MongoDBで作る XMLデータ検索アプリ

OSC.DB 2012
2012/07/25

株式会社野村総合研究所
情報技術本部
オープンソースソリューション推進室
藤崎 祥見

目次

- はじめに
- NoSQLとは
  - NoSQLとは
  - NoSQLの種類
  - よくある誤解
- MongoDB概要
  - 歴史と現在の状況
  - 事例紹介
  - MongoDBの特徴
- XML検索アプリで見るMongoDBのメリット

はじめに

NoSQLとは

MongoDB概要

XML検索アプリで見るMongoDBのメリット

自己紹介

- 研究室でLinuxを使用したことをきっかけにオープンソースに興味を持ち、Debian/Ubuntuのコミュニティで活動を始める。
- 2005年 Ubuntu 5.04 の翻訳に関わる。
- 2008年 NRI入社 OpenStandiaへ配属。以後、オープンソースを使用したシステム開発に携わる。
- 2009年 オープンソースであるLiferayの日本コミュニティ、日本Liferayユーザグループを共同設立。カンファレンスでの発表、勉強会、Wikiの整理を行う。
- 2012年 MongoDBの翻訳に関わる。7/30 に丸の内MongoDB勉強会を開催予定。

7/30 丸の内MongoDB勉強会

本日はNoSQLの一つであるMongoDBの
特徴を説明します。

はじめに

NoSQLとは

MongoDB概要

XML検索アプリで見るMongoDBのメリット

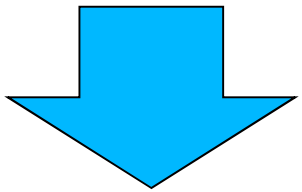
Not only SQL

- 「No! SQL」ではなく、「Not only SQL」
- RDBが得意なことはRDBで、得意でないことは無理にRDBにこだわらず、用途に合ったデータストアを使おう、というのが最近のコンセンサス
- 競合関係ではなく、補完関係という考え方が浸透してきている
- RDBの強み
  - トランザクションによってデータの一貫性を保証できる
  - 正規化を前提としているため、更新時のコストが小さい
  - JOINや複雑な検索条件での検索が可能
- RDBが得意ではないこと
  - 大量データの書き込み処理
  - 更新の発生するテーブルに対するインデックス作成やスキーマ変更
  - カラムを固定しずらい用途での利用

RDBが得意でないことを、NoSQLで補完する

NoSQLの特徴

- RDBが得意ではないこと
  - 大量データの書き込み処理
  - 更新の発生するテーブルに対するインデックス作成やスキーマ変更
  - カラムを固定しずらい用途での利用



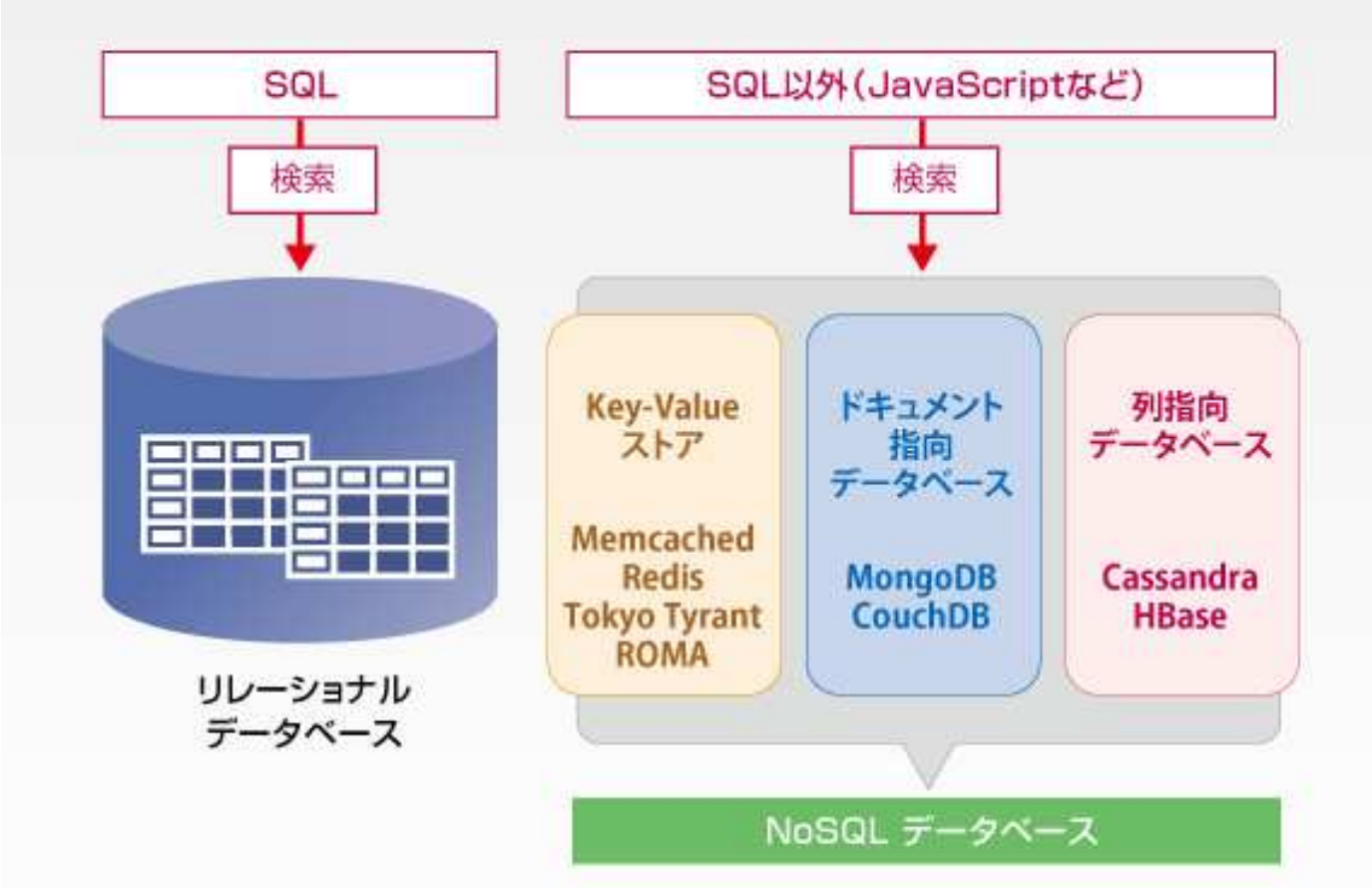
- NoSQLの特徴
  - ノード数に比例したスケーラビリティ
  - ノード数に比例しない運用コスト
  - 障害耐性

利用目的/利用範囲を絞り、RDBが持っている機能を一部制限することで、RDBには無い特性を持つ

出所: http://www.slideshare.net/yutuki/cassandrah-baseno-sql

NoSQLの種類

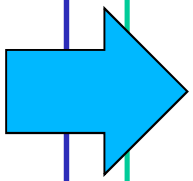
- NoSQLと呼ばれている代表的なものは大きく分けて3つに分類される



出所: http://openstandia.jp/oss_info/mongodb/

NoSQLへの流れ

- Web 1.0
  - 情報は一方通行
  - 通信回数は1回
  - HTMLでの静的コンテンツ
  - データは1つのデータセンター内で完結



- Web 2.0
  - 双方向通信
  - AJAXを利用した複数通信、随時通信
  - DBのデータからの動的コンテンツ
  - データは複数拠点のデータセンターに配置

データストアに求められるものが変わってきている

出所: http://www.slideshare.net/yutuki/cassandrah-baseno-sql

代表的なNoSQL：KVS

- KVS（Key-Value ストア）
  - 最も一般的なNoSQL。データをkeyとvalueの形で持つ。
  - 基本的にはkeyでの完全一致検索でしかデータを取得できない制限があるが、高速に動作する。
  - シンプルな方式なので気軽に使用することが可能。
  - Valueを条件とした検索、KeyでのLike検索はできない。
- 代表的なプロダクト
  - Memcashed, Redis, Tokyo Tyrant, ROMA, Infinispan(JBoss Cache)

KeyとValueしか無く、Keyでの完全一致でしかデータを取得できない。

key	value
app2020120420	‘www.xxx.com’
app2020120421	‘ [552.9565,N,1401]’
app2020120423	{Javaオブジェクト}

代表的なNoSQL：列指向データベース

- 列指向データベース
  - RDBMSのような行単位では無く、列単位での処理に特価している。
  - 全行に対する特定列の一括更新や、指定した範囲の大量行に対する複数列の取得に向いている。
  - 強力なスケーラビリティを備えており、データが増えても処理速度がそれほど低下しない。
  - RDBのデータストアとは考え方が大きく異なっているので、扱いが難しい。
  - Cassandra, HBase

Hbaseの例。Column単位で処理を行う。

Row Key	Time Stamp	Column Family " data "	
user 01	t3	data:loginstatus	1
	t3	data:lastlogin	20110201173000
	⋮		
	t2	data:pwdhistory	oldpwd
	t2	data:pwd	p@ssw0rd
user 02	t3	data:loginstatus	1
	t3	data:lastlogin	20110120173000
	⋮		
	t2	data:pwdhistory	pwdold
	t2	data:pwd	p@ssw0rd

代表的なNoSQL：ドキュメント指向データベース

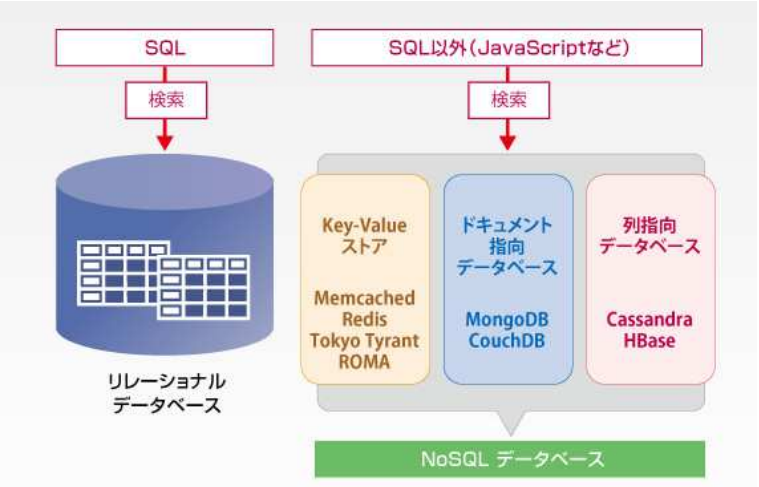
- ドキュメント指向データベース
  - スキーマを定義しなくても使用できるスキーマレスである。
  - スキーマ内でネストが可能。
  - 複雑な検索条件でデータを取得することが可能。
  - 一部の機能はRDBライクに使用することができる。
  - MongoDB, CouchDB

MongoDBの例。スキーマ内でのネストが可能。

```
{
        "_id" : ObjectId("4f769ba675c676"),
        "created_info" : {
                "owner" : "本田",
                "date" : 2012/07/07,
                "grope":{
                    "id":10,
                    "name":"営業本部"
                }
        },
        "title" : "test title",
        "body" : "test body",
        "has_image_url" : "true",
        "image_url" : "http://example.com/t001.jpg"
}
```

NoSQLの種類：まとめ

- 共通点
  - Join、Transactionなど一部のRDBMSの機能をサポートしないことによりハイパフォーマンスを実現している
  - スケールアウト、分散管理を前提に設計されている
- KVS（Key-Value ストア）
  - データをkeyとvalueで持つHash形式。
  - 基本的にはkeyでの完全一致検索でしかデータを取得できない制限があるが、高速に動作する。
- 列指向データベース
  - 行単位では無く、列単位での処理に特価している。
  - 書き込みに対して強力なスケーラビリティを発揮する。
- ドキュメント指向データベース
  - スキーマを定義しなくても使用できるスキーマレスである。
  - 複雑な検索条件でデータを取得することが可能。
  - 一部の機能はRDBライクに使用することができる。



- NoSQLと呼ばれているものは3つに大きく分けられ、MongoDBはその中で「ドキュメント指向データベース」に当てはまる

MongoDBとその他NoSQLとの比較

	MongoDB	memcached	HBase	Cassandra
SQLライクな検索	○	×	×	×
性能	○	◎	○	○
分散/スケーラビリティ	○	△	◎	◎
開発のしやすさ	◎	◎	×	×
日本語ドキュメント	○	○	○	○
商用サポート	○	×	○	○

MongoDBは、
KVSや列指向DBほど特化していないが、RDBライクに開発できる。

よくある誤解

- RDBMSは古い！最近とっても話題のNoSQLにRDBMSから乗り換えよう！
  => 最近は、NoSQLはRDBMSの競合ではなく補完となるもの、という考え方が一般的です。
- NoSQLはRDBMSよりも早いよね！
  => 一概には言えません。それぞれの得意・不得意があります。
- MongoDBはNoSQLだからKVS（Key Value Store）だよね。
  => 違います。MongoDBはドキュメント指向と呼ばれています。KVSよりも柔軟なデータ挿入が可能です。

はじめに

NoSQLとは

MongoDB概要

XML検索アプリで見るMongoDBのメリット

MongoDBとは

ドキュメント指向データベースで、NoSQLの一つに分類される

スキーマレス

データ形式にJSON（BSON）、shellにJavaScriptを採用

スケールアウトが容易

高パフォーマンス・スケーラビリティを保持しつつRDBライクな機能をバランスよく組み込むことを目指している

歴史と現在の状況

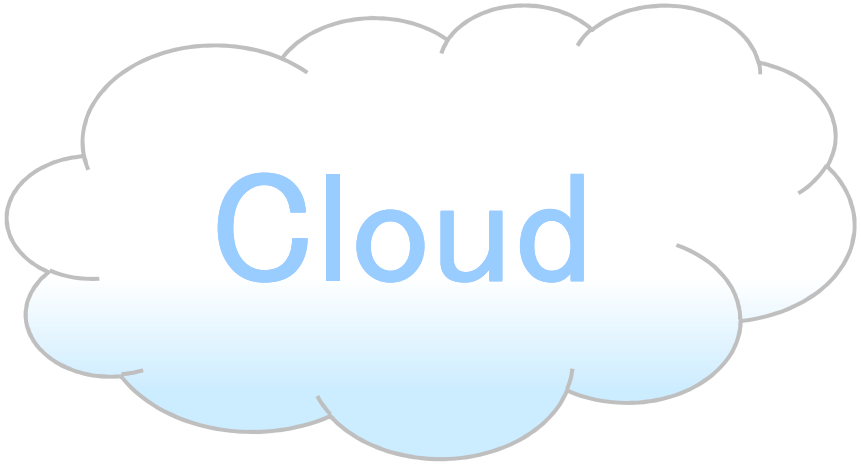
- 10gen,Inc.がオープンソースとして開発。開発言語はC++。ソースコードはGithubで公開。
  https://github.com/mongodb/mongo
- 2007/10 ファーストコミット
- 2009/02 ver.0.2.x
- 2009/08 ver.1.0.0・・・初のGAリリース
- 2009/12 ver.1.2.0・・・Map/Reduce
- 2010/03 ver.1.4.0・・・レプリケーション、クエリ言語強化
- 2010/06 ver.1.6.0・・・シャーディング/レプリカセット
- 2011/06 ver.1.8.0・・・ジャーナリング機能追加
- 2011/09 ver.2.0.0・・・開発・運用・信頼性・スケーラビリティに関する多くの機能追加
- 2012/06 ver.2.0.6・・・ 7/19現在の最新版

開発元が商用サポートを提供している点はMySQLと同様。
（Cassandra,HBaseの商用サポートは開発元とは別企業）

最近のイベント/ニュース
2011/02 MongoDB Tokyo 2011
2012/01 MongoDB Tokyo 2012
2012/04 RedHatと提携
2012/06 雲屋株式会社が代理店に

クラウドでの利用

Hosting	IaaS	PaaS
MongoOd.com	Amazon EC2	dotCloud
MongoHQ	Windows Azure VM	Heroku
MongoLab	Rackspace Cloud	NodeGrid
HostedMongo.com		RedHat OpenShift
MongoGrid		Vmware
		CloudFoundry



MongoDBの事例

- 日本
  - 楽天
  - サイバーエージェント
  - NHN Japan（NAVER）
  - nifty

※出所 MongoDB Tokyo 2012
http://www.10gen.com/events/mongo-tokyo-2012

- 国外
  - foursquare
  - New York Times
  - bit.ly
  - github
  - sourceforge
  - Forbes
  - Disney
  - 他多数

※出所 MongoDB Production Deployments
http://www.mongodb.org/display/DOCS/Production+Deployments

MongoDBの特徴

- ドキュメント指向スキーマレス
  MongoDBは、JSON形式でドキュメントを表現し、そのコレクションを管理します。ドキュメントはスキーマレスとなっており、任意のフィールドを好きなときに追加できます。フィールドはネスト可能で複雑なリレーションシップを表すことができます。
  また、KVSでは苦手な、Valueを検索の条件としたり、Valueでのソート・集計を実現できます。

- オートシャーディング機能
  MongoDBは、オートシャーディング機能を通じて、スケールアウトが容易にできるように設計されています。負荷やデータ分布の変化に伴う自動バランシングや自動フェイルオーバー機能を備えています。公式ドキュメントによると1000ノード以上でのスケールアウトが可能です

- 開発のしやすさ
  開発のしやすさはMongoDBプロジェクトの目的の一つです。MongoDBは内部データにJSONスタイルを採用し、Mongo ShellからはJavascriptを使用してアクセスできます。一部、RESTインターフェイスも備えているため、Javascriptとの相性が良く、生産性高く開発できます。また、各言語のドライバも充実しており、インターフェイスにはネイティブソケットプロトコルが用意されており高いパフォーマンスを実現しています。

スキーマレスのメリット

- カラムを固定できないデータ
  - 汎用的にしたいがための予備カラムを余分に持たなくても良い
    - Id, 名前, メールアドレス, 予備01, 予備02,…
  - 以下のデータが該当
    - ログデータ解析
    - アンケートのデータ解析
    - 管理台帳のインポート解析
    - XMLデータ
    - 行動履歴（友達になった、写真を追加した、写真にコメント書いた）
  - XMLの任意の要素をキーに検索可能
    - あるプロジェクトではXMLデータをMySQLのカラムに入れているが、インデックスが作成できずに困っている。
      例：item.total_valueが10のものを全て、がLike検索になってしまいできない。

スケールアウトが容易

- RDB
  - 参照系をスケールすることは比較的容易だが、更新系をスケールすることは困難
  - テーブルを分けるとjoinできない
  - Id（奇数、偶数）で分けるとjoinできない
  - 設計フェイズでの綿密な計画が必要となる

- MongoDB
  - ノードを増やせば自動でスケーリング
  - データ量の多いコレクションを自動で最適化

トランザクションに関して

- MongoDBは伝統的なロックを前提とする複数ドキュメントのトランザクションをサポートしていない。

  理由（公式ドキュメントより）

  - 分散された環境で、ロックの情報を分散させるのは、コストが高く、そして遅いです。MongoDBの目標は、軽く速いことです。
  - デッドロックという考えかたが嫌いです。そういうことがない、シンプルでわかりやすいシステムにしたいです。
  - MongoDBを、リアルタイムな問題に対して、よく動くようにしたいです。オペレーションが大量のデータをロックしてしまうと、長い時間、軽い小さなクエリーを止めてしまうことがあります。

- ただし、1つのドキュメントに対するAtomicな操作はサポートしている。
  - まったく実行されないか、完全に実行されるかのどちらか。

※出所 MongoDB公式ドキュメント
http://www.mongodb.org/pages/viewpage.action?pageId=7209573

MongoDBが向いているケース

非常に更新頻度が高いシステムに向いている

1. 監査ログやイベントログ
2. ドキュメントやコンテンツマネジメント
    - 柔軟なスキーマで対応できる
3. E-コマース
    - RDBMSとMongoDBのハイブリッド利用
4. ゲーム
    - 小規模なRead/Writesが大量にある場合
5. モバイル
6. webサイトの操作データの蓄積
7. アジャイル開発
8. リアルタイム統計分析

※出所 [Mongo] Use Cases
http://www.mongodb.org/pages/viewpage.action?pageId=21266728
http://www.mongodb.org/display/DOCS/Use+Cases
http://www.mongodb.org/display/DOCS/Production+Deployments

MongoDBがあまり向いないケース

1. 銀行系のように複雑なトランザクションに重点を置くシステム
    - mongoDBは単一のドキュメントに対するアトミックな操作はサポートしていますが、マルチ･オブジェクト･トランザクションをサポートしていません。

2. 伝統的なビジネスインテリジェンス(BI)
    - mongoDBはリアルタイムにデータを計算したり集約したりするのに向いていますが、夜間にバッチで動かすようなBIには向いていません。

3. SQLを必要とする問題
    - mongoDBはSQLをサポートしていません。

※出所 [Mongo] Use Cases
http://www.mongodb.org/pages/viewpage.action?pageId=21266728
http://www.mongodb.org/display/DOCS/Use+Cases
http://www.mongodb.org/display/DOCS/Production+Deployments

その他のデメリット

- Global Lock
- システムリソースが肥大化
- データ圧縮未対応
- セキュリティ周りが弱い

※出所 [Mongo] Use Cases
http://www.mongodb.org/pages/viewpage.action?pageId=21266728
http://www.mongodb.org/display/DOCS/Use+Cases
http://www.mongodb.org/display/DOCS/Production+Deployments

MongoDBの概要：まとめ

- KVSの高パフォーマンス・スケーラビリティと、RDBライクな機能をバランスよく備えた非リレーショナルデータベース
  - NoSQLの中でもドキュメント指向データベースとして分類
  - 非リレーショナルモデルだが、RDBライクな機能やI/Fを持ち、位置づけとしてはKVSとRDBの中間と表現される
  - スケーラビリティの高さ、パフォーマンス、開発のしやすさの特徴を保持しつつ、従来のデータベースでは一般的である重要な機能を組み込むことを目的に開発されている。
- 非常に更新頻度が高いシステムに向いている。
  - 小規模なRead/Writesが大量にあるようなソーシャルゲーム、Webサイトの操作データの蓄積など
  - システムの規模が拡大しても、オートシャーディング・アーキテクチャーを通じてスケールアウトができるように設計されている
  - またスキーマレスという特徴から、スキーマを固定できないコンテンツのマネジメントにも利用可能
  - 複雑なトランザクションを必要とするシステムは苦手だが、RDBとMongoDBのハイブリッド利用することによってECサイトでも利用可能

はじめに

NoSQLとは

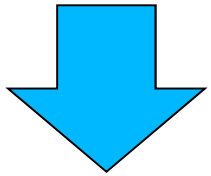
MongoDB概要

XML検索アプリで見るMongoDBのメリット

課題

- 現状の課題
  - XMLでデータが送られてくるが、MySQLの1つのカラムにデータをそのまま入れている。検索しようとするとLike文になるので実質検索ができない。

    -> なぜ1つのカラムにデータをそのまま入れるのか？
    XMLを格納する汎用的なテーブルを定義しようとすると、予備カラムが必要となるため。例：id, name, エレメント1,エレメント2, …



MongoDBの特徴であるスキーマレスで解決する

例：XMLデータの格納

CD

```
<?xml version="1.0" encoding="UTF-8" ?>
<items>
   <category>music cd</category>
   <title>アルバム1</ title >
   <stock>10</stock>
   <price>1000</price>
   <artist>nomura band</artist>
</items>
```

検索の条件にしたい

本

```
<?xml version="1.0" encoding="UTF-8" ?>
<items>
   <category>book</category>
   <title>本1</ title >
   <stock>10</stock>
   <price>1200</price>
   <ISBN>4797327421</ISBN>
   <publish>
     <company>nomura pub</company>
     <url>http://nomura.pub</url>
     <address>….</address>
   </publish>
</items>
```

ノートPC

```
<?xml version="1.0" encoding="UTF-8" ?>
<items>
   <category>note pc</category>
   <name>let's book air</ name>
   <stock>10</stock>
   <price>120000</price>
   <software>
     <os>windows 7 pro</os>
     <option>Office 2010</option>
   </software>
   <hardware>
     <cpu>Intell core i 7</cpu>
      ….
</items>
```

- どのような属性を持った商品が将来追加されるか不明な場合だが、共通属性（必須属性では無い）を検索の条件にしたい場合、RDBでは基本的に2つのアプローチしかない
  1. 商品が追加されるごとに別テーブルを作成する。Unionして一括検索。
     → 毎回だれがテーブル追加するんですか。。
     → スケールアウトするときどうするの。。

     CD | 本 | ノートPC | ？

  2. 予備カラムを用意して汎用的なテーブルを作成する。
     → 予備カラム数を超える数の属性を持ってたらどうするの。。

category	title	stock	price	other01	other02	other03	…

例：XMLデータの格納

CD

```
<?xml version="1.0" encoding="UTF-8" ?>
<items>
   <category>music cd</category>
   <title>アルバム1</ title >
   <stock>10</stock>
   <price>1000</price>
   <artist>nomura band</artist>
</items>
```

検索の条件にしたい

本

```
<?xml version="1.0" encoding="UTF-8" ?>
<items>
   <category>book</category>
   <title>本1</ title >
   <stock>10</stock>
   <price>1200</price>
   <ISBN>4797327421</ISBN>
   <publish>
     <company>nomura pub</company>
     <url>http://nomura.pub</url>
     <address>….</address>
   </publish>
</items>
```

ノートPC

```
<?xml version="1.0" encoding="UTF-8" ?>
<items>
   <category>note pc</category>
   <name>let's book air</ name>
   <stock>10</stock>
   <price>120000</price>
   <software>
     <os>windows 7 pro</os>
     <option>Office 2010</option>
   </software>
   <hardware>
     <cpu>Intell core i 7</cpu>
      ….
</items>
```

- MongoDBはスキーマ定義が必要ないので、どんなXMLが来ても格納可能。
- 検索条件にどのエレメントも指定できる。
  例1：> db.items.find({"stock" : {&gt: 10} }); // where stock > 10
  例2：> db.items.find({"title" : /^アルバム/ }); // where title like 'アルバム%'
- スケールアウトもできます

MongoDBのメリットを検証する

- スキーマレス
  - エレメント数の違うXMLのInsert
- Index付データのInsert/Updateが高速
  - RDBMSと比較してどうか
- JSONが標準語
  - 生産性が高いか

MongoDBのメリットを検証する

- スキーマレス
  - エレメント数の違うXMLのInsert

アプリ作成で検証

- Index付データのInsert/Updateが高速
  - RDBMSと比較してどうか
- JSONが標準語
  - 生産性が高いか

MongoDBのメリットを検証する

- スキーマレス
    - エレメント数の違うXMLのInsert
- Index付データのInsert/Updateが高速
    - RDBMSと比較してどうか
- JSONが標準語
    - 生産性が高いか

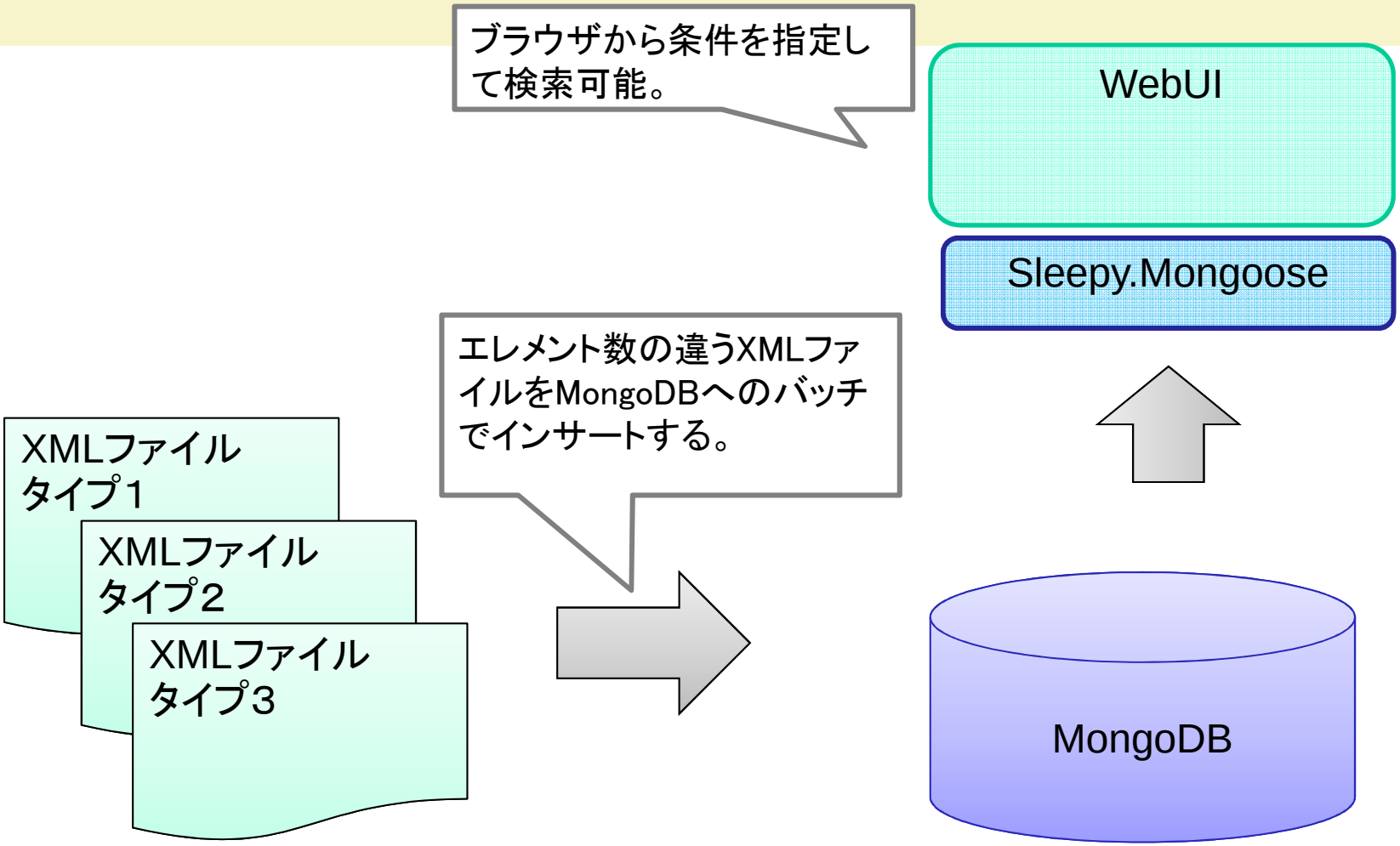
テストデータで
性能検証

MongoDBのメリットを検証する

- スキーマレス
    - エレメント数の違うXMLのInsert
- Index付データのInsert/Updateが高速
    - RDBMSと比較してどうか
- JSONが標準語
    - 生産性が高いか

見積もり依頼を出して検証

XML検索アプリシステム構成図



XML検索アプリデモ

証券コード、日付、会社名、件名、本文を選択可能。
また、日付は範囲検索が可能。

データ表示画面へのリンク

表示するデータは、ドキュメントタイプごとに違うエレメントを参照する。
※詳細はマッピングシートを参照

xml画面へのリンク

XmlSearch

証券コード val: 検索

id	証券コード	ドキュメントタイプ	日付	会社名	件名	xml
1	7551	業績予想修正	2011-10-25	株式会社 ウェッズ	業績予想の修正に関するお知らせ	xml
2	5201	決算短信	2011-11-04	旭硝子株式会社	第3四半期決算短信〔日本基準〕(連結)	xml
3	8518	決算短信	2011-11-04	日本アジア投資株式会社	第2四半期決算短信〔日本基準〕(連結)	xml
4	8766	業績予想修正	2011-11-04	東京海上ホールディングス株式会社	業績予想の修正に関するお知らせ	xml
5	4914	業績予想修正	2011-11-01	高砂香料工業株式会社	業績予想の修正に関するお知らせ	xml
6	3060	業績予想修正	2011-10-25	マガシーク株式会社	業績予想及び配当予想の修正に関するお知らせ	xml
7						xml
8						xml
9						xml
10		正				xml
11	2719	決算短信	2011-11-04	株式会社 キタムラ	第2四半期決算短信〔日本基準〕(連結)	xml
12	1848	業績予想修正	2011-10-27	株式会社富士ピー・エス	業績予想の修正に関するお知らせ	xml
13	6113	業績予想修正	2011-11-02	株式会社 アマダ	業績予想の修正に関するお知らせ	xml
14	7841	決算短信	2011-10-28	株式会社遠藤製作所	第2四半期決算短信〔日本基準〕(連結)	xml
15	8209	業績予想修正	2011-11-04	株式会社 フレンドリー	平成24年3月期第2四半期累計期間業績予想との差異に関するお知らせ	xml
16	4967	決算短信	2011-10-26	小林製薬株式会社	第2四半期決算短信〔日本基準〕(連結)	xml
17	6147	業績予想修正	2011-11-02	株式会社 ヤマザキ	業績予想の修正に関するお知らせ	xml
18	7979	決算短信	2011-11-02	株式会社 松風	第2四半期決算短信〔日本基準〕(連結)	xml
19	3060	決算短信	2011-10-28	マガシーク株式会社	第2四半期決算短信〔日本基準〕(非連結)	xml
20	8622	決算短信	2011-10-28	水戸証券株式会社	第2四半期決算短信〔日本基準〕(非連結)	xml

上場企業公開情報のxmlデータを検索するアプリ

Node-mecabでわかち分け & tagsに入れてIndex化

```
function insertTags(result){
  var str_array =[];
  result['tags'] = [];
  if(result[conf.xbrl] != void 0){
    var xlink_href = result[conf.xbrl][conf.link][conf.href];
    switch(true) {
      case /tdnet-rvfc/.test(xlink_href):
        str_array = [
               result[conf.xbrl][conf.tdnet_rvfc.code_]["#"],
               result[conf.xbrl][conf.tdnet_rvfc.date_]["#"],
               result[conf.xbrl][conf.tdnet_rvfc.company_]["#"],
               result[conf.xbrl][conf.tdnet_rvfc.document_]["#"],
               result[conf.xbrl][conf.tdnet_rvfc.body_]["#"]
              ];
        break;
      case /tdnet-rvdf/.test(xlink_href):
        str_array = [
               result[conf.xbrl][conf.tdnet_rvdf.code_]["#"],
               result[conf.xbrl][conf.tdnet_rvdf.date_]["#"],
               result[conf.xbrl][conf.tdnet_rvdf.company_]["#"],
               result[conf.xbrl][conf.tdnet_rvdf.document_]["#"],
               result[conf.xbrl][conf.tdnet_rvdf.body_]["#"]
              ];
        break;
      case /tdnet-qnedjpsm/.test(xlink_href):
        str_array = [
               result[conf.xbrl][conf.tdnet_qcedjpsm.code_]["#"],
               result[conf.xbrl][conf.tdnet_qcedjpsm.date_]["#"],
               result[conf.xbrl][conf.tdnet_qcedjpsm.company_]["#"],
               result[conf.xbrl][conf.tdnet_qcedjpsm.document_]["#"],
               result[conf.xbrl][conf.tdnet_qcedjpsm.body_]["#"]
              ];
        break;
      default:
        break;
    }
    for(i=0;i<str_array.length;i++){
      result['tags'] = result['tags'].concat( wakati(str_array[i]) );
    }
  }
  return result['tags'];
}
```

```
//insert
db.open(function() {
  db.collection(coll_name, function(err, collection) {
    fs.readdir(file_dir,function(err,files){
      files.forEach (function(file){
        //非同期でファイルオープンする場合、OSの制限に注意する
        fs.readFile(file_dir + file,function (err, data) {
          if (err) throw err;
          //insert
          parser.parseString(data, function (err, result) {
            //先頭に_idを追加
            var result_clone = {};
            result_clone['_id'] = new db.bson_serializer.ObjectID();
            for (var i in result){
              result_clone[i] = result[i];
            }
            //xml fieldを追加
            result_clone['xml'] = data.toString();

            //検索のためにtagsに追加
            result_clone['tags'] = insertTags(result_clone);

            //insert実行後のcountと比較するためにsafe:trueを指定
            collection.insert(result_clone, {safe:true}, function(err,
              collection.count(function(err, count) {
                if(count >= condition) {
                  console.log("db.count() = " + count);
                  condition += 100;
                }
                if(count >= files.length){
                  db.close();
                  console.log(new Date());
                  console.log("db.count() = " + count + "; process.exi
                }
              });
            });
          });
        });
      });
    });
  });
});
```

Indexを使用したキーワード検索が可能に

MongoDBのメリットを検証する：結果

- スキーマレス
  - エレメント数の違うXMLのInsert

アプリ作成で検証

- Index付データのInsert/Updateが高速
  - RDBMSと比較してどうか
- JSONが標準語
  - 生産性が高いか

複数種類のxmlデータのインサートに対応可能なことを検証できた。

sandbox / src / ruby / insert_xml.rb

s-fujisaki June 27, 2012 initial commit

0 contributors

file | 20 lines (17 sloc) | 0.413 kb | Edit | Raw | Blame | History

```
#!/usr/local/bin/ruby
# -*- coding: utf-8 -*-

require 'crack'
require 'json'
require 'xmlsimple'
require "mongo"

puts Time.now
Dir::glob("/root/src/xml_/*.xml").each {|f|
  file = open(f)
  json = Crack::XML.parse(file.read)
  #puts "insert hash into mongodb "
  db = Mongo::Connection.new.db('test',:pool_size => 1000)
  db['batch_ruby'].insert(json)
  #f.each {|line| print line}
  file.close
}
puts Time.now
```

複数種類のxmlでも、単純なINSERTなら、20行程度で可能。

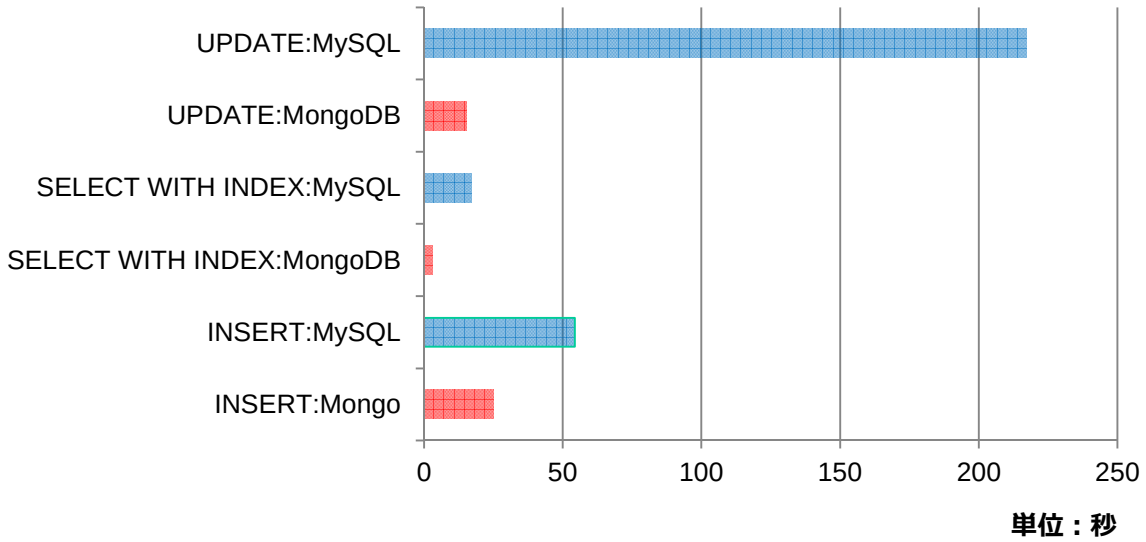
NRI 未来創発 Dream up the future.

MongoDBのメリットを検証する：結果

- スキーマレス
  - エレメント数の違うXMLのInsert
- Index付データのInsert/Updateが高速
  - RDBMSと比較してどうか
- JSONが標準語
  - 生産性が高いか

テストデータで
性能検証

MongoDBが全般的に早く、特にUPDATEと
SELECTで顕著。

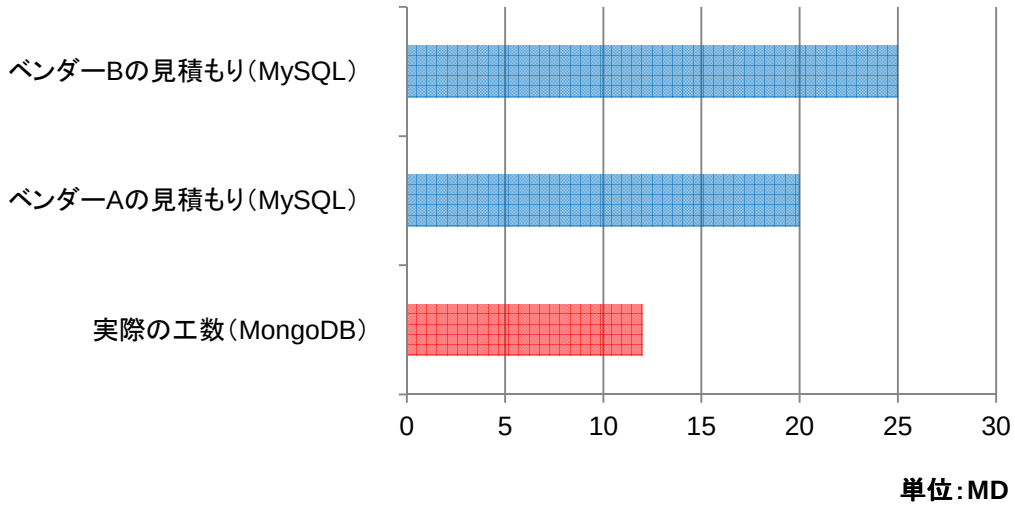



```
[root@vadev src]# ruby bench_insert.rb
==== Clean collection and table ====

==== Create random data ====

Bench: MySQL vs MongoDB
100000 times:
                                        Results
-------------------------------------------------
INSERT:MongoDB                           25.060
INSERT:MySQL                             54.375
SELECT WITH INDEX:MongoDB                 2.950
SELECT WITH INDEX:MySQL                  17.011
UPDATE:MongoDB                           15.193
UPDATE:MySQL                            217.105

[root@vadev src]#
```

MongoDBのメリットを検証する：結果

- スキーマレス
  - エレメント数の違うXMLのInsert
- Index付データのInsert/Updateが高速
  - RDBMSと比較してどうか
- JSONが標準語
  - 生産性が高いか

見積もり依頼を出して検証



設計書を作成し、見積もり依頼を行った。
ただし、RDBMSであるMySQLを使用することを条件に入れている。
パートナー会社Aの見積もり：20MD
パートナー会社Bの見積もり：25MD

実際にかかった工数：12MD

本日のまとめ

MongoDBとは、ドキュメント指向データベースで、NoSQLの一つに分類される

小規模な更新が大量に起こるデータや、スキーマの定義しにくいデータの管理に向いている

オートシャーディングにより、自動的にスケーリングする機能を備えている

世界的にみると実用フェイズに入っている。日本での事例も増えてきており、普及フェイズに入ろうとしている。

検証の範囲では、性能・生産性ともにRDBMSよりも高い。

本資料に掲載されている会社名、製品名、サービス名は各社の登録商標、又は商標です。

オープンソースまるごと

OpenStandia™

Open Source Technology

お問い合わせは、NRIオープンソースソリューションセンターへ

ossc@nri.co.jp

http://openstandia.jp/

NRI 未来創発 Dream up the future.