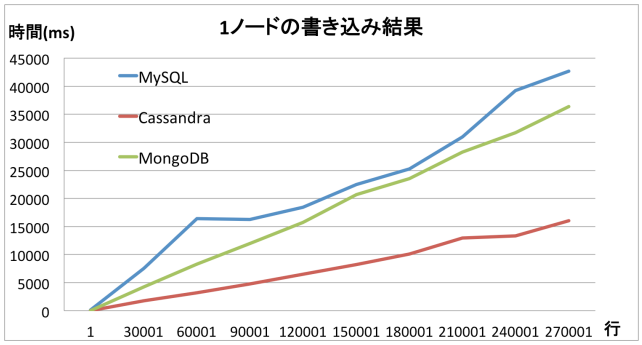

図 5　行により 1 ノードの書き込み結果

Figure 5　Writing Result in One Node by Row

図 6 はカラムにより 1 ノードの各データベースの書き込み結果を示す．この図では，横軸はカラムの数，縦軸は実行時間である．図 5 と同じ，実行時間が一番短いのは Cassandra，二番は MongoDB，MySQL の実行時間は一番長い．また，1 カラムの時，MongoDB と MySQL の実行時間はほぼ同じで，カラムが増えるにしたがって MySQL が遅くなる

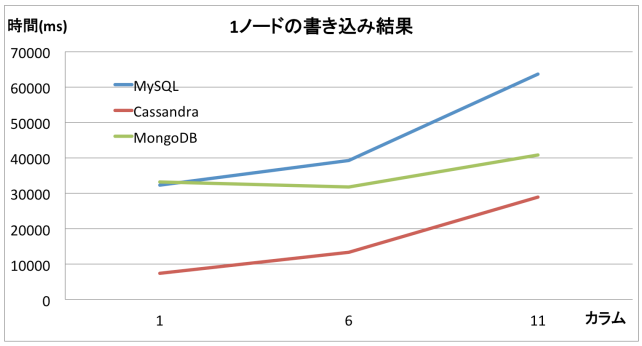

図 6　カラムにより 1 ノードの書き込み結果

Figure 6 Writing Result in One Node by Column

図 6 は 3 ノードの書き込み結果を示す．この図では，横軸は行の数，縦軸は実行時間である．この図より，行数が 65000 以前 Cassandra と MongoDB の実行時間はほとんど同じであるが，行数が増えるにしたがって Cassandra の実行時間が MongoDB より短くなる．

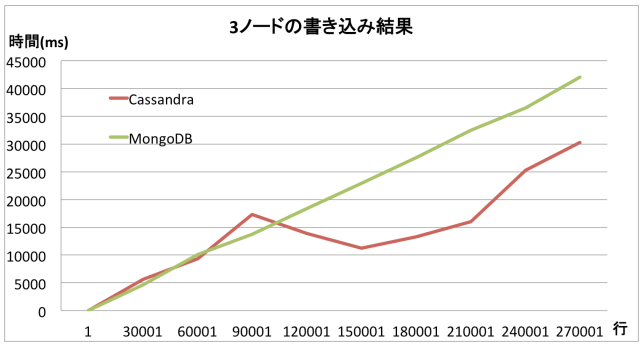

図 7　3 ノードの書き込み結果

Figure 7　Writing Result in Three Node

3.4 読み出し実験

読み出し実験では，我々は 7 つの検索条件を与えて，データベースからデータを読み出す．

1. テーブル中の最初の行から最後の行まで全部行を検索する．
2. テーブル中の行の数を計算する．
3. テーブル中の一部の行を検索する．例えば，1 行から前の 10000 行まで．
4. テーブルの中でキーワードを検索する．
5. テーブルの中でキーワードを検索し，結果の数を計算する．
6. テーブルの中でキーワードを検索し，結果を順番に配列する．
7. テーブルの中でキーワードを検索し，一部の結果を順番に配列する．

そして，各条件の検索時間を記録して，収集の時間を比較する．

3.5 読み出し実験の結果

読み出し実験の結果は表 4 に示す，表 4 では数字番号で条件を示す．

表 4　読み出し実験の結果

Table 4　Reading Result

条件	MySQL	Cassandra	MongoDB
1	16832ms	15289ms	23410ms
2	3446ms	8327ms	64ms
3	5094ms	11387ms	7655ms
4	4235ms	2221ms	1398ms
5	3714ms	2241ms	856ms
6	3958ms	2238ms	2153ms
7	3888ms	2246ms	842ms

図 8 は最初の三つの条件の読み出し結果を示す．この図では，各条件の結果がそれぞれ異なる．条件 1 の結果で，Cassandra の検索時間が一番短い，その次は MySQL，MongoDB の検索時間が一番長い．逆に条件 2 で，MongoDB の検索時間が他のデータベースより短い，Cassandra の検索時間が一番長い．そして，条件 3 の MySQL は検索時間が短くなっている．

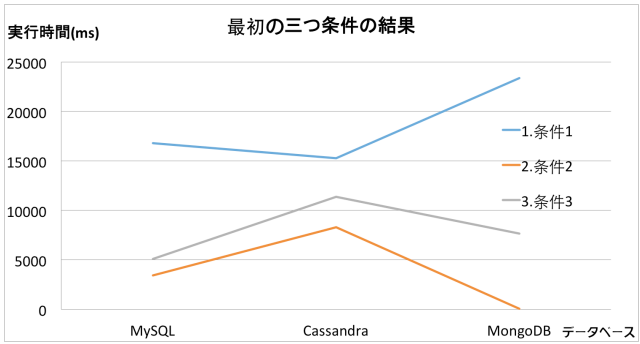

図 8　条件 1~3 の結果

Figure 8 The First Three Condition's Reading Result

図 9 は残りの四つの条件の読み出し結果を示す．この図では，各条件の結果ははっきり見える．MongoDB の検索時間が一番短い，そして，二番短いのは Cassandra，MySQL の検索時間が他のデータベースより長い．また，各条件では，Cassandra の検索時間がほとんど同じで，条件 5 と条件 7 で，MongoDB の検索時間がほぼ同じである．

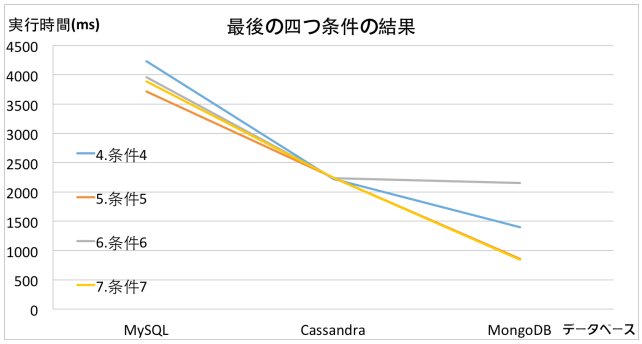

図 9　条件 4~7 の結果

Figure 9　The Last Four Condition's Reading Result

4. システムの入出力機能の実現

4.1 入力機能の実現

図 10 のように，システムの入力画面が示し．先ず，選択ボタンを押すと，入力ファイルを選択する．次は，キースベースの検索ボタンを押すと，データベース中のキースベースをプルダウンリストで示して，または新規ボタンによ

る新しいキースベースを作る．最後はテーブル名を入力して，入力開始のボタンを押す．入力結果は下の部分で示す．先ず，ファイルとテーブル名によってテーブルを作る，その成功のメッセージを出す．入力終了後，入力完了と入力時間のメッセージを下の部分で出す．

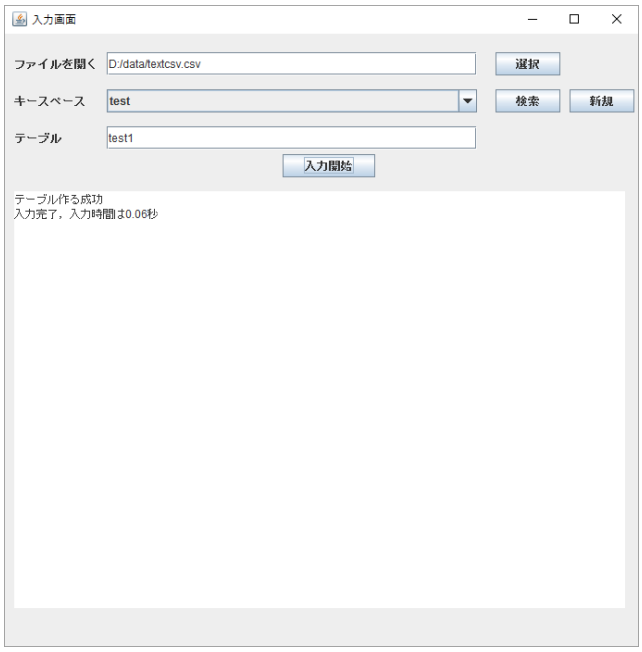

図 10　入力画面
Figure 10　Insert Page

4.2 出力機能の実現

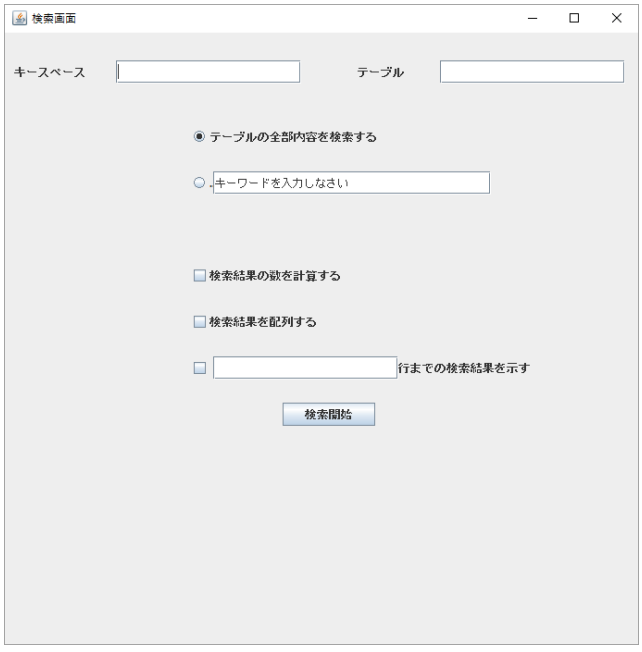

図 11　検索条件画面
Figure 11　Search Condition Page

図 11 は検索条件画面を示す．この画面では，キースベースとテーブルの入力が必要である．そして，各条件を選択して，検索する．検索の結果は図 12 のように，新しいページに出す．閉じるボタンを押すと，条件画面に戻ることができる．

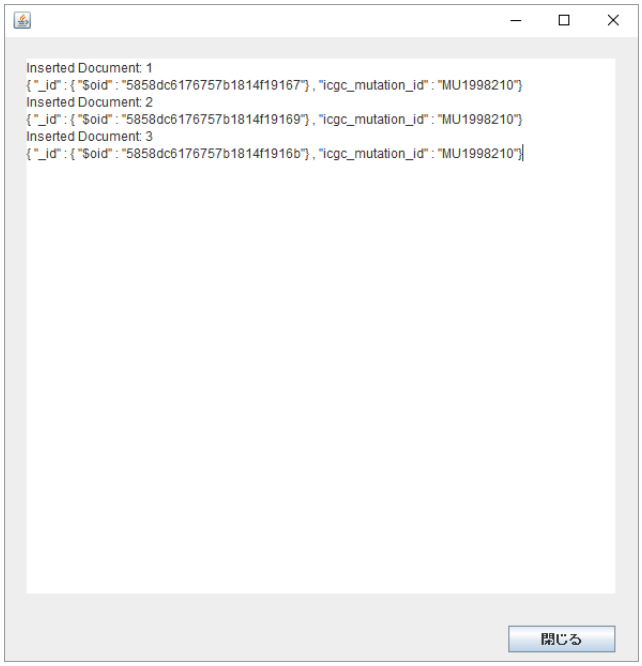

図 12　結果画面
Figure 12　Result Page

5. まとめ

今回の実験の結果では，Cassandra の書き込み性能は他のデータベースより実行時間が短い，つまり，書き込み速度が一番速い．読み出しの実験結果では，各条件の結果が別々だが，キーワード検索について，MongoDB の性能が一番優れている．そして，全テーブル検索は Cassandra のほうが良い．また，実験の結果によって，システムの入出力機能を実現する．

今後の課題として，先ず各データベース間の接続方法を探し，システムの自動通信機能を完成する．次は，システムの各データベース操作機能を完成する．また，分散データベースの HBase を加える．

参考文献

[1] Srinath Perera : “Consider the Apache Cassandra database”. http://www.ibm.com/developerworks/opensource/library/os-apache-cassandra/
[2] “Cassandra Wiki “. http://wiki.apache.org/cassandra/DataModel_JP
[3] “MongoDB Manual”. https://docs.mongodb.com/manual/introduction/
[4] “Apache HBase” . https://hbase.apache.org
[5] “MySQL”. http://www.mysql.com