

MongoDB Tokyo 2013

MongoDBの位置づけと事例紹介

2013年12月12日

野村総合研究所 情報技術本部

OpenStandiaチーム

テクニカルエンジニア

渡部 徹太郎



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センター(OSSC)

Mail : osscc@nri.co.jp Web : <http://openstandia.jp/>



- MongoDBの位置づけ
- MongoDBの特徴
- MongoDBを使う上での注意点
- 代表的なユースケース
- 海外事例
- 国内事例
- 製品採用状況
- サポートサービス
- まとめ

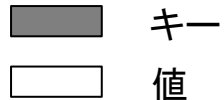
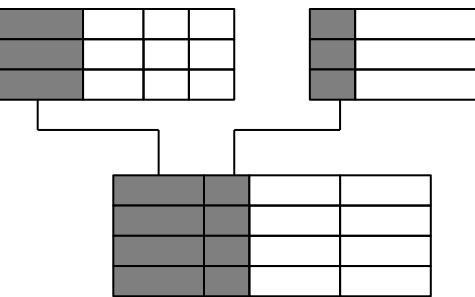
```
{  
  名前 : { 漢字 : “渡部 徹太郎”,  
            かな : “わたなべ てつたろう” },  
  所属 : “野村総合研究所 OpenStandiaチーム”,  
  学生時代 : “情報検索の研究( @日本データベース学会 )”,  
  仕事 : { 昔 : “証券会社のオントレシシステムのWeb基盤”,  
            今 : “オープンソース全般のサポート／コンサル” },  
  MongoDB関連 : [ “日本MongoDBユーザ会所属”,  
                   “丸の内MongoDB勉強会の主催者”,  
                   “Gihyo.jp「MongoDBでゆるふわDB体験」”,  
                   “日経SYSTEMS8月号 「ドキュメント指向データベース」” ],  
  Facebook ID : fetarodc  
  twitter : @fetarodc  
}
```

MongoDBの位置づけ (1/3)

● MongoDBはドキュメント指向データベース データベース

RDBMS

MySQL, PostgreSQL,
Oracle, SQL Server, DB2

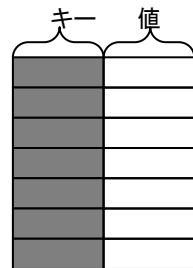


NOSQL (Not Only SQL)

KVS

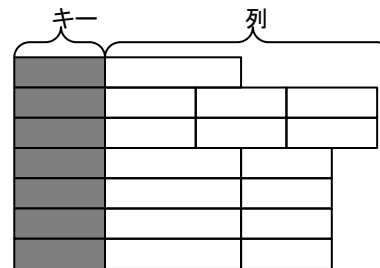
キーバリュ型

Dynamo, Riak,
memcached,
Redis



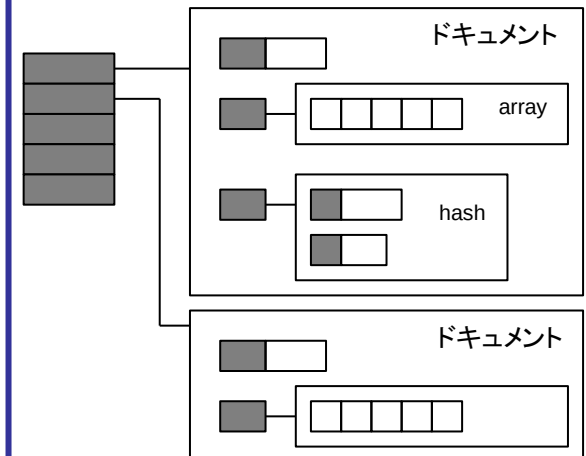
列指向

Bigtable, Cassandra,
HBase



ドキュメント指向データベース

MongoDB, CouchDB,
CouchbaseServer



MongoDBの位置づけ (2/3)

- ドキュメント指向データベースとは
 - ▶ データを階層構造のドキュメント (≒JSON) で扱う
- JSONとは
 - ▶ ハッシュと配列をネストして使うことができる
 - ▶ XMLよりシンプルに表現できる
 - ▶ 読みやすく直観的
- JSONの例

```
{  
  ID : 12345 ,  
  name : "渡部",  
  address : {  
    Company : "日本",  
    City : "東京",  
    ZipNo : "045-3356",  
  }  
  friendID : [ 3134 , 10231 , 10974 , 11165 ] ,  
  hobbies :  
    [  
      { name : "自宅サーバ", "year" : 6 } ,  
      { name : "プログラミング", "year" : 10 } ,  
      { name : "麻雀", "no" : 16 }  
    ]  
}
```

キーと値

ハッシュ

配列

ハッシュの配列

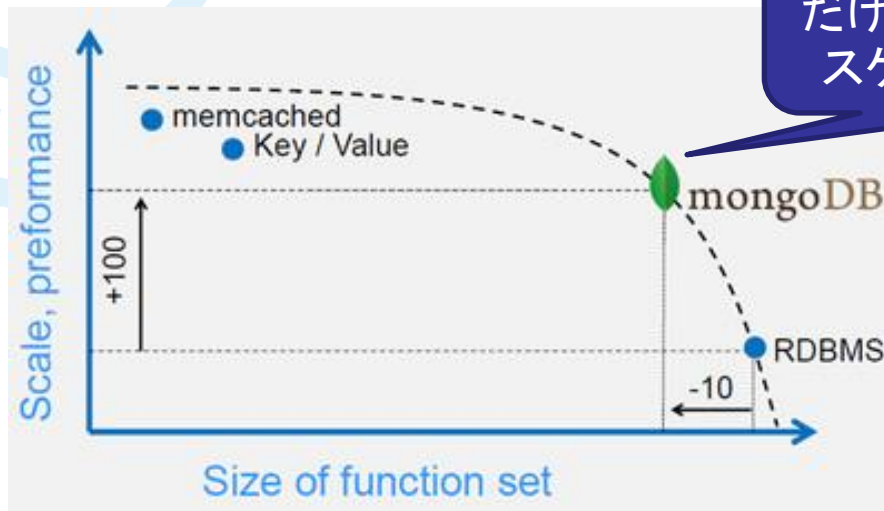
●ドキュメント指向データベースの比較

	MongoDB	CouchDB	Couchbase Server
データ構造	データベース └コレクション └ドキュメント(JSON)	データベース └ドキュメント(JSON)	バケット └ドキュメント(JSON)
インデックスの生成	インデックスを張りたいキーを指定する	MapReduce関数を用いたビューを作成することで対応	
クエリー	動的クエリ。SQLライクな記述が可能。	基本的なCRUD以外は、静的クエリ。上記MapReduceで作成したビューへのアクセスがクエリにあたる。	
主なインターフェース	独自プロトコル。各言語用専用ドライバを利用。	REST (HTTP)	memcachedプロトコル
レプリケーション	シングルマスタ型レプリケーション (1ノードにしか書き込めない)	マルチマスタ型レプリケーション (複数ノードに書き込める)	
開発言語	C++	Erlang	C,C++,Erlang

MongoDBの特徴 (1/6)

● MongoDBを一言でいうと

▶ RDBMSとKVSのいいとこどり



RDBMSから機能を少しだけ削ることにより、高いスケーラビリティを獲得

(図の出典: Meiko Hori, Jonathan Reams, 「MongoDBが目指すもの」より
<https://wiki.mongodb.com/pages/view.page.action?pagelId=20743144>)

▶ MongoDBの特徴

リッチなデータ

柔軟なクエリ

RDBMSの
いいところ

水平分散

レプリケーション

KVSの
いいところ

スキーマレス

使いやすい

多機能

MongoDBの特徴

MongoDBの特徴 (2/6)

リッチなデータ

- ▶ 前述の通りJSONは階層構造のリッチなデータモデル

柔軟なクエリ

- ▶ 動的なクエリ

- ✓ SQLライクな扱いやすいクエリで扱いやすい。
- ✓ 動的に作成(事前に定義不要)

```
db.person.find( { "name" : "watanabe", "age" : 30 } ).limit(3)
```

例) コレクションpersonに、“name”が“watanabe”で、“age”が30のドキュメントを3つだけ取得したい

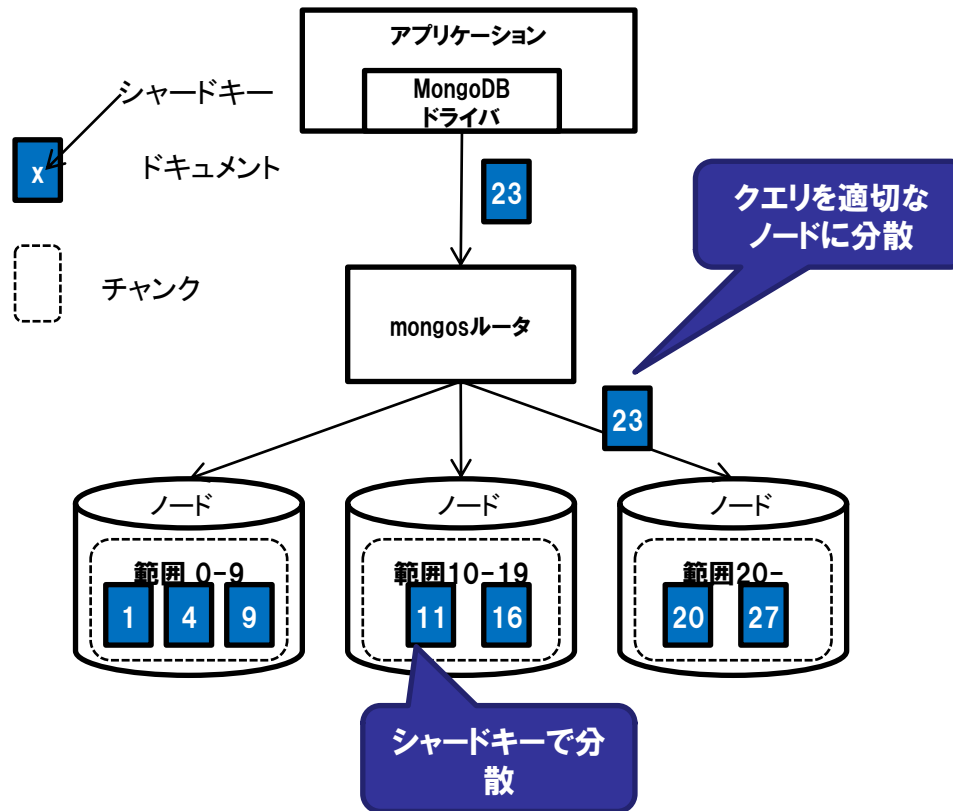
- ▶ 柔軟なインデックス

- ✓ 主キー以外にもインデックスを作成可能。複数のキーでインデックスを作ることも可能。

水平分散

▶ 水平分散(シャーディング)が簡単

✓ キーによってデータをノードに分散することができる。また、ノードを動的に追加し、データの自動バランシング機能もある。

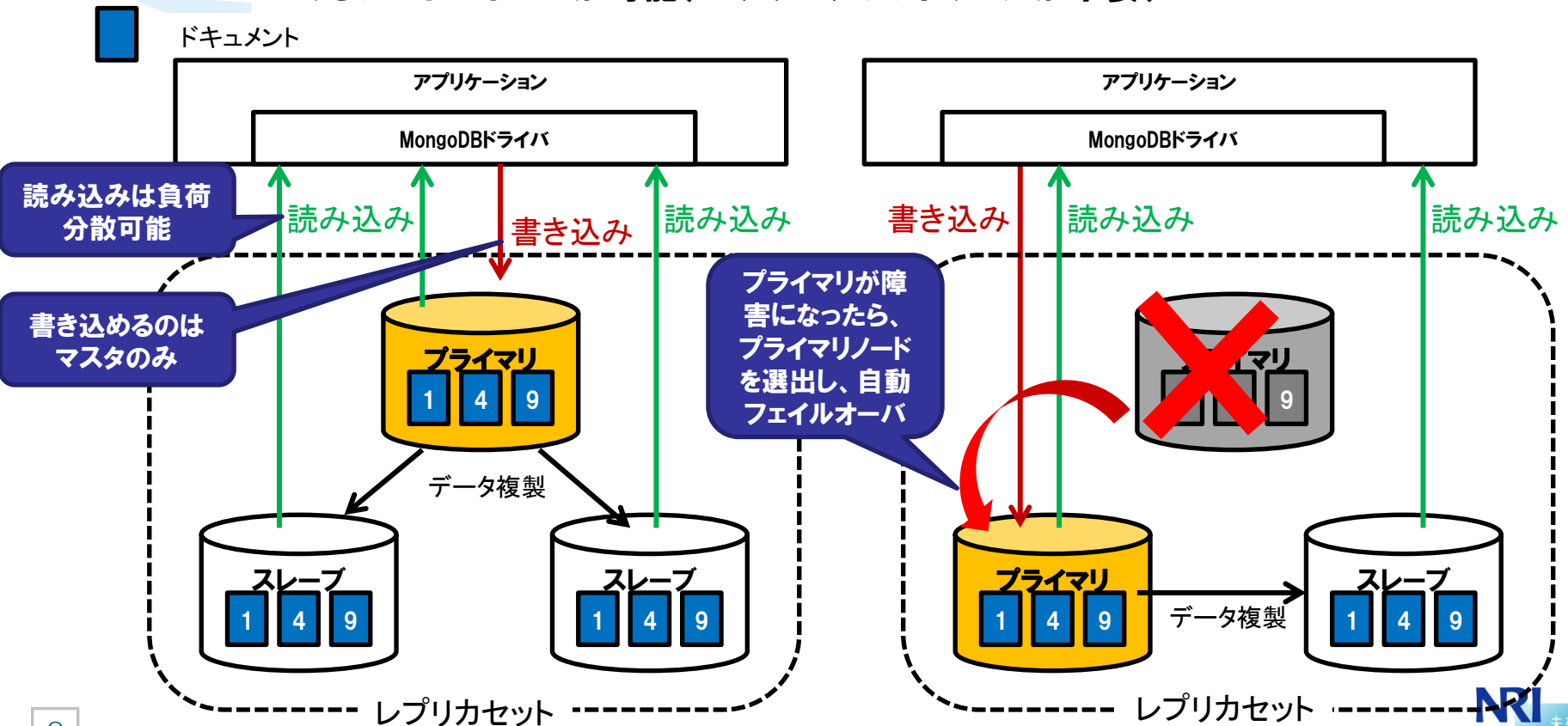


MongoDBの特徴 (4/6)

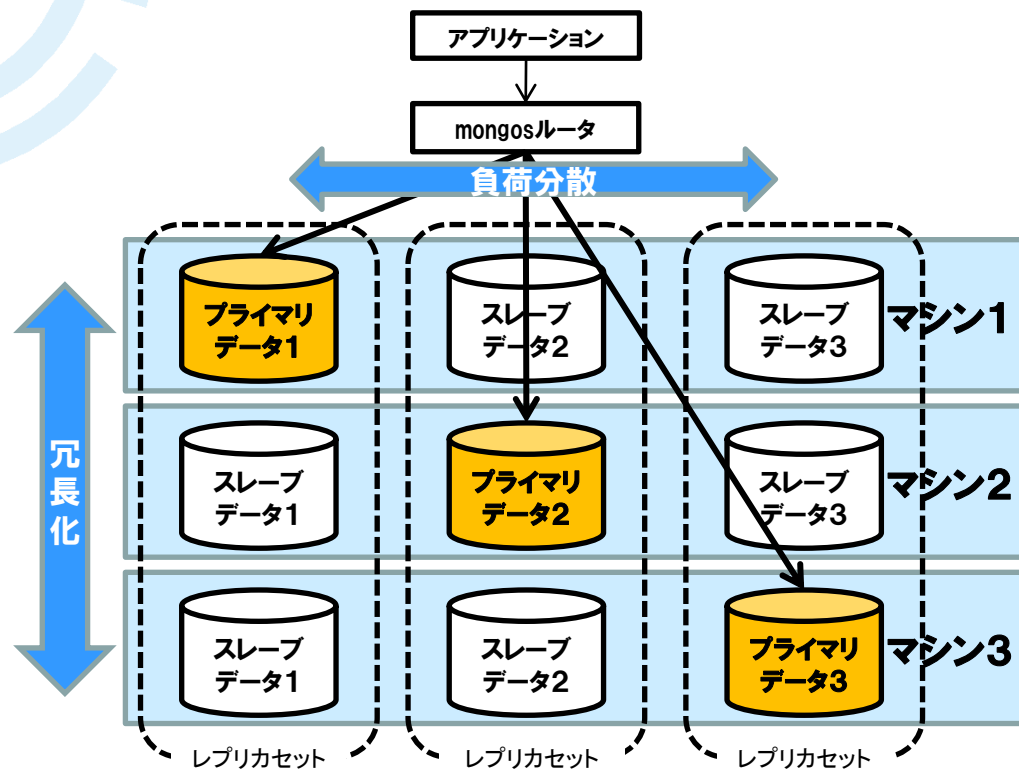
レプリケーション

▶ 複製(レプリケーション)が簡単

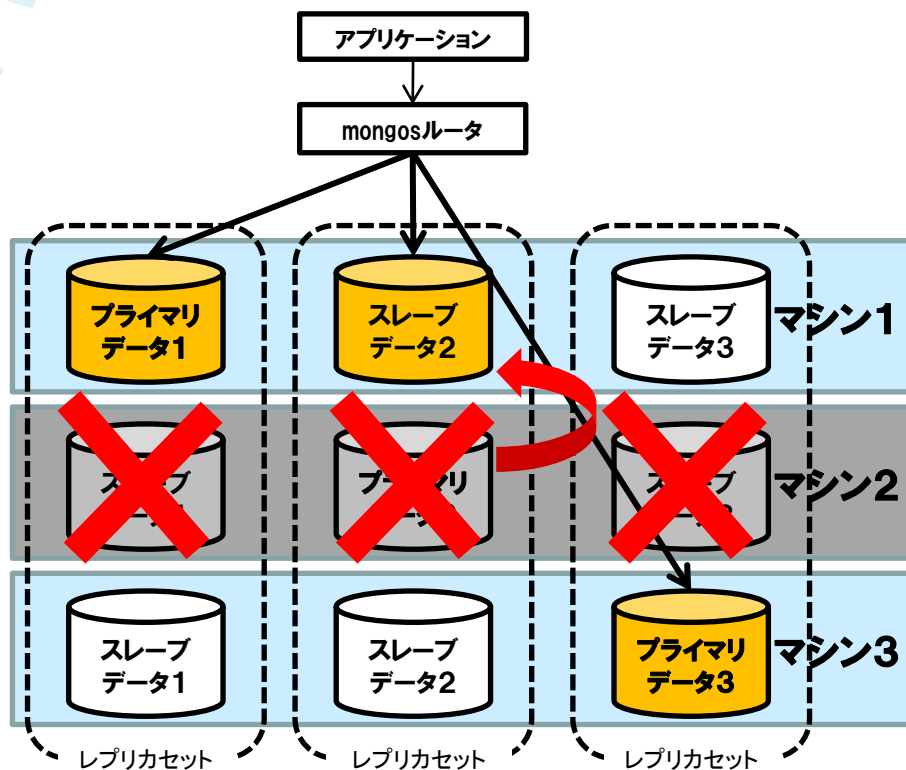
- ✓ 簡単なコマンドで、マスタスレーブ型のレプリケーションを構築可能。
- ✓ シャーディングと組み合わせることも可能
- ✓ MongoDBドライバが自動的に書き込み先を切り替えるため、仮想IPなどを用意しなくてもフェイルオーバーが可能(≒クラスタソフトウェアが不要)



- ▶ レプリケーションとシャーディングを組み合わせて、負荷分散と冗長化を両立



- ▶ レプリケーションとシャーディングを組み合わせて、負荷分散と冗長化を両立



MongoDBの特徴 (6/6)

スキーマレス

- ▶ スキーマレスデータを扱える
- ▶ テーブル定義など無しに、すぐにデータをCRUDできる

使いやすい

- ▶ セットアップが簡単
- ▶ 便利で使いやすいコンソール
- ▶ RDBMSを使っていた人がとっつきやすいクエリやデータ構造

多機能

機能	説明	ユースケース
ジャーナリング	ドキュメント単位で書き込みの一貫性が保持できる。	書き込みを保証する必要があるアプリケーション
GridFS	大容量ファイルを扱うことができる	大容量ファイルの管理
地理空間インデックス	2Dや3Dのデータを格納し、それに対して近傍などの検索を書けることができる。	地図アプリのデータベース
キャップ付きコレクション	期限を指定したデータを持つことができる	ログ保管
集計機能	SQLのグループ関数のように集計できる。またmap reduceによる集計も可能	データの集計
統計情報出力	様々なサーバの統計情報をJSON形式で出力することができる	運用監視ツールとの連携

● トランザクションが無い

- ▶ MongoDBが複数のドキュメントを一貫性をもって更新する事ができない
- ▶ ミッションクリティカルで複数のテーブルの更新を保証しなければならないようなシステムでは、利用してはならない。

● 外部キー・結合が無い

- ▶ 他のドキュメントへの参照はアプリケーションで実装する必要がある。
- ▶ 当然ながら、外部キー制約もないため、テーブル間の整合性が重要なシステムには向いていない。
- ▶ 複数のドキュメントの内容を結合して取得することはできない。

● スキーマが無い

- ▶ どのようなキー名でデータが入っているかわからない。データ型もわからない。
- ▶ データ登録間違えの際にエラーが発生しない。
- ▶ 設計書を厳格に管理しないと、どのようなデータが入っているかわからなくなり、保守性の低下を招く恐れがある。

● インターネット上のWebアプリケーション

- ▶ 複雑なデータモデルを扱う **水平分散** **リッチなデータ**
- ▶ 利用ユーザの増加などによるトラフィックの増加が激しく、データの水平分散が求められることが多い
- ▶ 特にユーザログインがあるようなWebアプリケーション

● オンラインゲーム

水平分散

リッチなデータ

- ▶ Webアプリケーションと特性が似ている

● アジャイル開発

スキーマレス

多機能

使いやすい

- ▶ スキーマの変更頻度が非常に高い
- ▶ 直観的にデータを表現できる
- ▶ ORMマッパーを使う必要はなく、ライトウェイトなスクリプト言語 (javascript, ruby) との相性がよい。
- ▶ アプリ開発をサポートする機能が沢山ある

● データ分析

水平分散

柔軟なクエリ

▶ 大量のデータを扱う

✓ 様々なキーに対して、複雑なインデックスを張る必要があり

▶ 柔軟にクエリー組み立てる必要あり

✓ 集計等のリッチなクエリーが既存機能だけで実現できる

▶ 動的にクエリが書けるため、新規分析軸の導入が用意

● ログ情報の蓄積

スキーマレス

多機能

レプリケーション

▶ 様々なログの形式を蓄積可能

▶ キャップ付きコレクションで、古いログを自動的に消せる

▶ MongoDBにとりあえずログをためておき、そのほかの集計ミドルウェアで集計するという使い方がよい

▶ とりあえずレプリケーションしておけば、データは冗長化できる

代表的なユースケース (3/3)

● とりあえずスキーマレスデータを格納したい場合

▶ スキーマレスデータの例

スキーマレス**使いやすい****レプリケーション**

✓ Web画面のレイアウト、帳票のレイアウト

- レイアウトは階層的であり、スキーマも決まっていない。

✓ 施設の統合

- 施設ごとにドキュメントのフォーマットが異なり、項目も膨大であるため、とてもスキーマを定義できない。

▶ シンプルなJSONに変換するだけでよい

▶ スタートアップが簡単

▶ 簡単に冗長構成も構築することができるため、データがなくなる心配もない

● セキュリティサービスのビッグデータ解析にMongoDBを利用

● 課題

- ▶ 他技術ではスケーラビリティと機能がともに十分なものが無い
- ▶ Hbase/Hadoopでは複雑なクエリに対応できない
- ▶ Luceneではスケーラビリティに問題があり

水平分散

柔軟なクエリ

多機能

● 解決策

- ▶ MongoDBの自動シャーディングでスケーラビリティを実現
- ▶ 動的に柔軟なクエリが書けるため、新しい分析結果を追加する場合の開発が簡単
- ▶ 地理空間インデックスの利用により、地理的な観点でのデータ分析が容易に

● 結果

- ▶ レイテンシーを1/3に削減
- ▶ 動的スキーマの変更が可能になり、開発者の生産性が大幅に向上
- ▶ 市場に対する新しいサービスの投入が迅速化

(事例の出典 MongoDB, Inc <http://www.mongodb.com/customers/mcafee>)

海外事例 (2/4) orange

- 700万のウェブ・モバイルユーザに対する広範囲コンテンツ・サービス提供
- 課題
 - ▶ MySQLがスケーラビリティの上限に達して性能要件を達成できなくなった
 - ▶ RBMSでは非定型なメタデータの管理が困難
- 解決策
 - ▶ 性能とスケーラビリティに期待しMongoDBを導入
 - ▶ 60億におよぶ属性情報データの代わりに、1コンテンツを1ドキュメントにする構造を導入
- 結果
 - ▶ 秒間11万件以上のクエリに対応
 - ▶ 3年で200万ドル以上のコスト削減
 - ▶ 新規機能の導入のスピードが著しく早くなった
 - ▶ 新規プロジェクトでは全てMongoDBを利用する方針となった

水平分散

スキーマレス

(事例の出典 MondoDB, Inc <http://www.mongodb.com/customers/orange-digital>)

スキーマレス

使いやすい

水平分散

● 70以上の既存RDBMSに拡散した顧客情報をMongoDBで統合

● 課題

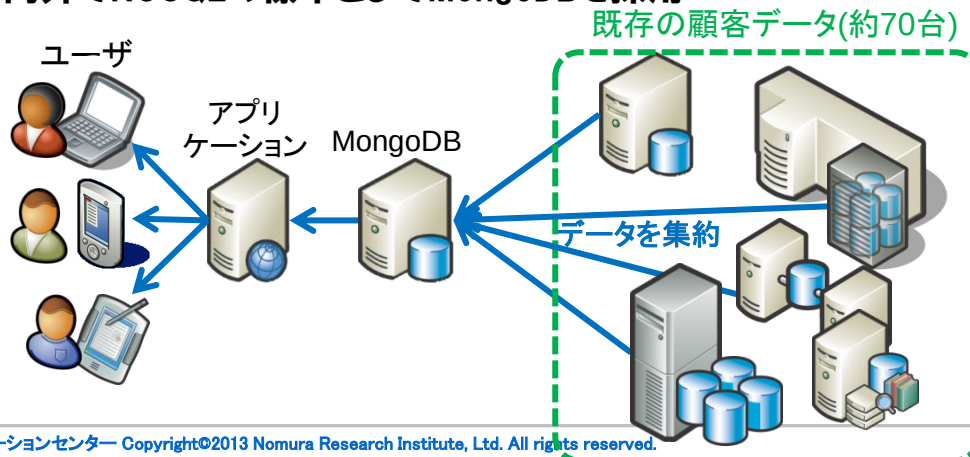
- ▶ 顧客データを個別に管理する70以上の既存RDBMSが存在し、そのデータを統合をしたいが、RDBMSでは工数がかかりすぎた
- ▶ モバイルで利用したいという要件があるが、端末の増加に合わせてスケールアップすることがRDBMSでは難しかった

● 解決策

- ▶ MongoDBがスキーマレスである特徴を利用して、既存のRDBMSの情報を統合して見せるアプリケーションを開発。
- ▶ MongoDBの開発容易性から、2週間でプロトタイプが作成でき、90日でリリースできた。

● 結果

- ▶ 10年間でできなかった顧客データの統合が実現。それも既存の顧客データには手を入れずに実現できた
- ▶ 巨額な投資が必要なRDBMS統合を、MongoDBを用いることで安価に、迅速に、達成できた
- ▶ 企業内外でNOSQLの標準としてMongoDBを採用



(出典 MongoDB Inc
<http://www.mongodb.com/press/metlife-leapfrogs-insurance-industry-mongodb-powered-big-data-application>)

● その他幅広い採用実績



(出典 MongoDB Inc <http://www.mongodb.com/press/>)

国内事例 (1/4) 大手ニュースサイト

● 大手ニュースサイトにてポータル

● 課題

レプリケーション

水平分散

- ▶ MySQLベースのストレージシステムがEOSの為、システム再構築を行う必要があった

● 解決策

- ▶ ポータルサイトは書き込みが少なく、読み出しが非常に多い非対称なクエリバランスである事から MongoDBを採用した。

● 結果

- ▶ 性能検証の結果、キャッシュ層が必要無い程の性能が確認できた。
- ▶ レプリケーションによるデータ冗長性、安全性も優れていた事から RDBMSから完全に脱却した。
- ▶ システム全体としては、様々な初期トラブル(CPU過負荷など)に見舞われたが ロジック過負荷になってもMongoDBの超えることは無かったためデータ破壊など致命的な状態には至らなかった。

国内事例 (2/4) ビデオオンデマンドシステム

● メタデータやユーザの行動履歴の格納

● 課題

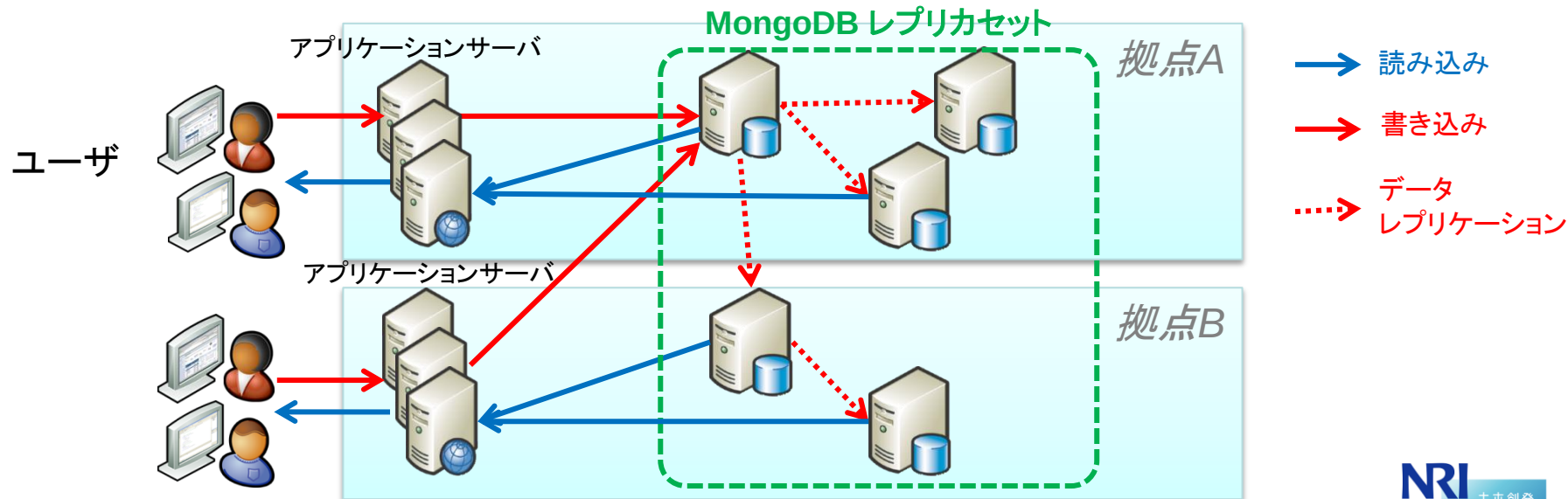
- ▶ ビデオのメタデータやユーザの行動履歴等、高スループットで高更新が求められるデータを格納したい
- ▶ データ毎に格納すべき情報が異なるため、スキーマレスで格納したい
- ▶ 拠点間をまたがってデータをレプリケーションしたい
- ▶ 商用RDBMSを利用していたが、ライセンス費用が高かったため、低コストで実現したい

スキーマレス

レプリケーション

● 解決策

- ▶ MongoDBのレプリカセット機能を用いて、拠点間でデータをレプリケーション。拠点ごとに読み込みを分散させると同時に、冗長性も確保。
- ▶ 課金などのミッションクリティカルな部分は、RDBMS (PostgreSQL) を利用。SQLとNOSQLのハイブリッド構成



● Webシステム作成案件で可用性を求めてMongoDBを利用

● 課題

レプリケーション

スキーマレス

- ▶ 新規に構築するWebサービスにおいてダウンタイムレスでレプリケーションによる高可用性を実現したい。
- ▶ RDBMSではレプリケーションを行うにはそれなりの手間が必要で、ダウンタイムレスでの自動フェイルオーバーは難しい
- ▶ 新規案件であるためスピード感のある開発が求められた

● 解決策

- ▶ MongoDBを採用し、レプリカセットの構成を実現

● 効果

- ▶ スキーマレスの利点を生かし作りながら考えることにより、短期間で開発を完了できた。
- ▶ レプリカセットによりシンプルな構成でダウンタイムレスの高可用性を実現できた。



● スタートアップ企業にてRuby on Railsのバックグラウンドで採用

● 課題

スキーマレス

水平分散

多機能

- ▶ 機能追加・仕様変更が多い
- ▶ 新規Webサービスではデータサイズの見積りが難しい。
- ▶ 位置情報(経度・緯度)の扱いに適したデータベースを探していた。
- ▶ Ruby on Railsで利用できるDBを利用したい

● 解決策

- ▶ MongoDBを採用

● 効果

- ▶ 日常的に起こる機能追加・仕様変更にも素早く対応できた。
- ▶ ユーザー数増加に伴うデータサイズの増加にも対応できるので、安心してサービス成長に取り組める。
- ▶ 地理空間インデックス機能を使って、位置情報を使用するクエリを簡単に実装できた。

- OSSのバックグラウンドDBとして利用 **水平分散** **レプリケーション**
- 今までRDBMSだけをサポートしていたOSSも、近年MonogDB接続用のプラグインを追加で開発している。
 - ▶ Jaspersoft (レポーティング・BI) 、Pentaho (BI)
 - ▶ Ruby on Rails (Web開発フレームワーク)
 - ✓ MongoDBを用いることにより、いちいちテーブルの定義を作成しなくてよくなる (rake db:migrateを打たなくてよい)
- MongoDBをデフォルトとしているOSSもある
 - ▶ fluentd (ログ収集基盤)

- 野村総合研究所はMongoDB Incと正式な代理店契約を結んでおり、MongoDBのサポートサービスを提供しております。
- MongoDB Incから技術的なサポートを受けることが可能です

● MongoDBのサポートメニュー

	Standard Subscription	Enterprise Subscription
サービスレベル	24時間365日	24時間365日
製品	MongoDB コミュニティ版	MongoDB Enterprise版
機能	—	コミュニティ版に以下の機能が追加 ・ケルベロス認証に対応。 ・MongoDB Monitoringサービスをオンプレミスで構築可能。モニタリングやバックアップが容易になる。 ・SNMPに対応して運用監視ツールとの連携が強化。 ・特定のOS(RHEL/CentOS, Ubuntu, SuSE, Amazon Linux) で動作テストし、認定している。

● 位置づけ

- ▶ MongoDBはRDBMSとは異なるものであり、RDBMSの代わりになるものではない
- ▶ RDBMSが不得意な水平分散が得意であり、RDBMSに近い機能性を備えている

● 事例

▶ 海外では

- ✓ 数多くの企業で大規模に採用されており、劇的な効果を出している
- ✓ Webや情報分析が多いが、保険業務などのミッションクリティカルな業務での採用もある

▶ 国内では

- ✓ Webやオンラインゲームを中心として、MongoDBを大規模に利用した企業が近年登場してきた

▶ 製品採用では

- ✓ BIのOSSをはじめとして、MongoDBをバックグラウンドDBに採用するOSSが増えている

▶ サポートサービスでは

- ✓ 日本でのサポートサービスも開始された

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



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



osscc@nri.co.jp



<http://openstandia.jp/>