

ブーム来てます！ MongoDB最新ユースケース & 事例紹介11社

野村総合研究所
オープンソースソリューション推進室



株式会社 野村総合研究所 オープンソースソリューション推進室

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

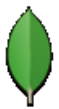


自己紹介



“ID” : “fetaro”

“名前” : “渡部 徹太郎”

“所属” : {  **OpenStandia**
Open Source Technology,  **mongoDB JP** }

“経歴” : “学生時代は情報検索の研究 (@日本データベース学会)”

“仕事” : { “昔” : “証券会社のオンプレシシステムのWeb基盤”,
“今” : “オープンソース全般” }

“特技” : [“サーバ基盤”, “Linux”, “KVM”, “ruby”, “MongoDB”]

“エディタ” : “emacs派”

“趣味” : “自宅サーバ”

“MongoDB関連” : {
— “丸の内MongoDB勉強会”
— “技評記事「[MongoDBでゆるふわDB体験](#)」”
— “日経SYSTEMS 8月号 「ドキュメント指向データベース」”
— “取材「[DBのプロに会いたい MongoDBの火付け役を目指して](#)」” },

“属性” : [“ギーク”, “スーツ”]

- MongoDBの基本(おさらい) (10分)
- MongoDBの必要性和流行 (5分)
- ユースケースと事例紹介 (20分)
 - ▶ ビックデータ処理
 - ▶ 非構造データ処理
 - ▶ その他
- MongoDBのヘビーユーザの紹介 (10分)
- 質疑応答 (5分)

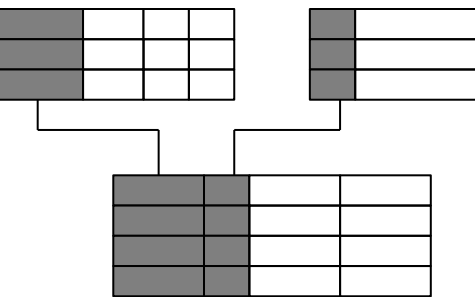
MongoDBの基本(おさらい)

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

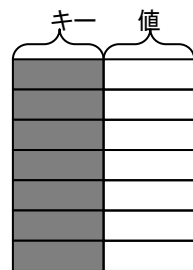


NOSQL (Not Only SQL)

KVS

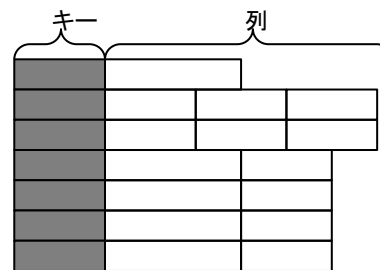
キーバリュ型

Riak, memcached,
Redis



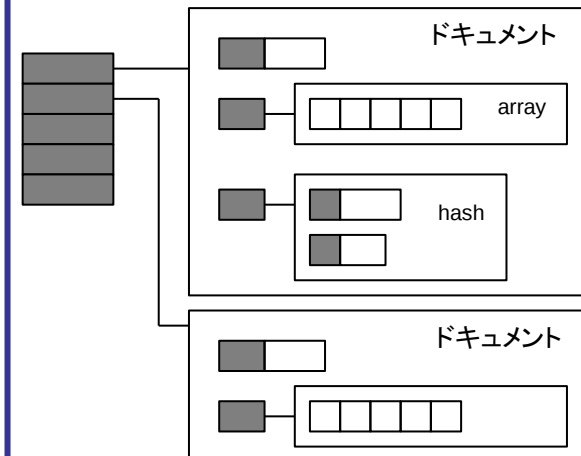
列指向

Bigtable, Cassandra,
HBase, DynamoDB



ドキュメント指向

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
- ▶ ネストが深くなる場合に、より効率的に扱える。

● 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 : "自転車", "year" : 6 } ,  
      { name : "インターネット", "year" : 10 } ,  
      { name : "読書", "no" : 16 }  
    ]  
}
```

キーと値

ハッシュ

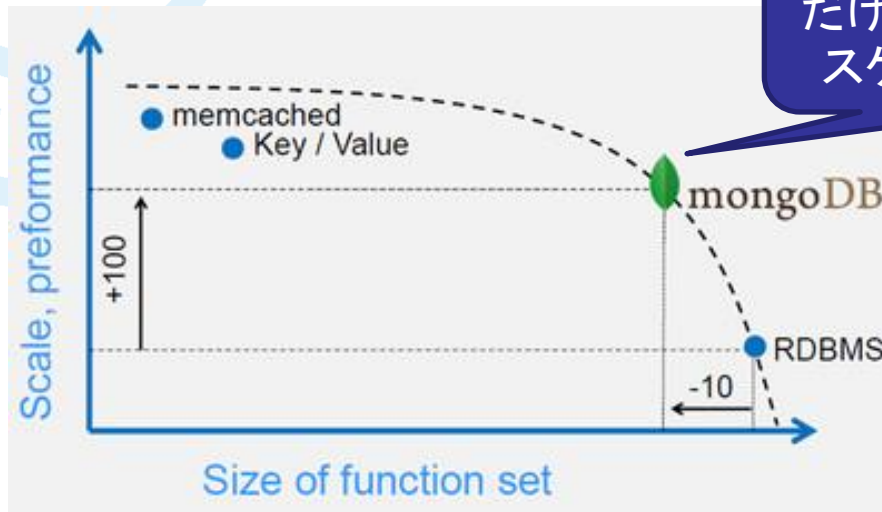
配列

ハッシュの配列

MongoDBの特徴 (1/7)

● 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

KVSと比較して

リッチなデータ

柔軟なクエリ

RDBMSと比較して

水平分散

レプリケーション

スキーマレス

使いやすい

多機能

MongoDBの特徴

リッチなデータ

▶ JSON (階層型データ) は、Key-Valueに比べて、リッチなデータモデル

```
{  
  id: 10  
  name: "watanabe"  
  friendId : [4, 7, 12, 19]  
}
```

key	value
id	10
10-name	"watanabe"
10-friendId-0	4
10-friendId-1	7
10-friendId-2	12
10-friendId-3	19

▶ このようにKey-Valueで1対多を表現しようとする、key名に"-1"等の配列の番号を持たせなければならず非常に扱いにくい。

柔軟なクエリ

▶表現力豊かなクエリ

- ✓SQLの文法に似せたクエリが扱いやすい。

```
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

- ✓動的に作成可能。事前に定義不要。
- ✓単純な条件検索だけでなく、集計等の高度なクエリも書ける。

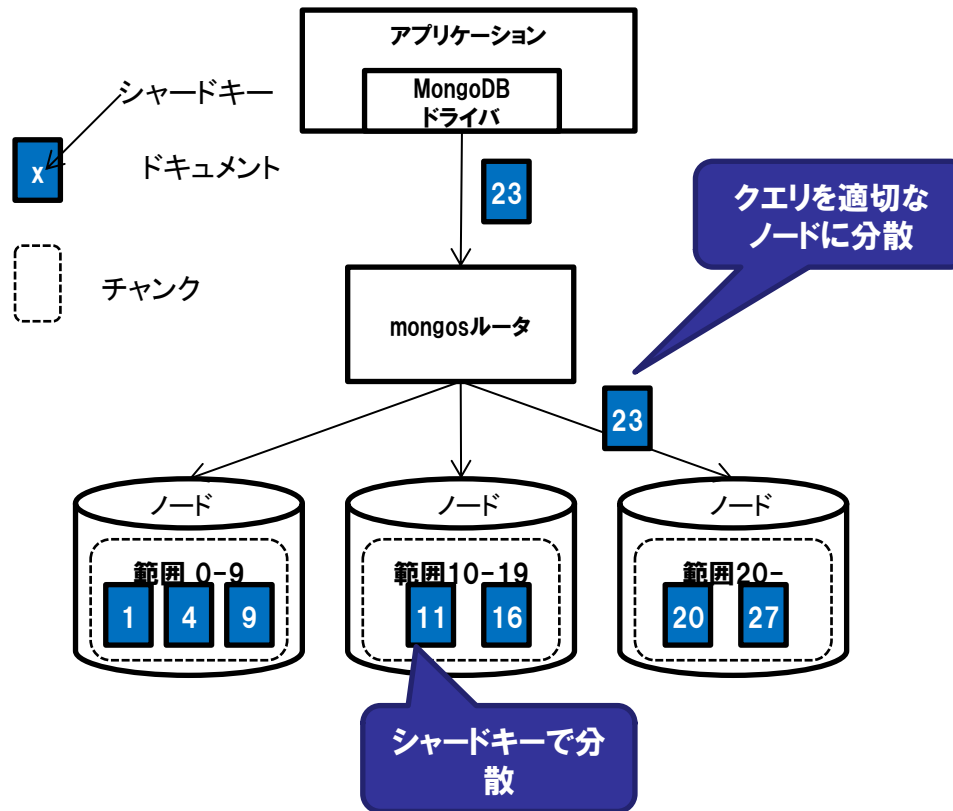
▶多様なインデックス

- ✓セカンダリインデックス:主キー以外でインデックスを作成可能
- ✓複合キーインデックス:複数のキーでインデックスを作成可能
- ✓マルチキーインデックス:配列の要素に対してインデックス作成可能

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。

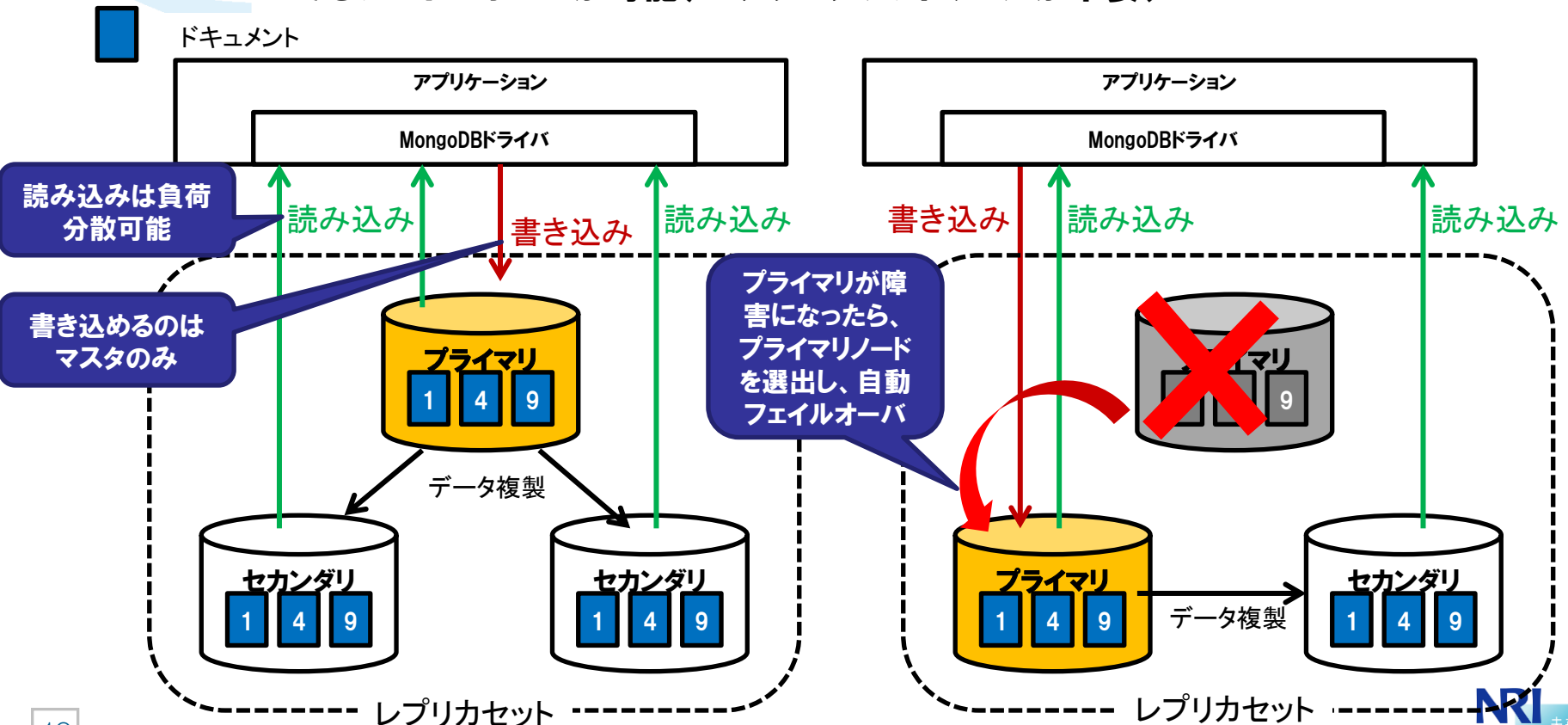


MongoDBの特徴 (5/7)

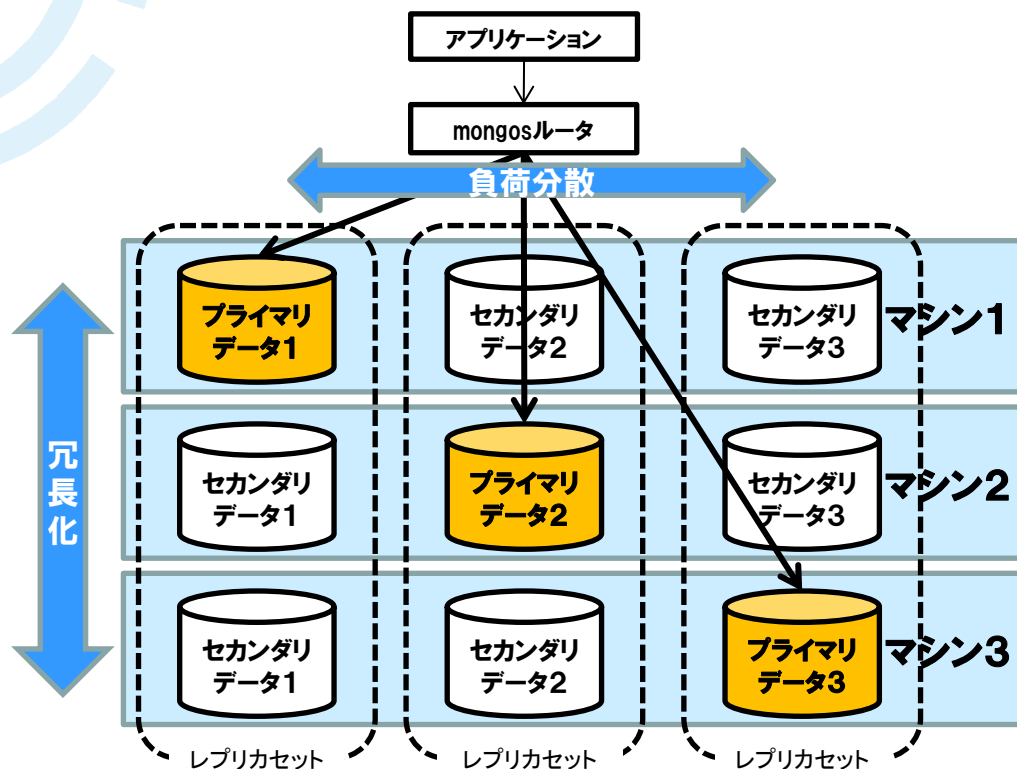
レプリケーション

▶ 複製(レプリケーション)が簡単

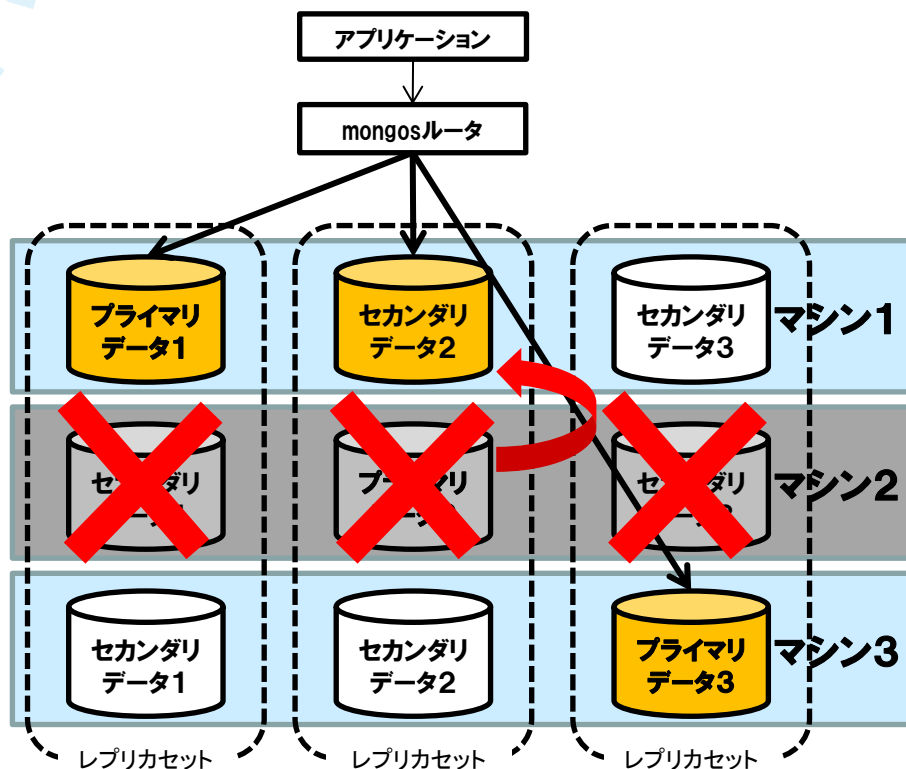
- ✓ 簡単なコマンドで、マスターセカンダリ型のレプリケーションを構築可能。
- ✓ シャーディングと組み合わせることも可能
- ✓ MongoDBドライバが自動的に書き込み先を切り替えるため、仮想IPなどを用意しなくてもフェイルオーバーが可能(≒クラスタソフトウェアが不要)



- ▶ 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 (6/7)

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

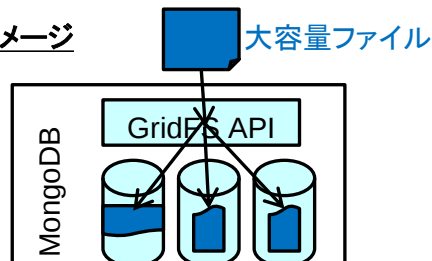
- ▶ セットアップが非常に簡単
 - ✓ OS毎にバイナリがあるため、ライブラリの追加インストール不要。
 - ✓ 起動までわずか3ステップ。
 - OS毎のバイナリをダウンロード
 - データディレクトリを作成
 - 起動
- ▶ RDBMSを使っていた人が使いやすいように作られている
 - ✓ データベース>テーブル(コレクション)>ドキュメント というデータ構造
 - ✓ SQLとMongoクエリ言語は大部分マッピング可能
 - ✓ インデックスもSQLと同じような宣言ができる
- ▶ 豊富なドキュメント・ノウハウ
 - ✓ 英語ではあるが公式ドキュメントは他のNOSQLに比べても豊富
 - ✓ 多くの人が使っているため、ノウハウが豊富。日本語のノウハウも多い。

MongoDBの特徴 (7/7)

多機能

分類	機能	説明	ユースケース
機能	GridFS	大容量ファイル (16M以上) を扱うことができる。 大容量ファイルをドキュメントに分割して格納し、アプリケーションには等価的なAPIを提供。	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、それに対して交点や近傍などの検索をかけることができる。 アプリでのつくり込不要。	地図アプリのデータベース
	キャップ付きコレクション	期限を指定したコレクションを作り、自動的に古いドキュメントを引き落とせる	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
耐障害	ジャーナリング	単一ドキュメントに対して、書き込みの一貫性が保持できる。	突然の電源停止等に対応したい
運用性	各種統計コマンド	様々なサーバの統計情報を取得するツールや、JSON形式で出力するコマンドがある	運用監視ツールとの連携 障害対応効率化
	MMS (MongoDB Management Service)	MongoDBの監視やアラート、自動バックアップ、ポイントインタイムリカバリ等ができるサービス	運用監視の仕組みを簡単に作りたい

GridFSのイメージ



地理空間インデックスを使ったデータに対するクエリ

```
db.map.find({loc:{$near:[ 139.701238, 35.658871 ]}})
```

\$nearにより、座標に近い点を検索

MongoDBを使う上での注意点

● 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。

● 集計が分散しない

- ▶ MongoDBのaggregationやMapReduceは完全には分散せず、なおかつmognodノードのCPU1コアを使って行われる
- ▶ 大規模な集計が必要な場合は他の集計ミドルとの連携が必要。Hadoopとは相性がよい

MongoDBの必要性と流行

はじめに (1/2)

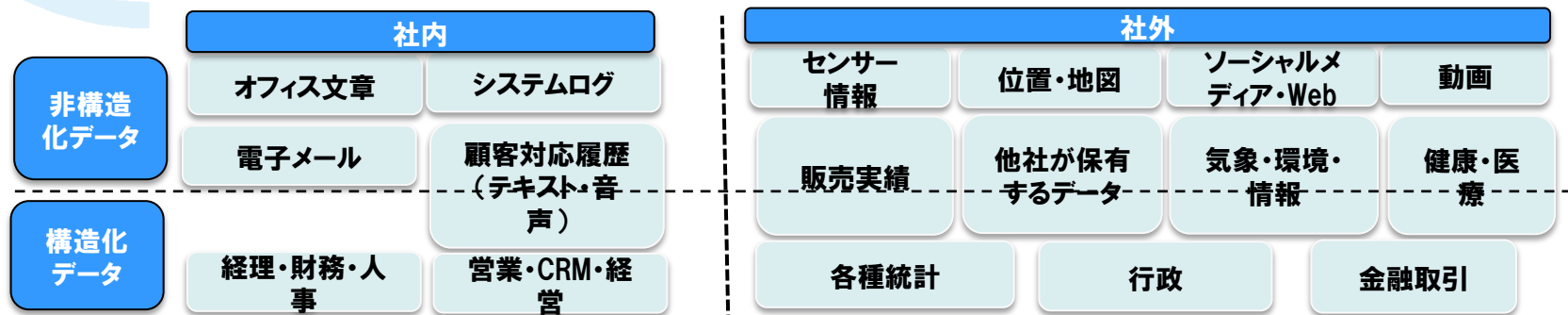
● 近年増えている「非構造データ」「ビッグデータ」処理はRDBMSには不向き

- ▶ 全体の90割のデータはこの2年で作られています→ビッグデータ
- ▶ 生成されるデータの8割が非構造データです→非構造データ

「MongoDB World 2014
key note」より

● 非構造データはRDBMSには格納できない

- ▶ 非構造データの例 「ビックデータ総覧」より

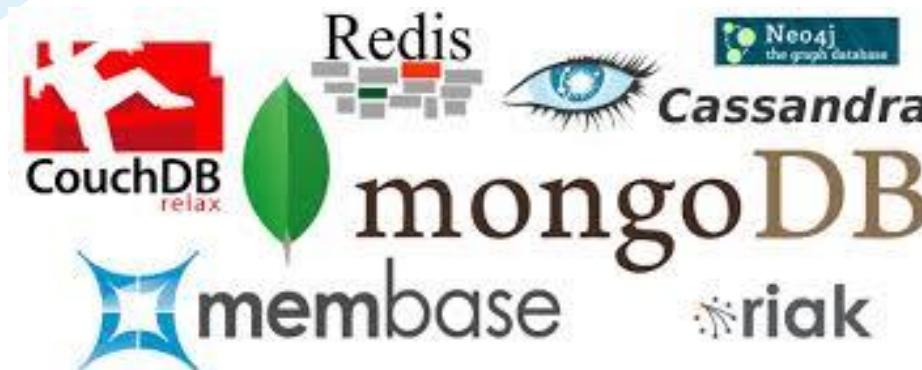


● ビックデータはRDBMSでは扱えない

- ▶ RDBMSのスケールアップでは限界がある。スケールアウト(水平分散)では高コスト。



- 「非構造データ」「ビッグデータの」処理であればNOSQLが得意です
- NOSQLを検討するのであれば、まずMongoDBを試してください
 - ▶ 世界には数多くのNOSQLがありますが、どれも独自のノウハウが必要で、学習コストは高いです



- ▶ OpenStandiaでは「まずMongoDB」を推奨しています。
 - ✓ MongoDBはRDBMSユーザに使いやすいように作られているため、NOSQLの中では**最も学習コストは低いものの一つ**です。
 - ✓ また数多くの機能があるため、多くの場合MongoDBを使えば、お客様のRDBMSの課題が解決すると思います。
 - ✓ 利用者が多いため、ノウハウが多い
 - ✓ AWS上で動作保証されている唯一のNOSQL
 - ✓ まずMongoDBを利用し、それでも課題が解決しない場合は、他のNOSQLを検討すべきと考えています

MongoDBの流行について(1/4)

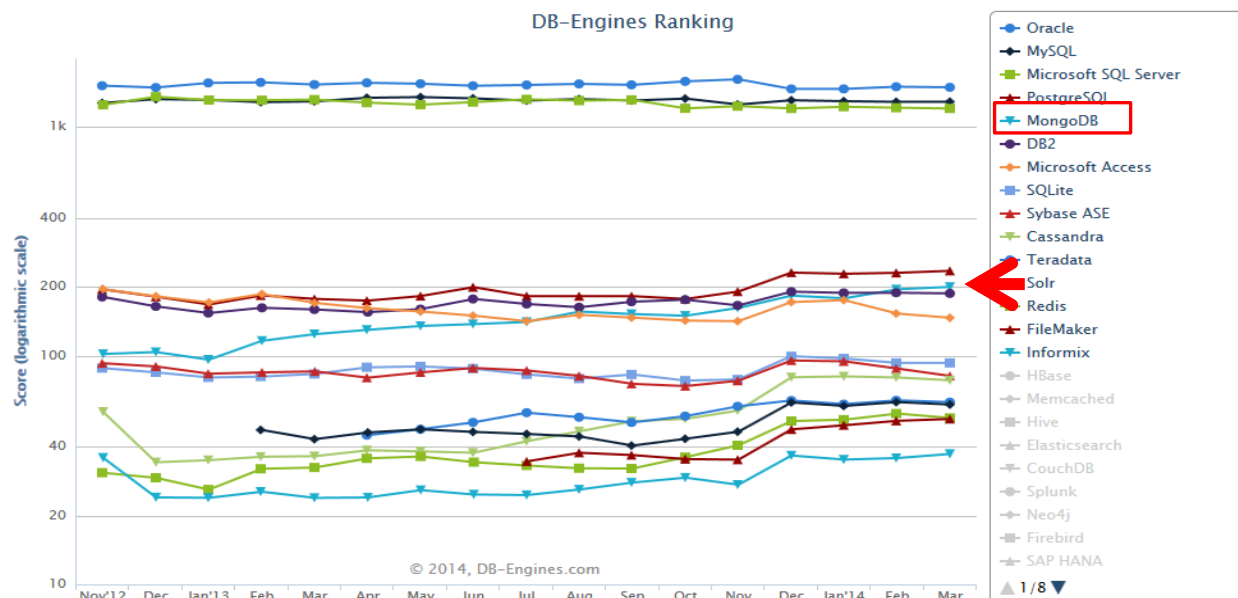
● db-engines.comでは上位にランキング(2014/3月)

216 systems in ranking, March 2014

Rank	Last Month	DBMS	Database Model	Score	Changes
1.	1.	Oracle 	Relational DBMS	1491.80	-8.43
2.	2.	MySQL 	Relational DBMS	1290.21	+1.83
3.	3.	Microsoft SQL Server 	Relational DBMS	1205.28	-8.99
4.	4.	PostgreSQL 	Relational DBMS	235.06	+4.61
5.	5.	MongoDB 	Document store	199.99	+4.81
6.	6.	DB2 	Relational DBMS	187.32	-1.14
7.	7.	Microsoft Access 	Relational DBMS	146.48	-6.40
8.	8.	SQLite 	Relational DBMS	92.98	-0.03
9.	9.	Sybase ASE 	Relational DBMS	81.55	-6.33

Wide column

March 2014



【指標の考え方】

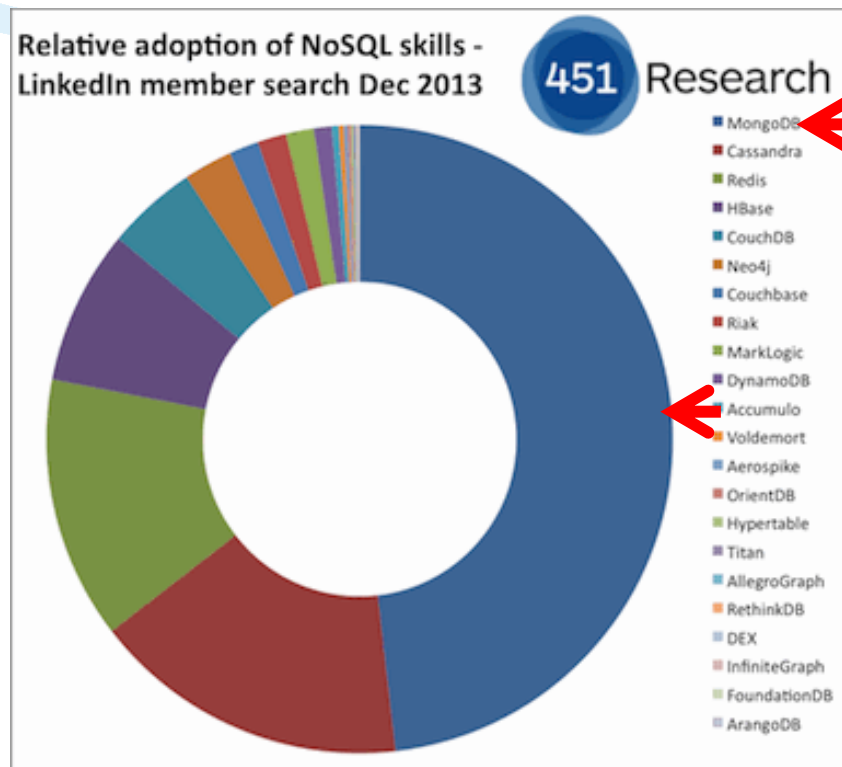
- ・ウェブサイトでのシステム名称の登場回数(Google, Bing)
- ・一般的な人気度(Google Trends)
- ・技術的なディスカッションの頻度(Stack Overflow, DBA Stack Exchange)
- ・求人サイトにおける募集スキル(Indeed, Simply Hired)
- ・プロフィール登場回数(LinkedIn)
- ・インストール数は考慮されていない

図の引用元:

<http://db-engines.com/en/ranking>

● 転職サイト (LinkedIn) における人気 NoSQLスキルランキングでは他を圧倒

- ▶ NoSQLではMongoDBの技術者が圧倒的に多い
- ▶ NoSQL技術の標準になりつつある



図の引用元:

http://blogs.the451group.com/information_management/2013/12/18/nosql-linkedin-skills-index-december-2013/

MongoDBの流行について (3/4)

● 採用企業600社以上



● 組み合わせられるソフトウェアも増えてきた

▶ Jaspersoft, Pentaho, qlik view, talend, Splunk

● 開発元であるMongoDB, Incは絶好調

- ▶ MongoDBはオープンソースなので誰でも開発できるが、現時点では実質MongoDB, Incが開発している。
- ▶ 2013年10月に150,000,000\$ (約150億円) の投資を受けた。
- ▶ 米MongoDB、1億5000万ドルの資金調達「Oracleに追いつく成熟度を目指す」

http://internet.watch.impress.co.jp/docs/news/20131007_618340.html

- ▶ 2014年6月には大規模なセミナーを開催、勢いは増すばかり

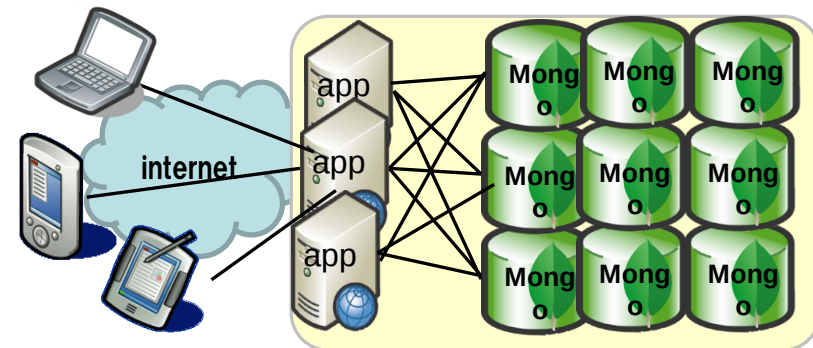


MongoDBのユースケースと事例紹介

ユースケース:ビッグデータ

● Webアプリ／オンラインゲーム

- ▶ 大量トラフィックのWebシステム／オンラインゲームでメインDBとして
- ▶ ユーザの増加に合わせて横に並べればOK
- ▶ リッチなデータ構造を扱えるので、複雑なアプリケーションにも対応



活用事例: orange (Webサービス事業)

700万ユーザを超えるWebサービスのバックグラウンドDBとして利用

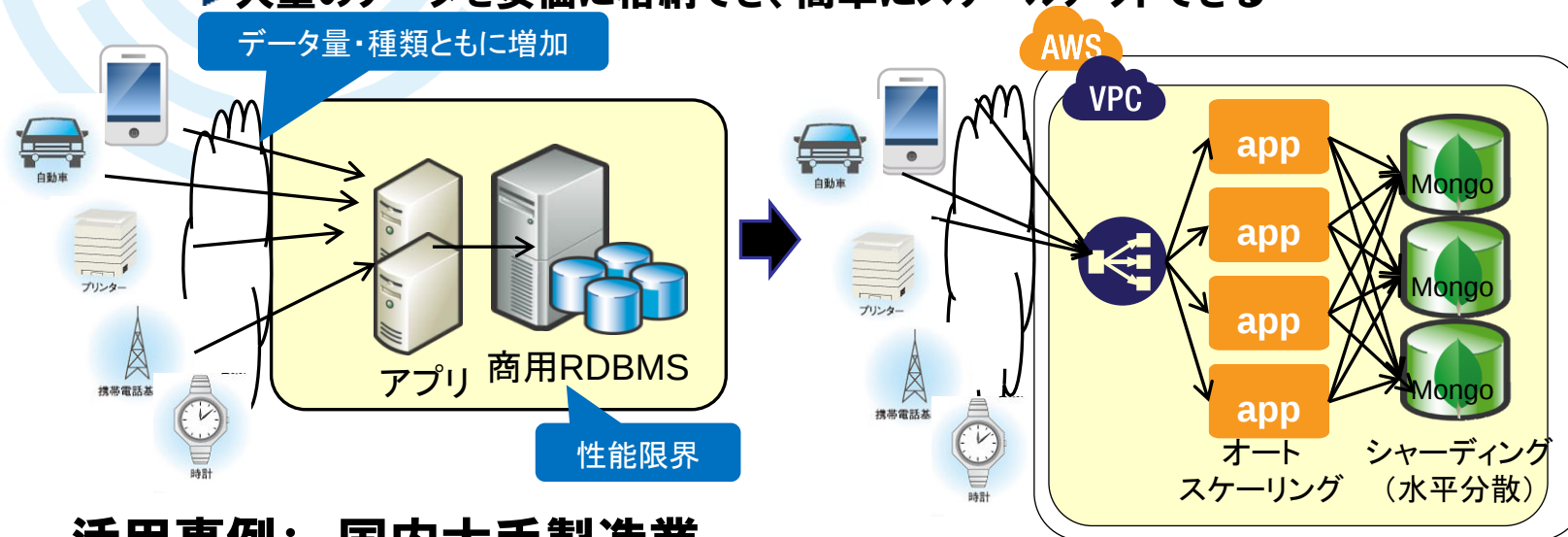
課題	選定理由・解決策	結果
• MySQLがスケーラビリティの上限に達して性能要件を達成できなくなった • RBMSでは非定型なメタデータの管理が困難	• 性能とスケーラビリティに期待しMongoDBを導入 • 60億におよぶ属性情報データの代わりに、1コンテンツを1ドキュメントにする構造を導入	• 秒間11万件以上のクエリに対応 • 3年で200万ドル以上のコスト削減 • 新規機能の導入のスピードが著しく早くなった • 新規プロジェクトでは全てMongoDBを利用する方針となった

ユースケース: ビックデータ

● M2M (マシン to マシン) IoT (Internet of Things)

- ▶ M2Mの標準規格になりつつあるJSONを格納できる
- ▶ 大量のデータを安価に格納でき、簡単にスケールアウトできる

データ量・種類ともに増加



活用事例: 国内大手製造業

M2MのメインDBを商用RDBMSからMongoDBに移行

課題	選定理由・解決策	結果
・商用RDBMSではスケールアップが高価すぎる	・スケールアウトで簡単に性能拡張できる ・AWS上で動作保証しているNOSQL ・RDBMSと同等の機能性(クエリ・集計)	・RDBMSと同じデータ構造、同じクエリが使えるため、移行が容易 ・3台で読込5000TPS、書込み1000TPSを達成

ユースケース:ビッグデータ

● 分析データ格納

- ▶ 安価なハードウェアに大量データを分散して格納できる
- ▶ データが大規模でなければ、MongoDBの集計機能で十分に要件を満たせる
 - ✓ 動的に柔軟にクエリー組み立てる事ができ、新規分析軸の導入が容易
 - ✓ 様々なキーに対して、複雑なインデックスを張ることができる
 - ✓ データのクリーニングせずにとりあえず入れて、集計時に必要な項目のみ集計する事が可能
 - RDBMSであればデータの挿入にひと手間あったが、それが不要
- ▶ 連携できるBIツールが多い (Pentaho, Jaspersoft等)
- ▶ データが大規模な場合は、他の分析基盤(Hadoop等)との連携を推奨

活用事例: McAfee

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	選定理由・解決策	結果
・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	・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	・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

情報システムリーダーのためのIT情報専門サイト

IT Leaders



データベース

[データベース記事一覧へ](#)

[事例ニュース]

アットホーム、不動産物件情報DBにNoSQLを採用しサービスをノンストップ化へ

今回刷新するのは、会員である不動産会社が70万件以上の物件データを検索するためのシステム。会員数が5万社を超え、取り扱い物件数が増加の一途をたどっていることから、データベースの強化が急務となっていた。

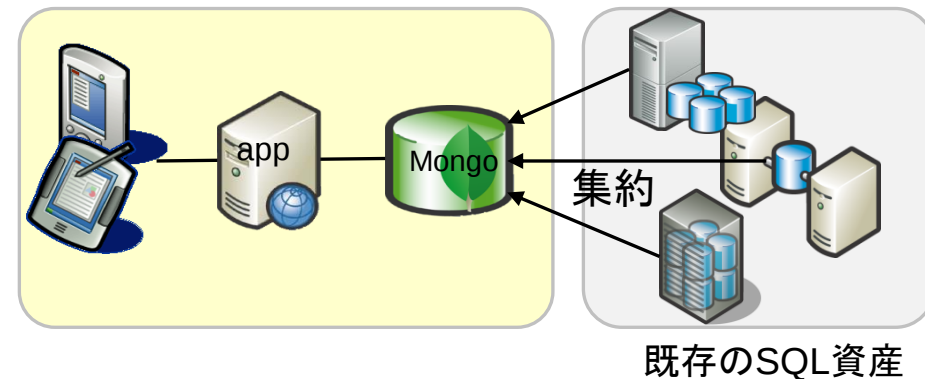
新システムは、プライベート・クラウド環境上に構築する。サーバーのメモリースロットにフラッシュストレージを16枚搭載することにより、システムの利用が集中する時間帯であっても平均2～3秒の検索速度を維持できる性能を備える予定だ。サーバーには、「IBM System x3850 X6」を3台導入する。

引用元: <http://it.impressbm.co.jp/articles/-/11657>

ユースケース: 非構造データ処理

● データ集約

- ▶ 既存のレガシーデータの集約基盤として利用。
- ▶ スキーマレスであるため、様々なスキーマのRDBMSからデータを集約することが容易。
- ▶ 集計したデータは、性能要件の高いモバイルアプリ等に提供



活用事例: Metlife (保険業)

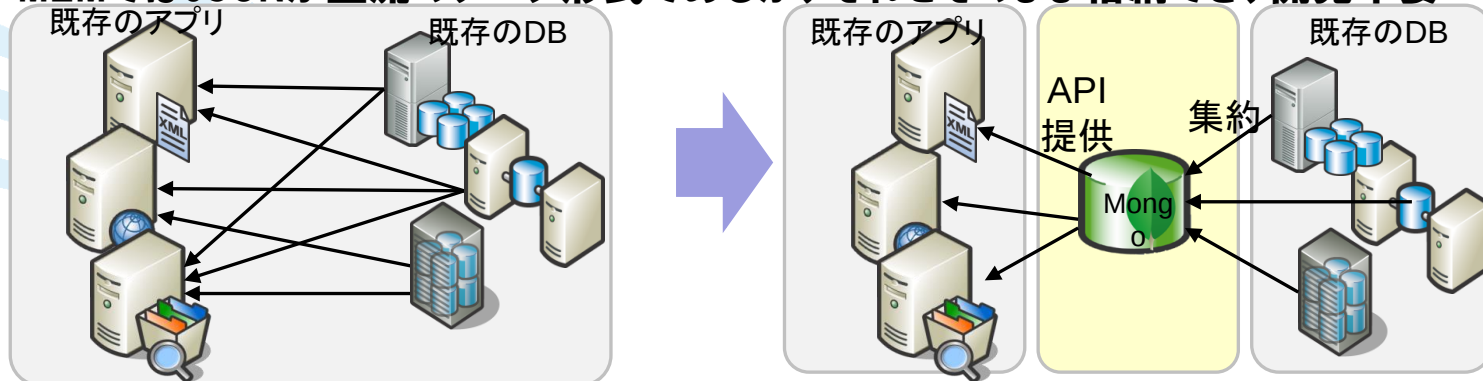
70以上の既存RDBMSに拡散した顧客情報をMongoDBで統合

課題	選定理由・解決策	結果
- 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	- 既存のRDBMSの情報を統合して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。	10年間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 - 過去同プロジェクトでは約\$25Mだったものが、約\$3Mで達成できた - 企業内外でNOSQLの標準としてMongoDBを採用

ユースケース: 非構造データ処理

● データハブ / M2M (Machine to Machine) / IOT (Internet of Things)

- ▶ コストのかかる商用製品の代わりに、MongoDBをデータハブとして利用
- ▶ M2MではJSONが主流のデータ形式であるが、それをそのまま格納でき、開発不要



活用事例: グローバル信託銀行X社 (金融業)

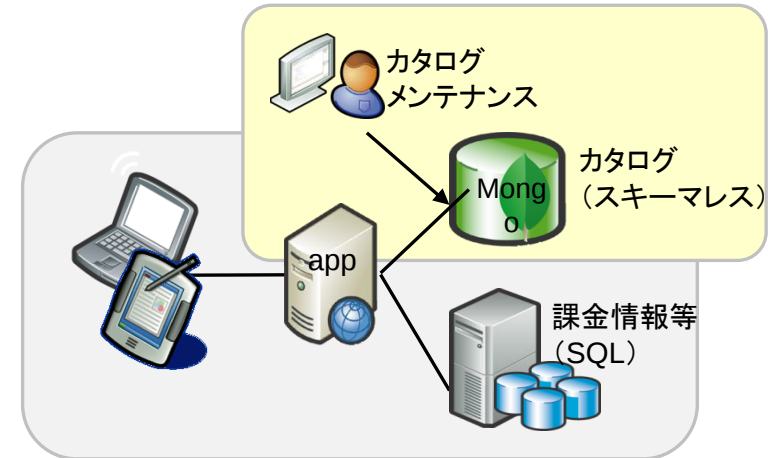
企業内でのデータアクセスを統合するために、データハブとして利用

課題	選定理由・解決策	結果
- データの複製がシステム間で無数に存在する - 一つのシステムでの変更が複数のグループに影響 - EDWのシステムレスポンスタイムが遅い - 頻繁にアクセスするデータは集中的に管理したい、というニーズ	- 動的なスキーマ: 必要な時だけデータを正規化する - 性能: 一つの論理DBで全てのデータを管理・運用 - シャーディング: スケールアウトによりデータを容易に追加	- 一カ所からバッチ、もしくはRESTでデータアクセス可能 - 顧客向けポータルサイトのレスポンスタイムが90%改善 - 開発期間の短縮データソースのエンハンスが容易

ユースケース: 非構造データ処理

● RDBMSとMongoDBのハイブリッド

- ▶ スキーマレスが向いているデータのみをMongoDBで処理することにより、スキーマ変更の負荷軽減や、性能向上が可能。
- ▶ 特に商品カタログ等のフォーマットが様々で更新頻度が多いデータに向いている。



活用事例: 野村総合研究所 (Sler)

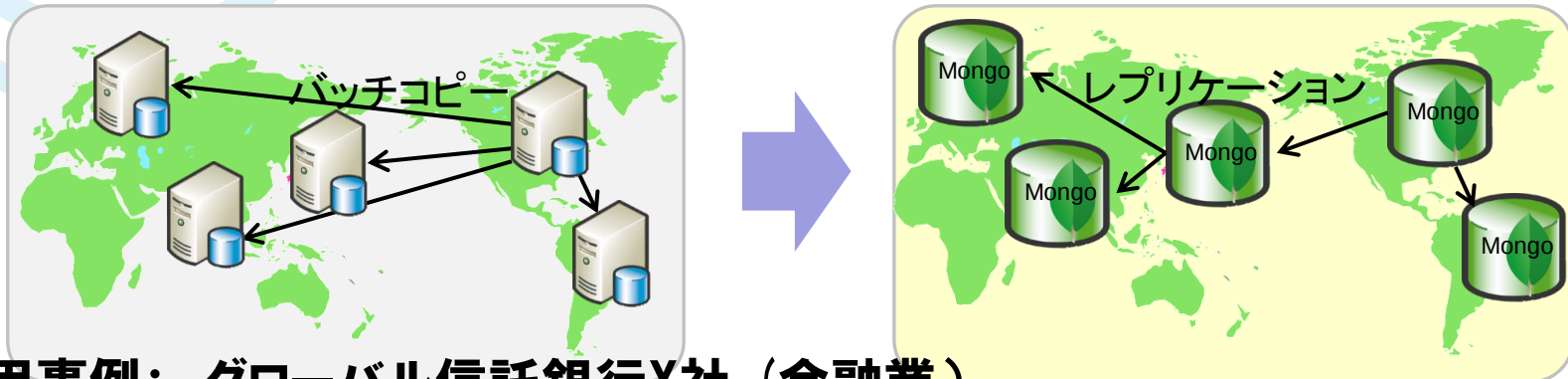
カード会社向けシステムで、スキーマレスデータ処理にMongoDBを利用

課題	選定理由・解決策	結果
- 要件として 「スキーマレスデータに対してSQLと同等のクエリをかけたい」 「NOSQLに不慣れな開発者にも簡単にクエリをかける」 というものがあつた - 従来のRDBMSでは上記の要件を満たせなかった。	- MongoDBであれば要件を満たしていた - 他のNOSQL技術と比較しても、利用実績が多く、流行してるため技術者も多かった - NOSQLの中では唯一社内のサポート体制が整っていた。	- 開発者が簡単にスキーマレスデータを操作でき、開発生産性を高く保つことができた - スキーマデータはRDBMS、スキーマレスデータはMongoDBという使い分けがうまくできた 冗長化の設計工数が、他のDB技術と比較し、飛躍的に少なくすんだ

ユースケース:その他

● 拠点間連携

- ▶ 各拠をまたがりレプリカセットを組むことにより、各拠点で同じデータが見れる
- ▶ レプリケーションの耐久性が高く、多少遅延のある通信経路でも構築可能
- ▶ レプリケーションの機能により、物理的に近い拠点からデータを複製することが可能



活用事例：グローバル信託銀行X社（金融業）

各拠点で迅速にローカルアクセス出来る様に、データをリアルタイムで配布

課題	選定理由・解決策	結果
・バッチ処理によるデータ配布の遅れが最大36時間に及ぶ ・同じデータのグローバル配信に複数課金されるSLA未達成による規制違反(罰金) ・同じを保有する20カ所の分散システムを管理する必要性	・自動レプリケーション: データ配信がリアルタイム、ローカルにデータを読む事が可能 ・キャッシュとデータベースの同期: キャッシュが常にアップデート ・単純なデータモデリングと分析: 変更が簡単、理解しやすい	・違反金\$40,000,000を5年間の間に節約 ・データ配信に対する課金は一回のみ ・グローバルにデータ同期と各拠点でのローカルReadが保証 ・統一したグローバルデータサービスに移行

ユースケース:その他

● ログ格納

- ▶ 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
- ▶ 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
- ▶ 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

● アジャイル開発

- ▶ アジャイル開発では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ため、データの扱いが簡単
- ▶ OR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

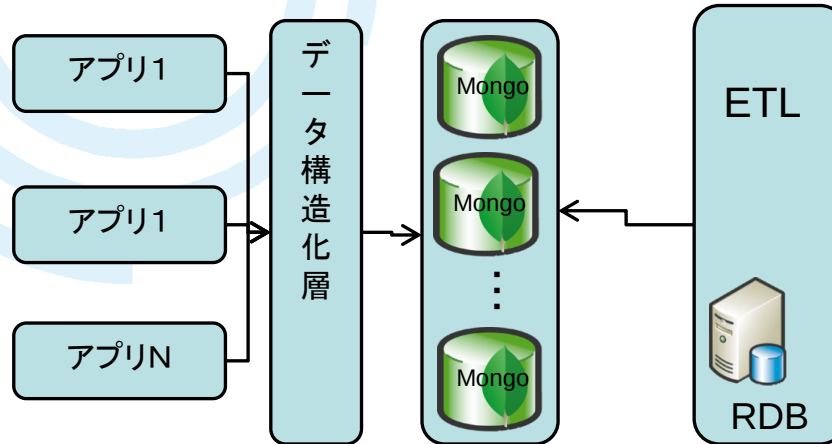
MongoDBのヘビーユーザ

- 社内PaasでMongoDBを標準に採用
- データのシフトが確実に起きているに対応する事が必要。
 - ▶ 従来の技術(RDB)は業務全体の99%を占めていながらも、新しい技術を取り込んでいく事に対する挑戦に取り組んでいる。
 - ▶ 大方のデータは構造型データであるが、最近急増しているのは非構造型のデータであり、クレジットカード等を含めたデバイスが精製するデジタルデータの大方は非構造型のデータである。
- MongoDBを導入した理由
 - ▶ 開発期間が短い→TCOが低い、DevOpsの連携、シャーディング、レプリカセット、安価なプラットフォームでよい
- MongoDBの効果
 - ▶ 平均的なInserts/secが大幅に向上
 - ▶ RDBでは達成出来なかったスピードでアプリ開発を達成
 - ✓ アプリケーションのPOC開発~Pre Productionが4ヶ月で達成
- MongoDBの課題
 - ▶ アプリケーションレイヤーはまだもう少し充実させたい
 - ▶ スキーマのフレキシビリティはいい面と悪い面がある。
 - ▶ BIツールが少ない
 - ▶ 標準が少ない。言語、機能セット、まだNoSQL業界はバラバラ

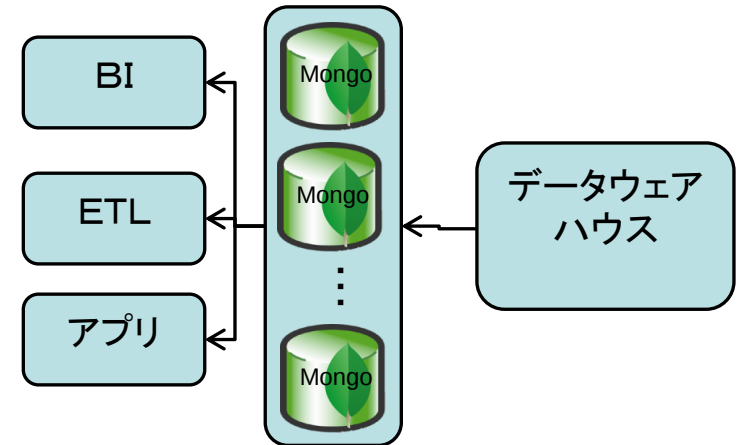
MongoDBのヘビーユーザ「Citi Bank」



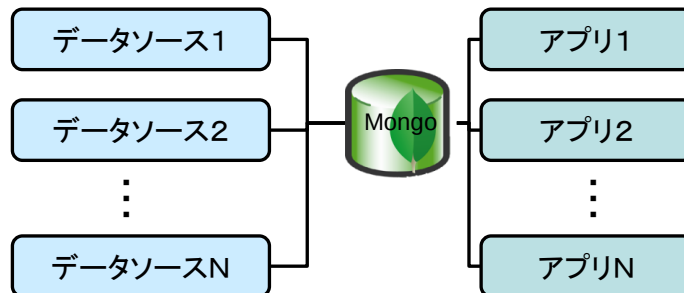
- MongoDBが有効に働くシステムパターンが幾つか確立している



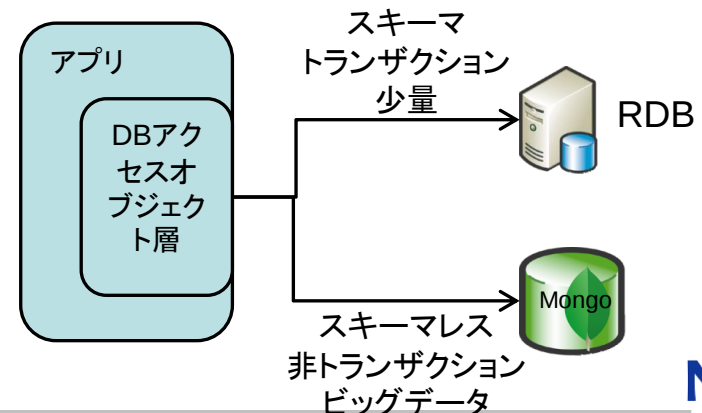
データウェアハウス フロントエンド



シングル オペレーション ビュー



DB負荷分割



- 社内プライベートクラウドでファイルシステムとRDBの中間として利用
- MongoDBは使いやすく運用がアプリケーションサイドで完結するため、少人数で運用可能
- 利用ユースケース1：社内ソーシャルサイト
 - ▶ 全てのデータはMongoDBで管理
 - ▶ リージョン間でのデータはReplicationで同期を取っている
 - ▶ 課題はトランザクション機能が無い事
 - ▶ ツイッターからのデータも引き出している
- 利用ユースケース2：クラウド上で社内ドキュメントを管理するシステム
 - ▶ ファイルの属性情報を記録するのにMongoDBを採用
 - ▶ これによってドキュメントの迅速な検索が可能
 - ▶ 課題はインデックス性能とトランザクション機能の不足

- 社内でMongoDBによる開発を積極採用
- なぜMongoDBなのか？
 - ▶ アジャイル開発に最も向いてる技術として評価した
 - ▶ 営業と開発が一緒になって設計活動に参画出来る
 - ✓ それはJSONが営業の人でも理解出来るデータ構造であるため
 - ▶ MongoDBは魅力的であるため、開発者を活気づける
 - 実はこれが最も重要
- 課題
 - ▶ MongoDBが魅力的で簡単にすぐ作れるため、研修などが置き去りになり、後でハマるパターンがある
 - 研修が重要

野村総合研究所の取り組み

- **野村総合研究所のOpenStandiaチームは、約50種類のオープンソースを取り扱っており、MongoDBは其中でも力を入れている物の一つです。**
 - ▶ nginx, Apache HTTP Server, Tomcat, JBoss, MySQL, Apache Solr, OpenAM, Liferay, Pentaho, Alfresco, Jaspersoft, etc...
- **野村総合研究所はMongoDB, Incと国内で初めてパートナー契約結び、正規販売代理店です。**
- **提供するメニューは以下の通り**
 - ▶ MongoDB Enterpriseサブスクリプションの販売
 - ▶ MongoDBの日本語サポート窓口
 - ▶ MongoDBおよびMMSの構築・コンサルティング
 - ▶ MongoDBおよびMMSのトレーニング

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。



お問い合わせは、NRIオープンソースソリューション推進室へ



osscc@nri.co.jp



<http://openstandia.jp/>