データベース部会　設立セミナー

NoSQLの必要性と主要プロダクト比較

2015/8/27
野村総合研究所
オープンソースソリューション推進室
主任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 オープンソースソリューション推進室
Mail：ossc@nri.co.jp　　Web：http://openstandia.jp/

オープンソースまるごと
OpenStandia™
Open Source Technology

NRI 未来創発
Dream up the future.

自己紹介

{"ID"　："fetaro"
"名前"　："渡部　徹太郎"
"所属"　：野村総合研究所
"研究"　："東京工業大学でデータベースと情報検索の研究
　　(@日本データベース学会)"
"仕事"　：{"昔":"証券会社のオントレシステムのWeb基盤",
　　"今":"オープンソースならなんでも"}
"エディタ"："emacs派"
"趣味"："自宅サーバ"
"属性"　：["ギーク","スーツ"]
}

アジェンダ

オープンソースまるごと
OpenStandia™
Open Source Technology

14:50～15:50

- NoSQLが必要とされる背景（15分）～15:05
- NoSQLの分類（10分）～15:15
- NoSQL主要プロダクトの紹介（25分）～15:40
- まとめ（5分）～15:45
- 質疑応答（5分）～15:50

NoSQLが必要とされる背景

背景　3つのVの増加

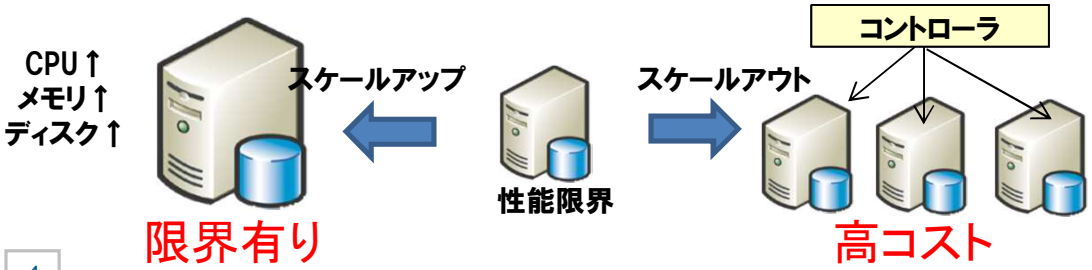
オープンソースまるごと
OpenStandia™
Open Source Technology

- **Volume（データ量）の増加**
  - Googleは１日に24ペタバイトのデータを処理している
    - （2008 MapReduce: simplified data processing on large clusters）
  - facebookの写真は1.5ペタバイト
    - （2009 Needle in a haystack: efficient storage of billions of photos）
- **Velocity（処理速度）の増加**
  - orange（大規模Web）サービス秒間11万アクセス
  - マウスの軌跡を解析→秒間万単位の書き込み、月間１１億PV
  - twitter「バルス」で秒間14万つぶやき

(出所)：日経SYSTEMS 2015年5月号

［課題］RDBMSでは扱いが困難

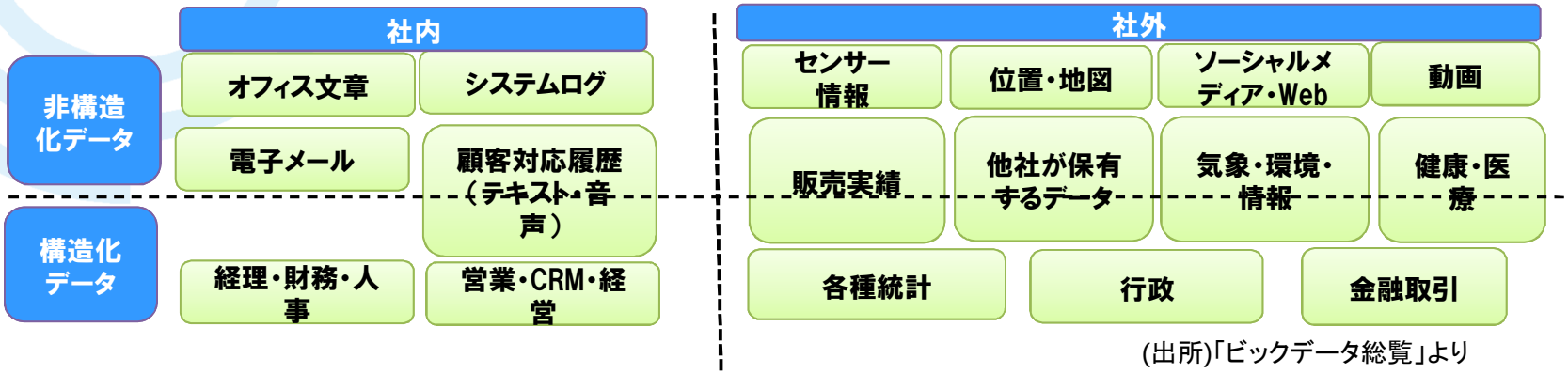
- RDBMSのスケールアップではハードウェアの限界
- RDBMSのスケールアウト（水平分散）は高コスト



背景　3つのVの増加

Variety（多様性）の増加

- 非構造データが増えてきている



(出所)「ビックデータ総覧」より

［課題］ RDBMSでは格納が困難

- 格納前に構造定義（Create Tabel）をする必要があり、工数大
- データソースの構造が更新されたら、スキーマ変更（Alter Table）が必要があり、工数大。
- そもそも事前に構造がわかっていない場合はどうしようもない

非構造データを扱うケース

オープンソースまるごと OpenStandia™ Open Source Technology

- 「Twitterのデータを分析して商品評判を知りたい。プロトを3日で作ってくれ」
- → TwitterのAPIを叩いてみると、こんなJSONが返ってきます


```
{
    "_id" : ObjectId("55b93f4bb427f0c12e080473"),
    "created_at" : "Wed Jul 29 20:57:42 +0000 2015",
    "id" : 626496848031690800,
    "id_str" : "626496848031690752",
    "text" : "@mchris4duke and certainly not concious, for a fleeting
              moment, of their prejudices",
    "source" : "<a href=¥"http://twitter.com¥" rel=¥"nofollow¥">Twitte
                Web Client</a>",
    "truncated" : false,
    "in_reply_to_status_id" : 626496667957751800,
    "in_reply_to_status_id_str" : "626496667957751808",
    "in_reply_to_user_id" : 14093339,
    "in_reply_to_user_id_str" : "14093339",
    "in_reply_to_screen_name" : "mchris4duke",
    "user" : {
        "id" : 772136,
        "id_str" : "772136",
        "name" : "Greg Smith",
...
```


これは何桁あるんだろう？

なんか、前と変わってるような


```
"geo" : null,
"coordinates" : null,
"place" : null,
"contributors" : null,"retweet_count" : 0,
"favorite_count" : 0,
"entities" : {
    "hashtags" : [ ],
    "trends" : [ ],
    "urls" : [ ],
    "user_mentions" : [
        {
            "screen_name" : "mchris4duke",
            "name" : "Chris Bourg",
            "id" : 14093339,
            "id_str" : "14093339",
            "indices" : [
                0,
                12
            ]
...
```


ここは一対多だなあ

ここも一対多だよなあ、テーブルは3つに分けて、JOINを2回するしかないか

- 皆さん、この状況でこのデータをRDBMSにいれますか？？
  - 全て型を(英語で)調べてCREATE TABLEしなくてはないけません
  - 一対多の部分はテーブル分割しないといけません
  - 一部だけ格納しますか？途中で追加が必要だと気づいた場合に手遅れです
  - 思い切って文字列で入れますか？中身の検索ができないです

背景　ビックデータ・非構造データ

- orange（Webサービス事業）　700万ユーザを超えるWebサービス
  - ▶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

- アットホーム株式会社（不動産）
  - ▶会員数の増加に伴うビックデータ化
  - ▶物件情報に付随する写真などの非構造化データの増加
  - ▶不動産公正取引協議会の規約改定やお客様の利便性向上のために物件情報データベースの項目を追加、変更する際にサービスを停止する必要があった

- OTTO（eコマース）　*毎日200万人が利用する*
  - ▶RDBMSベースの旧システムでは商品カタログのアップデートに12時間かかった

背景　ビックデータ・非構造データ

- A社（国内製造業）
  - ▶商用RDBMSではスケールアップが高価すぎる
  - ▶端末が出すデータが変わるたびに、スキーマ変更が発生し工数増大
- Metlife（保険業）
  - ▶70以上の既存RDBMSに拡散した顧客情報のデータ統合をしたいが、RDBMSではスキーマ定義・変更に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
- Citi（銀行）
  - ▶RDBは業務全体の99％を占めていながらも、最近急増しているのは非構造型のデータ
  - ▶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。

NoSQLの登場

オープンソースまるごと
OpenStandia™
Open Source Technology

- 「非構造データ」「ビックデータの」処理であれば、
  NoSQL（Not Only SQL）というデータベースが得意
- NoSQLは構造を定義することなくデータを格納できる

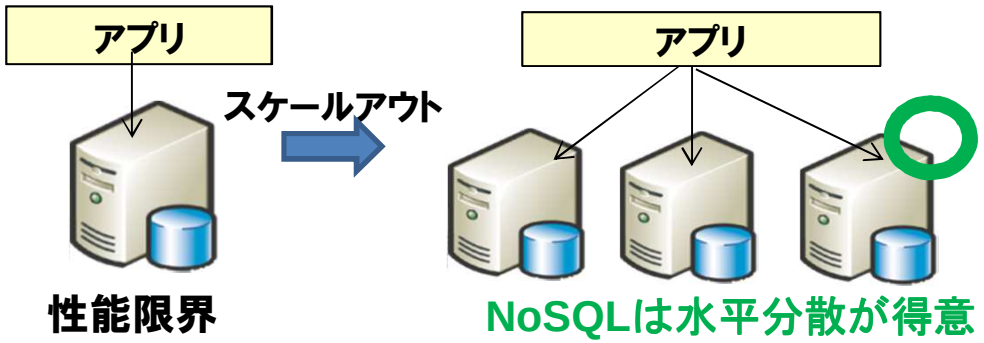
CREATE TABLE
ALTER TABLE
不要

	movie_id	value
1		"SF"
2	10	"キ[illegible]
3	10	[illegible]ウッド"
4	11	"ファンタジ[illegible]
5	11	"宮崎駿"
6	11	"ジブリ"

とりあえず溜めるて、出すときに考える ことができる

▶いざ何かしようとした時に、データがなければ何も始まりません。
とりあえず溜めることの重要性は「データレイク」という思想で広まりつつあります

- NoSQLは安価なハードウェアを並べて水平分散ができる（物が多い）

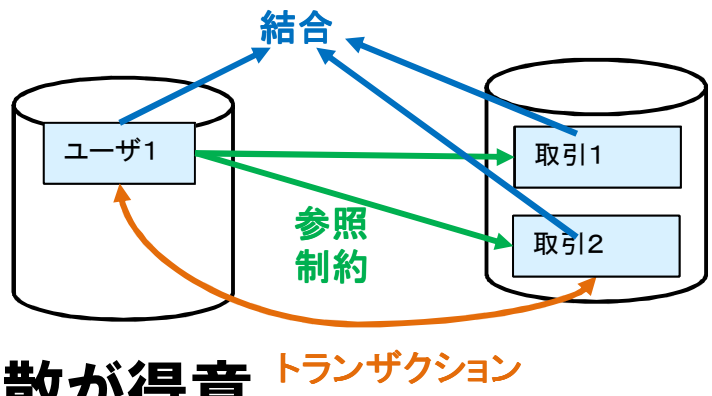


NRI 未来創発 Dream up the future.

NoSQLがなぜスケールアウトできるか

RDBMSは水平分散が苦手

- 水平分散する機能が**高価**であったり、**構築に手間がかかる**
- 水平分散しても、一貫性を保証するために、トランザクション、結合、参照制約といった機能を使う場合は**複数ノードをロックする必要がある** ***「強い整合性」***
  →システム全体のスループットが頭打ち

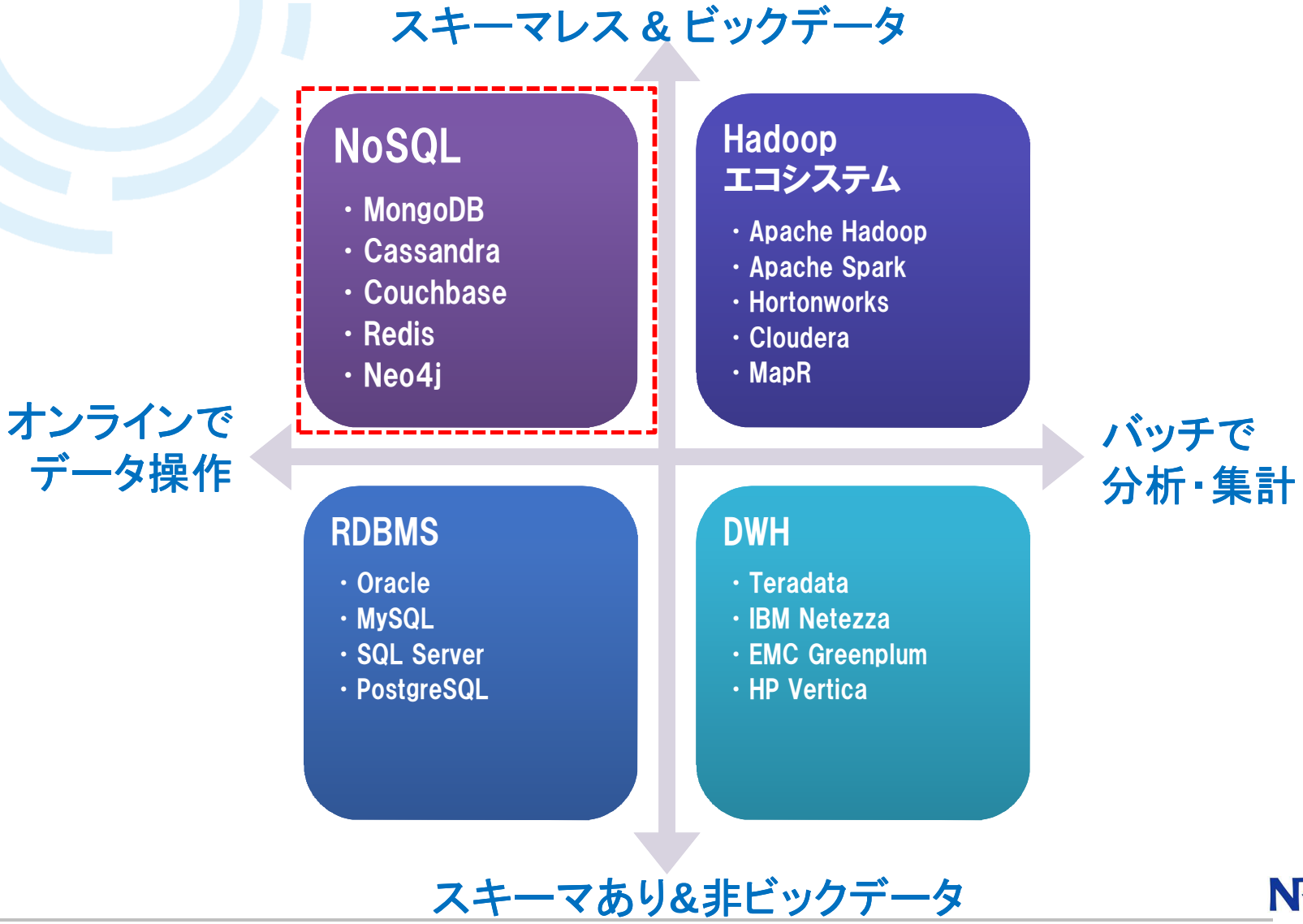


NoSQLが水平分散が得意

- もともと水平分散するように作られているため、安価で構築が容易
- 一貫性は厳密には保証せず、**最終的に整合があえば一時的に一貫性を損なっても良い**という考え方 ***「結果整合性」***
- トランザクション、結合、参照制約はほぼ提供されていない
  →ロックは最小限 →システム全体のスループットは線形に上昇

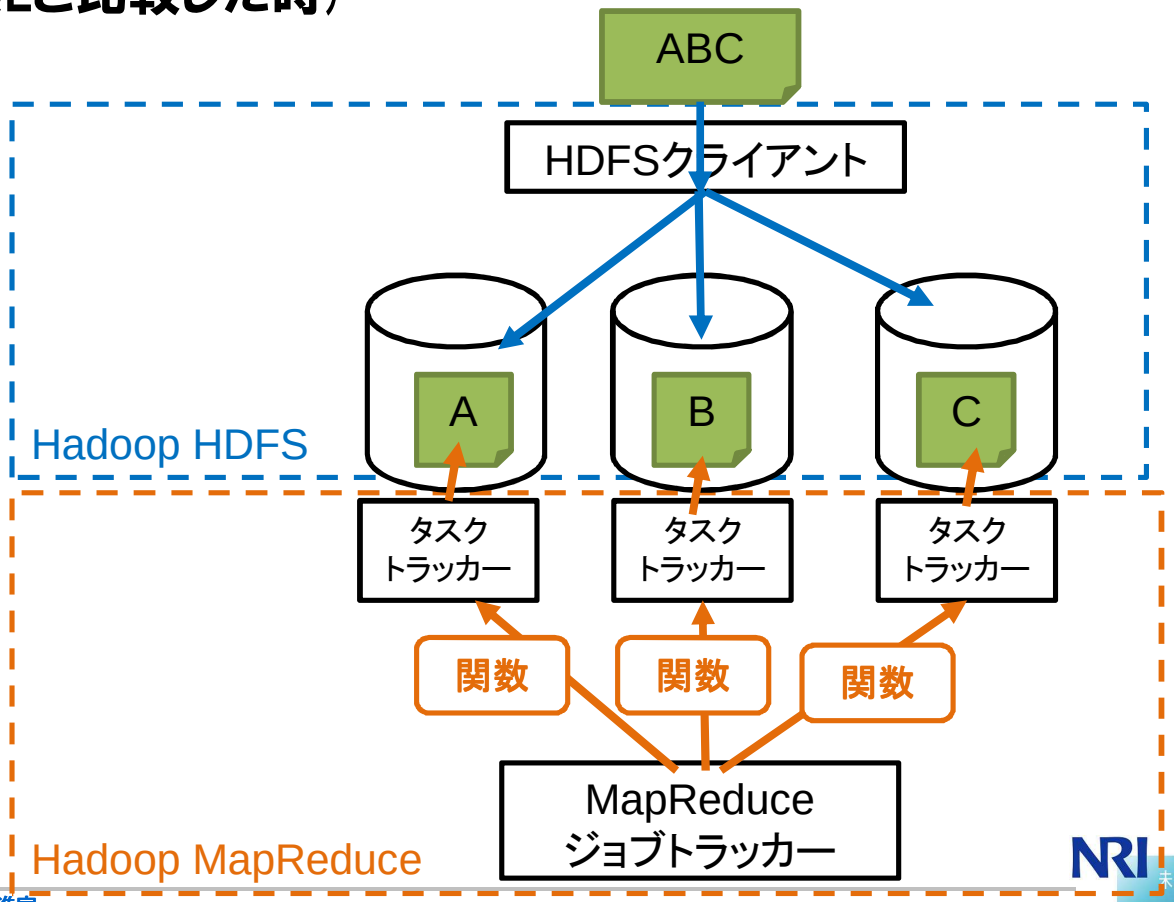
DBの中のNoSQLの位置づけ

- DBの中でのNoSQLの位置づけ



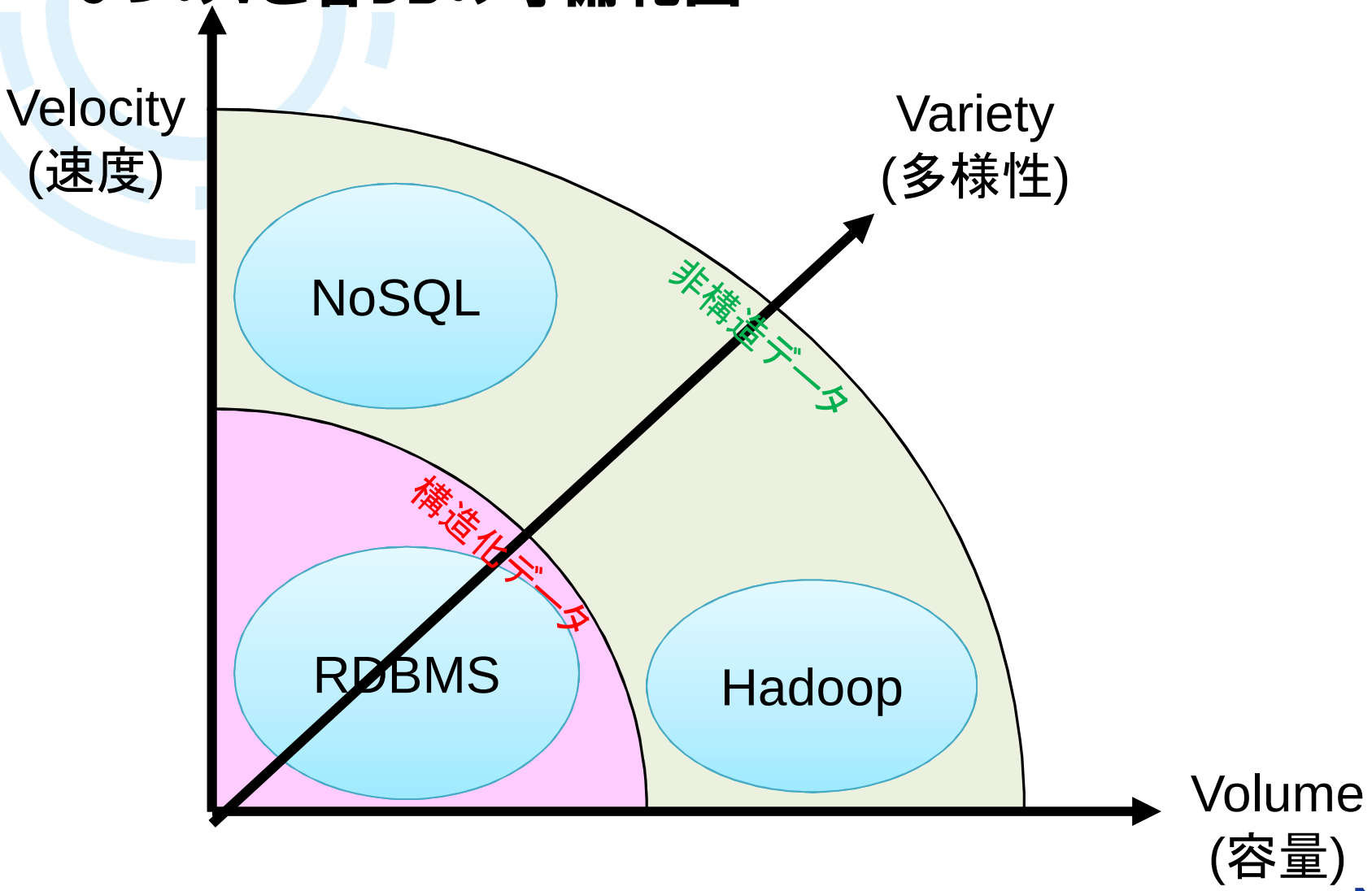
Hadoopとの違い

- Hadoopはバッチの分散処理
  - ペタバイト級のデータを数十台で並列分散するために作られたもの。
  - 事前にファイルを分散ファイルシステムHadoop HDFSに格納する
  - データに対して関数（map関数とreduce関数）を渡して、分散計算する
- Hadoopの特徴（NoSQLと比較した時）
  - 事前にデータを入れる
  - 応答速度よりも全体のデータ処理量を優先



DBの中のNoSQLの位置づけ

3つのVと各DBの守備範囲



NoSQLの注目度

オープンソースまるごと OpenStandia™ Open Source Technology

DB ENGINESによる注目度ランキング

以下の指標を総合的に判断
・ウェブ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など)

・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)

Rank			DBMS	Database Model
Jun 2015	May 2015	Jun 2014		
1.	1.	1.	Oracle	Relational DBMS
2.	2.	2.	MySQL	Relational DBMS
3.	3.	3.	Microsoft SQL Server	Relational DBMS
4.	↑ 5.	4.	PostgreSQL	Relational DBMS
5.	↓ 4.	5.	MongoDB ⊞	Document store
6.	6.	6.	DB2	Relational DBMS
7.	7.	7.	Microsoft Access	Relational DBMS
8.	8.	↑ 9.	Cassandra ⊞	Wide column store
9.	9.	↓ 8.	SQLite	Relational DBMS
10.	10.	↑ 12.	Redis	Key-value store
11.	11.	↓ 10.	SAP Adaptive Server	Relational DBMS
12.	12.	↓ 11.	Solr	Search engine
13.	13.	13.	Teradata	Relational DBMS
14.	14.	↑ 17.	Elasticsearch	Search engine
15.	15.	15.	HBase	Wide column store
16.	16.	↓ 14.	FileMaker	Relational DBMS
17.	17.	↑ 18.	Hive	Relational DBMS
18.	18.	↑ 20.	Splunk	Search engine
19.	19.	↓ 16.	Informix	Relational DBMS
20.	↑ 21.	↑ 23.	SAP HANA	Relational DBMS

（出所）DB ENGINES
http://db-engines.com/en/ranking

NoSQLの注目度

オープンソースまるごと
OpenStandia™
Open Source Technology

DB ENGINESによる注目度ランキング

以下の指標を総合的に判断
・ウェブ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など)

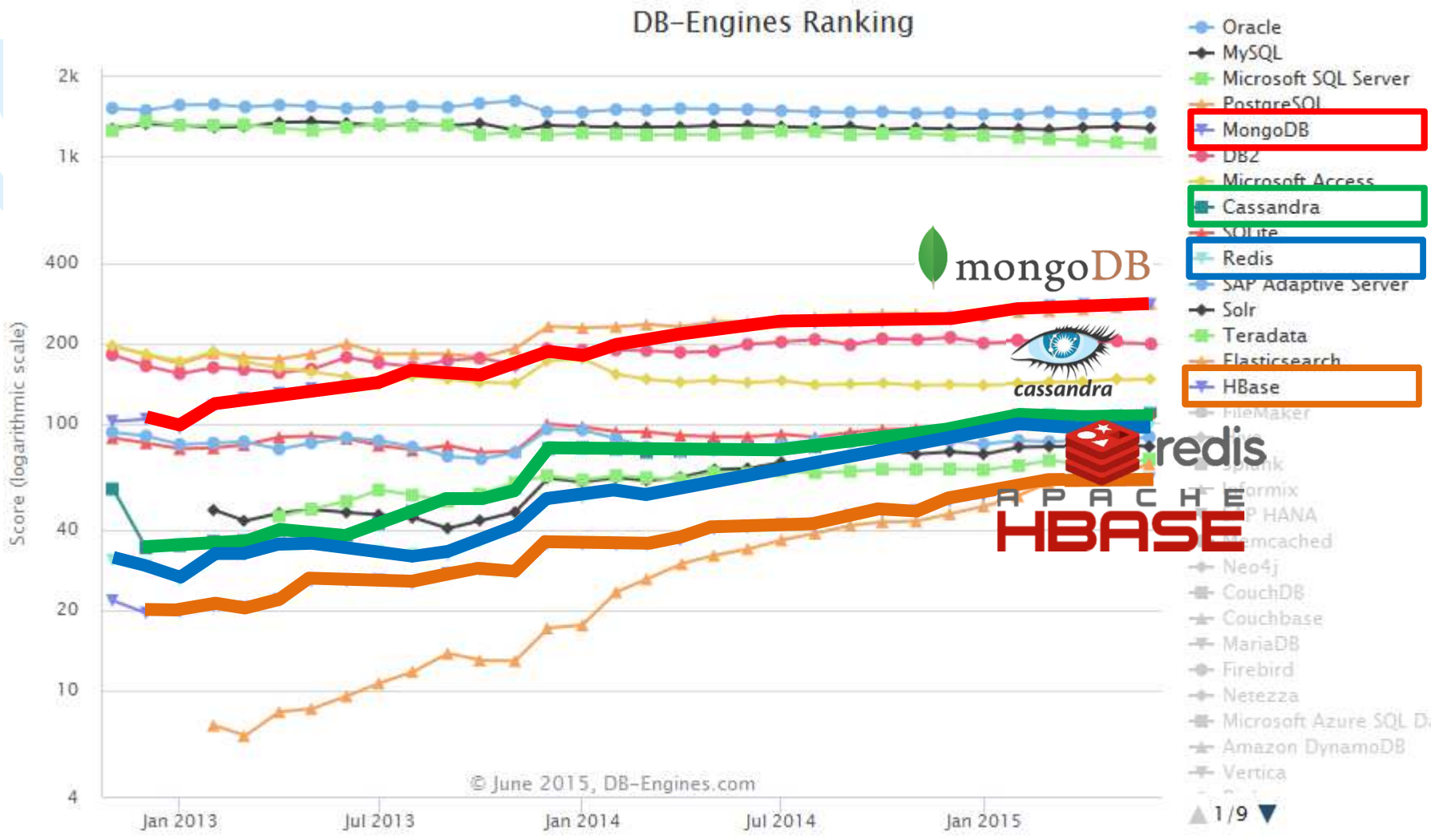
・求人サイトにおける募集スキル
(Indeed, Simply Hired)

・プロフィール登場回数
(LinkedIn)

(出所)DB ENGINES
http://db-engines.com/en/ranking

Rank Jun 2015	Rank May 2015	Rank Jun 2014	DBMS	Database Model
21.	↓ 20.	↓ 19.	Memcached	Key-value store
22.	22.	22.	Neo4j	Graph DBMS
23.	23.	↓ 21.	CouchDB	Document store
24.	24.	↑ 26.	Couchbase	Document store
25.	25.	↑ 28.	MariaDB	Relational DBMS
26.	26.	↓ 25.	Firebird	Relational DBMS
27.	27.	↓ 24.	Netezza	Relational DBMS
28.	28.	↓ 27.	Microsoft Azure SQL Database	Relational DBMS
29.	↑ 30.	↑ 32.	Amazon DynamoDB	Multi-model
30.	↓ 29.	↓ 29.	Vertica	Relational DBMS
31.	31.	↓ 30.	Riak	Key-value store
32.	↑ 34.	↑ 35.	MarkLogic	Multi-model
33.	↓ 32.	↓ 31.	dBASE	Relational DBMS
34.	↓ 33.	↑ 36.	Ingres	Relational DBMS
35.	35.	↓ 33.	Sphinx	Search engine
36.	↑ 37.	↓ 34.	Endeca	Search engine
37.	↓ 36.	37.	Greenplum	Relational DBMS
38.	38.	38.	Ehcache	Key-value store
39.	39.	39.	RavenDB	Document store
40.	40.	↑ 47.	Hazelcast	Key-value store

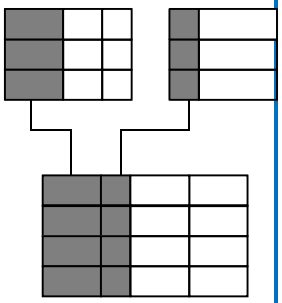
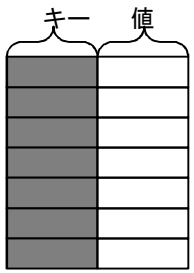
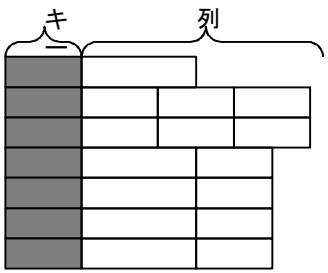
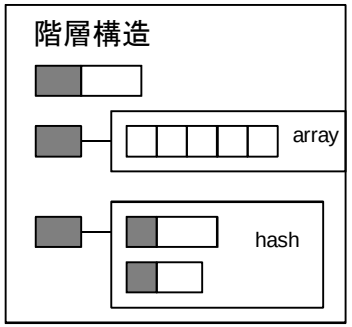
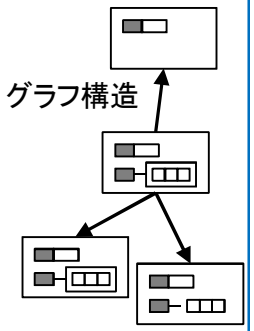
NoSQLの注目度


DB-Engines Ranking
Score (logarithmic scale)
2k
1k
400
200
100
40
20
10
4
Jan 2013
Jul 2013
Jan 2014
Jul 2014
Jan 2015
© June 2015, DB-Engines.com
mongoDB
cassandra
redis
APACHE HBASE
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
DB2
Microsoft Access
Cassandra
SQLite
Redis
SAP Adaptive Server
Solr
Teradata
Elasticsearch
HBase
FileMaker
Informix
SAP HANA
Memcached
Neo4j
CouchDB
Couchbase
MariaDB
Firebird
Netezza
Microsoft Azure SQL D
Amazon DynamoDB
Vertica
1/9

NoSQLの分類

NoSQLの種類

データ構造による分類

	RDBMS	NoSQL			
		KVS（キーバリューストア）		ドキュメント型	グラフ型
		キーバリュー型	列指向型 ワイドカラムストア		
データ構造 キー 値					

間違いやすい用語

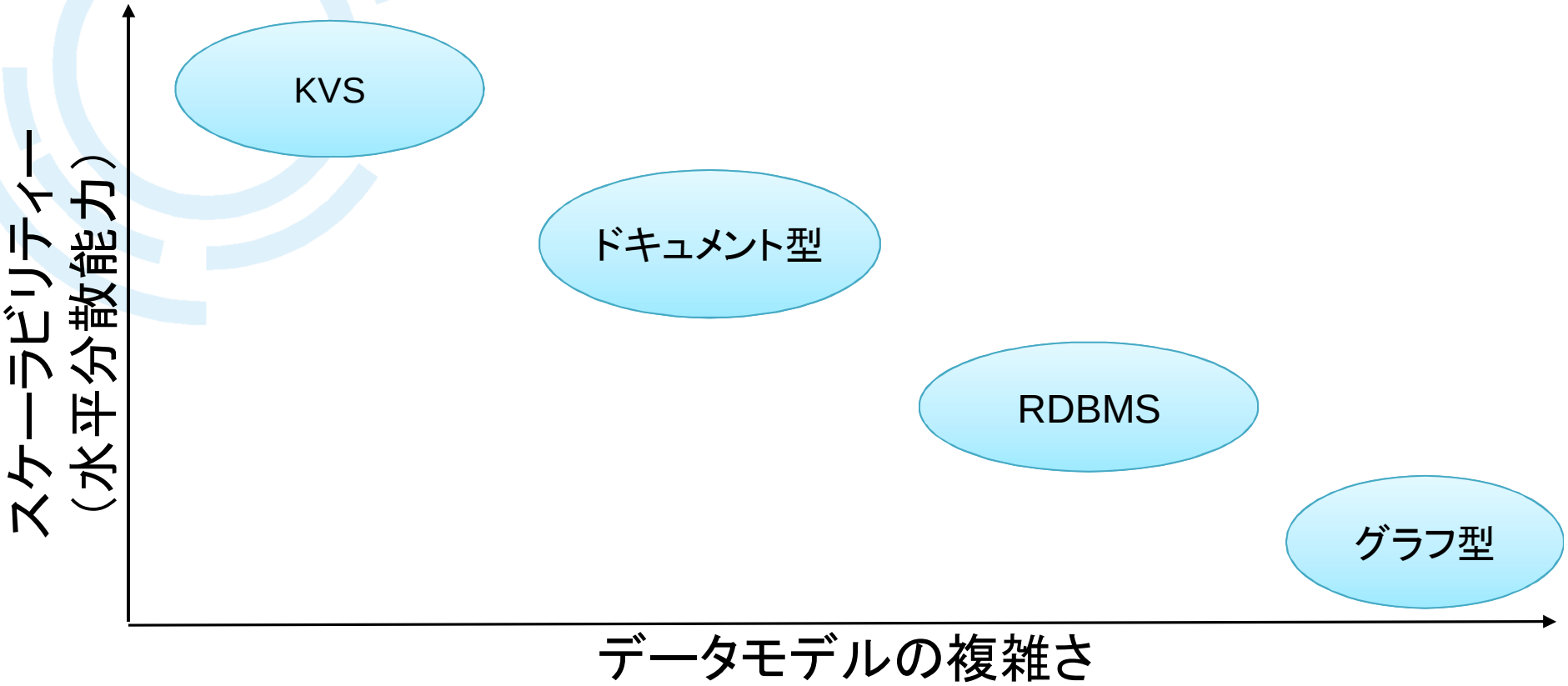
- ▶列指向RDBMS（カラムナー）
  - ✓列方向へのアクセスに特化した集計や分析に強いRDBMS
- ▶列指向型NoSQL（ワイドカラムストア）
  - ✓一つのキーに対して複数の値（列）をとるNoSQL

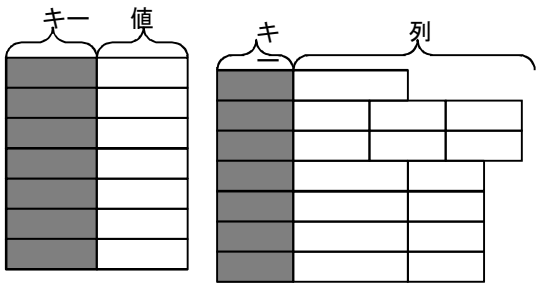
NoSQLの種類

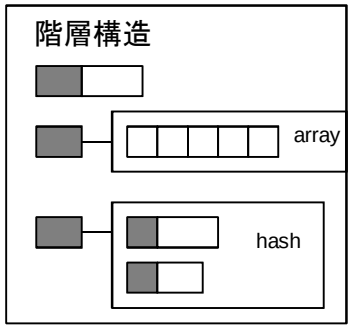
オープンソースまるごと OpenStandia™ Open Source Technology

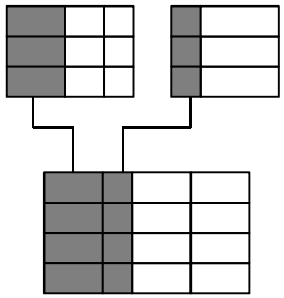
	NoSQL			
	KVS（キーバリューストア）		ドキュメント型	グラフ型
	キーバリュー型	列指向型		
OSS	redis memCached riak	cassandra APACHE HBASE	mongoDB Couchbase Apache CouchDB relax PostgreSQL MySQL	Neo4j
商用製品	ORACLE NOSQL DATABASE	AEROSPIKE	MarkLogic TERADATA IBM DB2	
サービス	Cloud Datastore AWS Elastic Cache Azure Redis Cache	amazon DynamoDB	Microsoft Azure DocumentDB IBM Cloudant®	
データ構造	キー 値 キー 値	キー 列	階層構造 array hash	グラフ構造

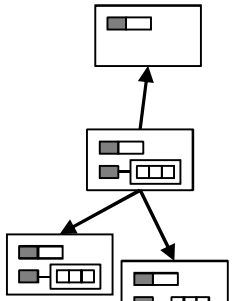
NoSQLの種類


KVS
ドキュメント型
RDBMS
グラフ型
スケーラビリティー
（水平分散能力）
データモデルの複雑さ

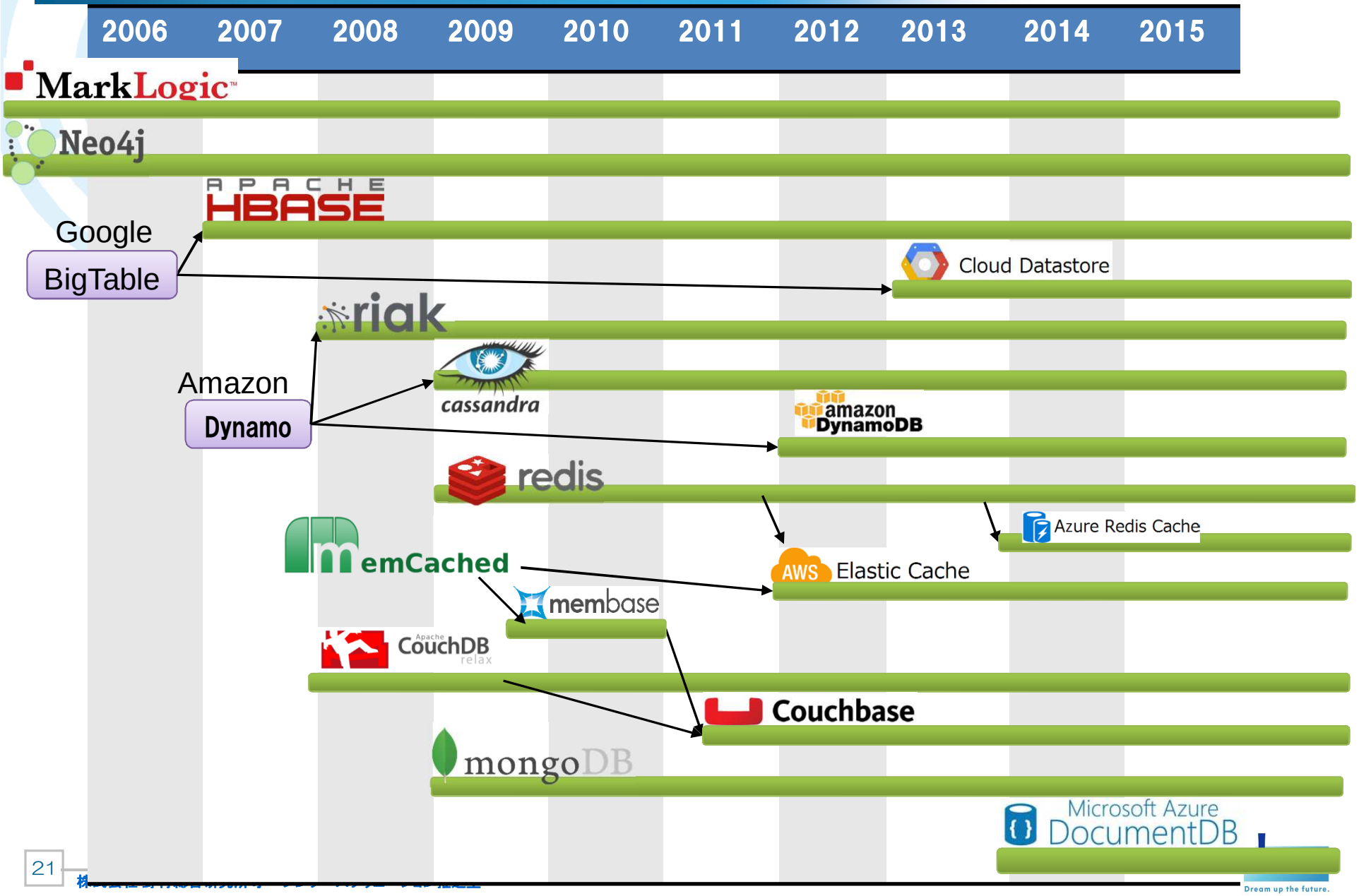

キー
値
キー
列


階層構造
array
hash



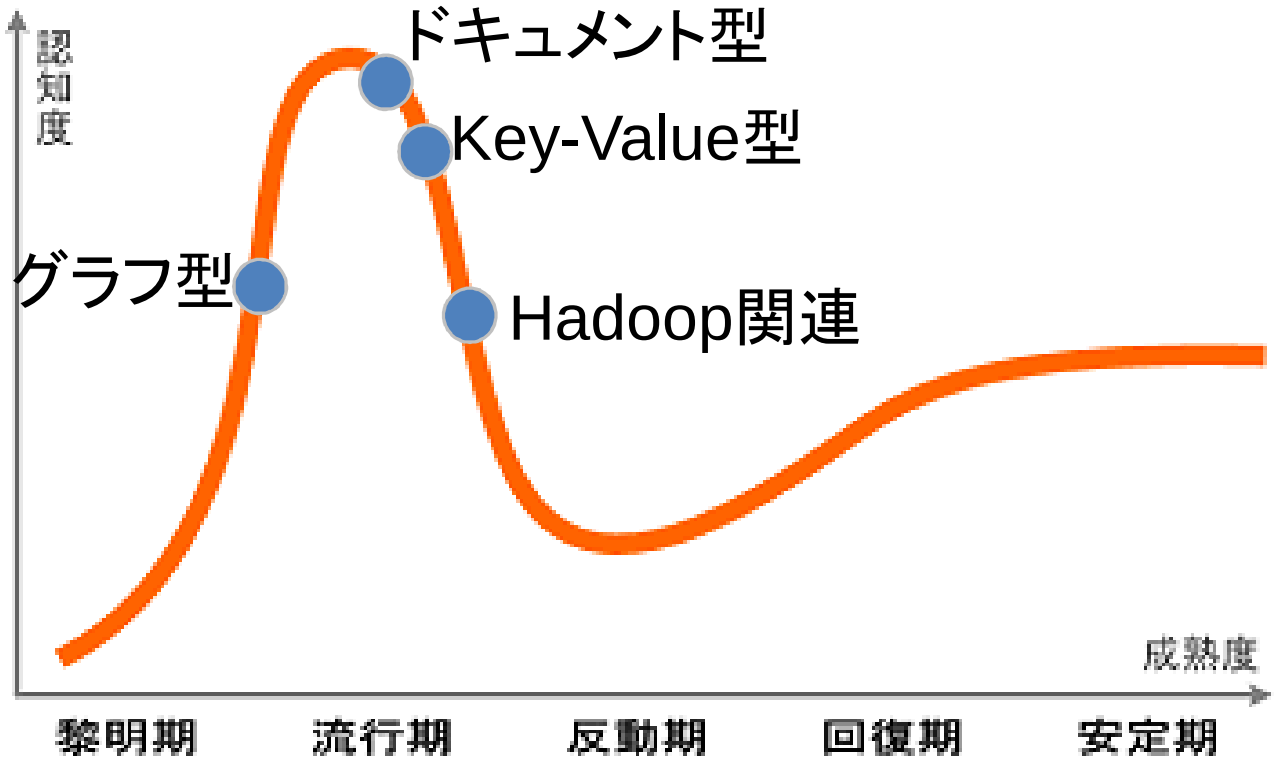


NoSQLの変遷


2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
MarkLogic
Neo4j
APACHE HBASE
Google
BigTable
Cloud Datastore
riak
cassandra
Amazon
Dynamo
amazon DynamoDB
redis
Azure Redis Cache
memCached
AWS Elastic Cache
membase
Apache CouchDB relax
Couchbase
mongoDB
Microsoft Azure DocumentDB

NoSQLの成熟度

- **ガートナーの調査**
  2014年度「ビックデータ」のハイプサイクル



典型的なハイプサイクルの例（出所：ガートナーの資料を参考に作成）

(出所) ガートナー ビッグ・データのハイプ・サイクル：**2014**年

主要プロダクト紹介

主要プロダクト紹介　キーバリュー型

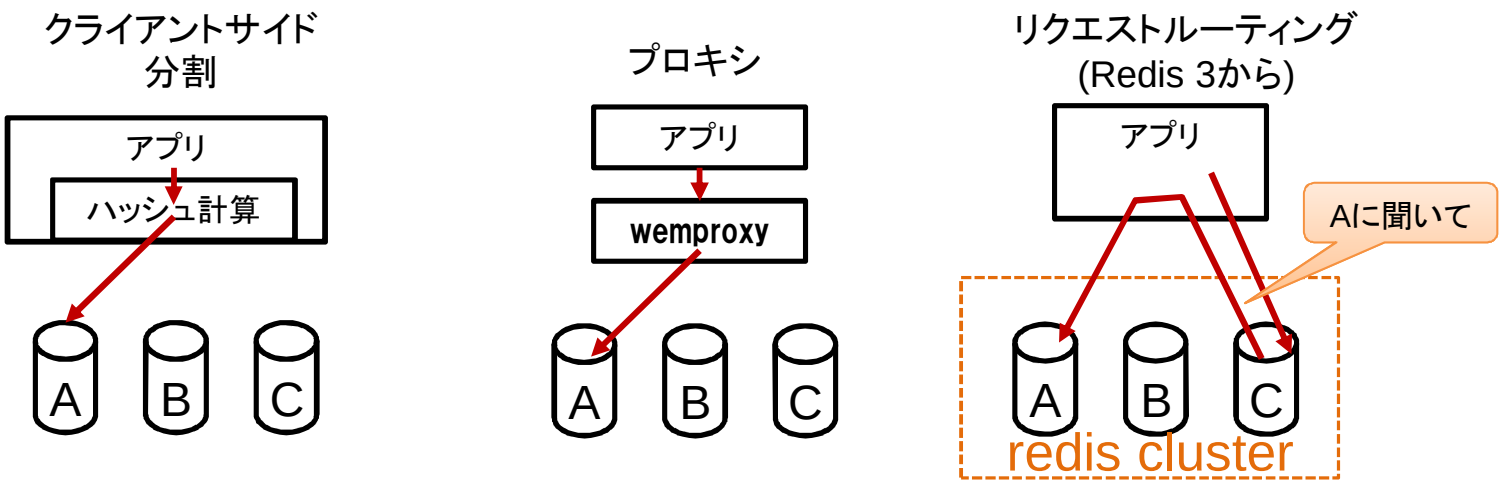
redis

Cloud Datastore

データ構造

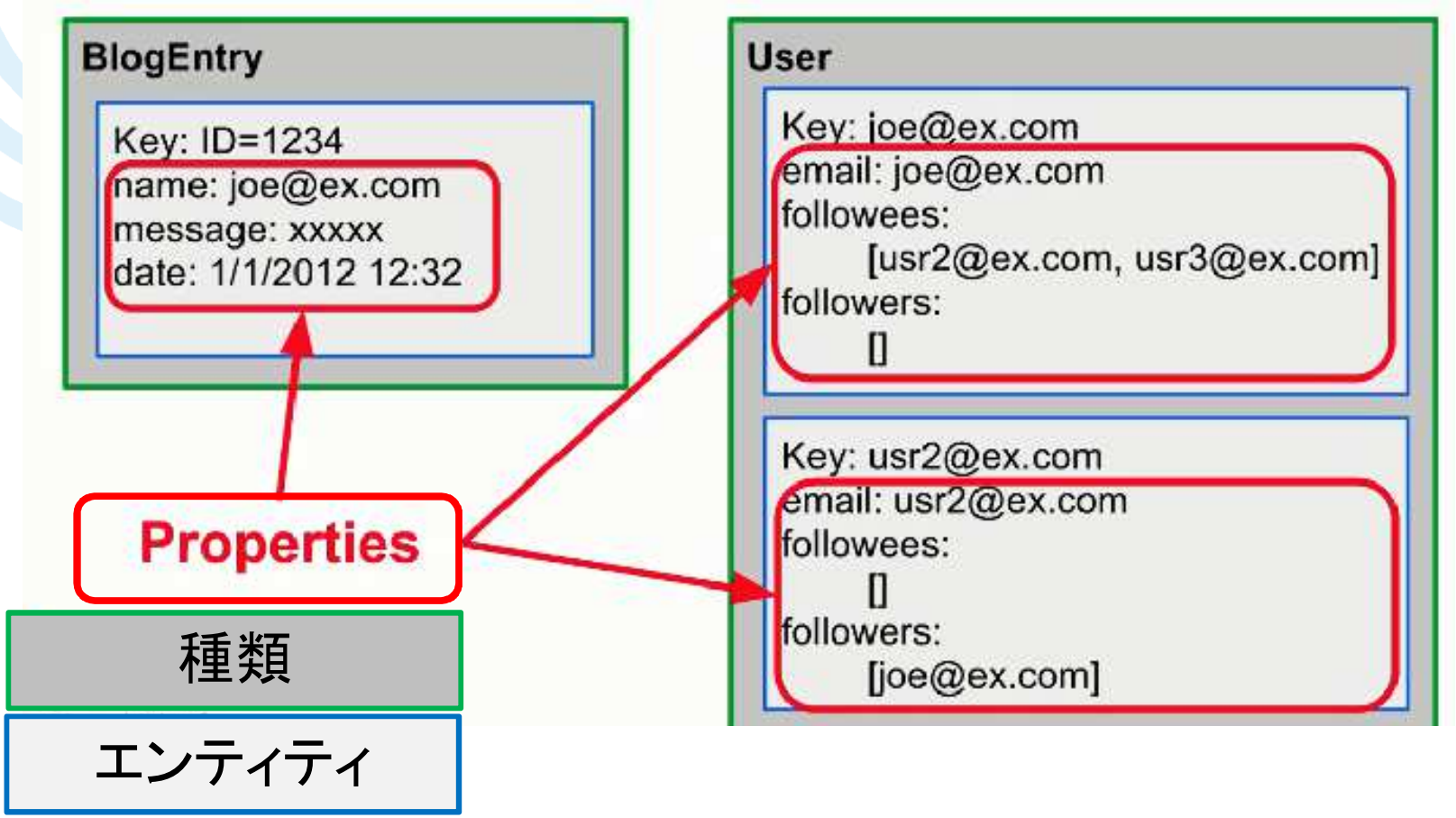
格納できるデータ	例	備考
キーバリュー	id : "watanabe"	
ハッシュ	watanabe : name : "watanabe" password : "hoge"	文字列のみ
配列	tweet : ["good morning","hello"]	40億まで格納できる
セット	friends : Kubota , Tamagawa , Fujisaki	重複の無い集合
ソート済みセット	friends : Fujisaki, Kubota, Tamagawa	

水平分散



特徴	内容
データ構造	データベース └データ 型：キーバリュー、配列、ハッシュ、セット、ソート済みセット
クエリ・インデックス	CRUD、 配列に対するPUSH,POP等 セットに対する、結合、和集合、積集合等
API	CLI, 各言語用ドライバ
水平分散	クライアントサイド分割、プロキシ、リクエストルーティングの3方式
レプリケーション	マスター　スレーブ
その他	巨大な配列（４億要素）の扱い 出版／購読（PUB/SUB） トランザクション
できないこと	メモリサイズ以上のデータ格納 ディスクへの先行書き込み 自動負荷分散

データ構造

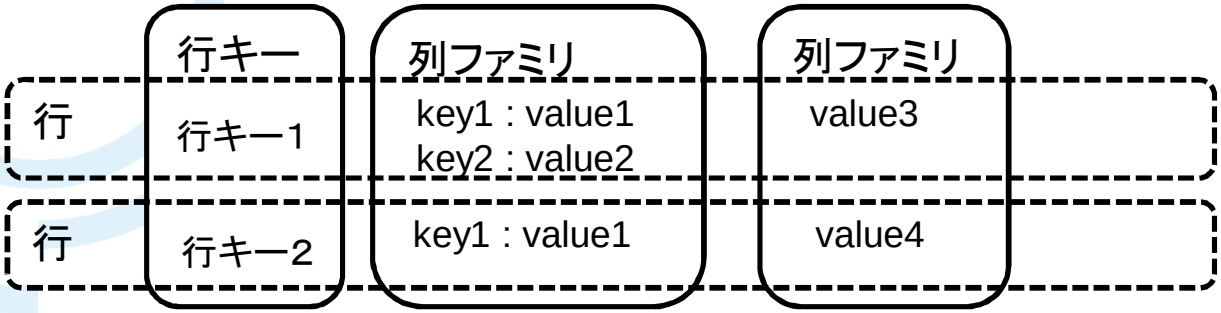

BlogEntry
Key: ID=1234
name: joe@ex.com
message: xxxxx
date: 1/1/2012 12:32
User
Key: joe@ex.com
email: joe@ex.com
followees:
[usr2@ex.com, usr3@ex.com]
followers:
[]
Key: usr2@ex.com
email: usr2@ex.com
followees:
[]
followers:
[joe@ex.com]
Properties
種類
エンティティ

Cloud Datastore

特徴	内容
データ構造	種類 （key　id = name） └エンティティ(ID, 祖先, 複数キー複数バリュー） └キーバリュー 型:キーバリュー、配列
クエリ・インデックス	エンティティの指定 キー、バリュー、祖先に対するフィルタ ソート
API	HTTP (JSON) 、ProtocolBuffer、SQLライクな言語 (GQL)
水平分散	不明(クラウドに隠蔽されている)
レプリケーション	
その他	部分的なトランザクションができる ・同じエンティティーグループ内 →強い整合性 ・異なるエンティティーグループ →5つのエンティティグループまで →更新は1秒に1件 →結果整合性
できないこと	Notはひとつの属性に限る 射影とソートの組み合わせはソートが先でなければならない

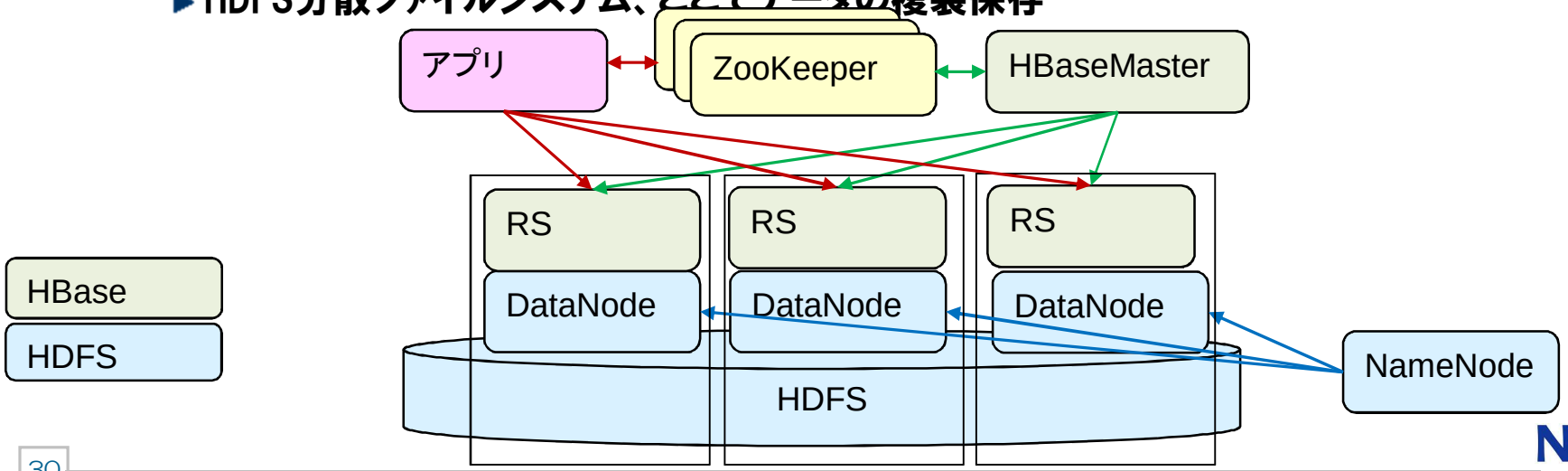
主要プロダクト紹介　列指向型

データ構造



水平分散

- HBaseMaster (HM)　リージョンファイルの配置管理
- RegionServer (RS) リージョンファイルの保持、読み込み書き込み
- ZooKeeper (ZK)　分散ロックサービス
- HDFS分散ファイルシステム、ここでデータの複製保存



特徴	内容
データ構造	データベース └キーバリュー 型：バイナリのみ
クエリ・インデックス	CURD インデックスは持たずに、行はキーの順に格納される Map Reduce
API	Shell, Java API, Thrift, HTTP (REST)
水平分散	マスタ型 自動負荷分散あり
レプリケーション	Hadoop HDFSにまかせる
その他	ユニーク制約、楽観ロック、カウンタ・連番 データ圧縮 強い整合性
できないこと	行キー以外のソート、インデックス スケールダウン 少数(５台未満)での利用 単独でデプロイできない、Hadoop HDFS, Zookeeprと一緒

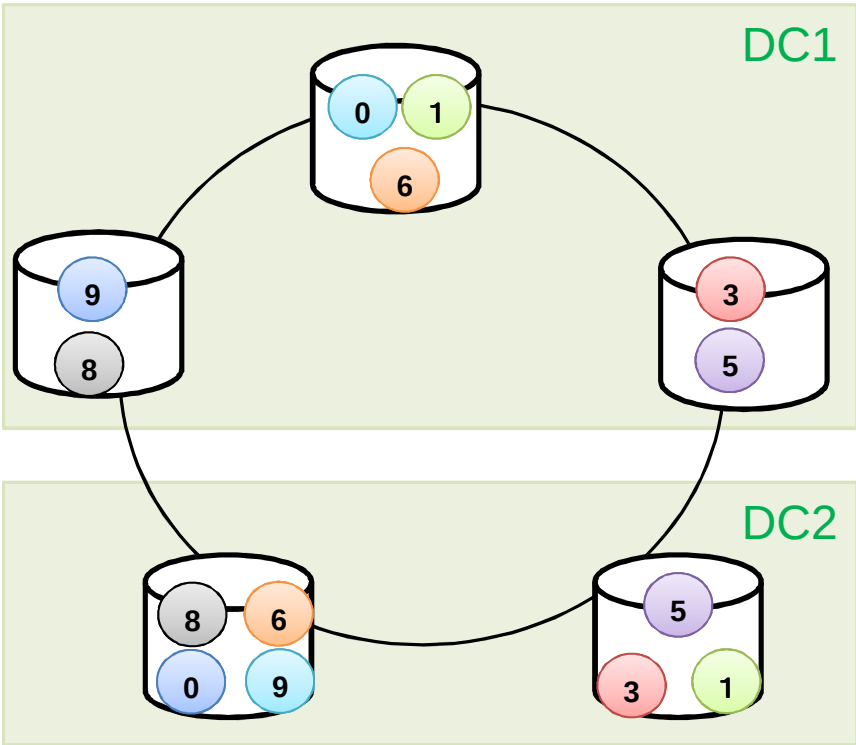
データ構造

```
cqlsh> use key_space_test ;
cqlsh:key_space_test> select * from testtable;

 pkey1   | pkey2   | skey | cvalue                | value
---------+---------+------+----------------------+-------------
 pkey1_1 | pkey2_1 |  103 | {'hoge1': 'fuga1'} | value11_103
 pkey1_1 | pkey2_1 |  102 | {'hoge1': 'fuga1'} | value11_102
 pkey1_1 | pkey2_1 |  101 | {'hoge1': 'fuga1'} | value11_101
```

水平分散

- クラスタへの書き込みはどこにでもできる
- 何台に書き込むか指定可能
- 何台から読むか指定可能
- レプリカは対等



特徴	内容
データ構造	キースペース └カラムファミリ └キー └(スーパーカラム) └カラム 型:文字列、数字、真偽値、小数、バイナリ、UUID、最小、最大
クエリ・インデックス	CURD 検索、ソート、Limit
API	CQL(Cassandra Query Language)
水平分散	ピアツーピア
レプリケーション	ピアツーピア
その他	トリガ プリペアードステートメント
できないこと	集計、JOIN、トランザクション 地理空間情報格納

https://cassandra.apache.org/doc/cql3/CQL.html#usingdates

主要プロダクト紹介　ドキュメント型

mongoDB

Couchbase

amazon DynamoDB

Microsoft Azure DocumentDB

MarkLogic™

ドキュメント型

- ドキュメント型データベースの特徴
  - 階層構造データであるドキュメント（JSON）を扱う
- JSONの特徴
  - JSONはXMLより**シンプル**
  - **可読性が高い**
  - 現在の**データ通信で主流のフォーマット**。
    - ✓ facebook twitterなどを始めとした多くのWebアプリケーションで採用

JSONの例

```
{
 ID        :  12345 ,
 name      : "渡部",
 address   : {
               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 }
             ]
```

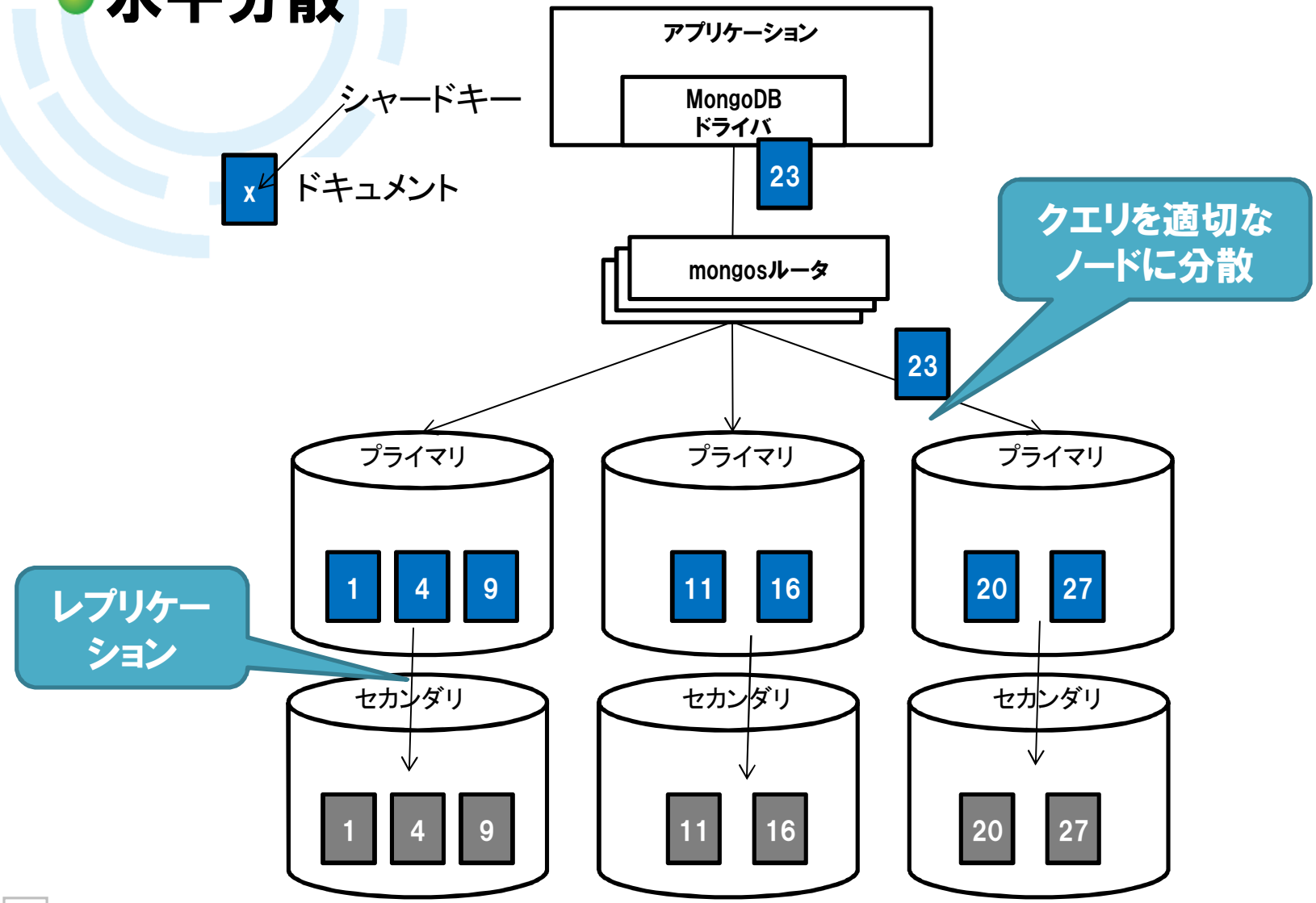
キーと値

サブドキュメント

配列

サブドキュメントの配列

水平分散

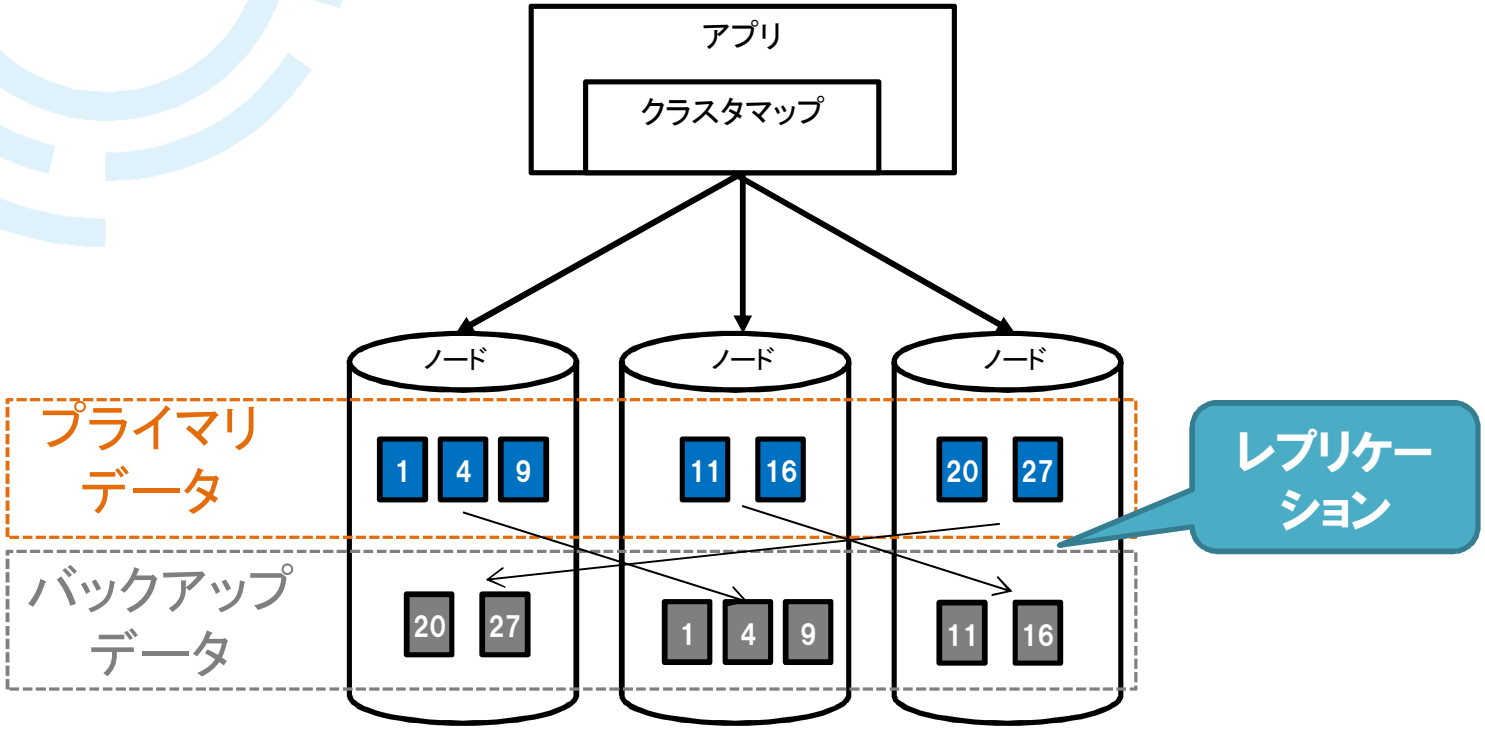

シャードキー
x
ドキュメント
アプリケーション
MongoDB
ドライバ
23
mongosルータ
23
クエリを適切な
ノードに分散
プライマリ
1
4
9
プライマリ
11
16
プライマリ
20
27
レプリケーション
セカンダリ
1
4
9
セカンダリ
11
16
セカンダリ
20
27

mongoDB

オープンソースまるごと
OpenStandia™
Open Source Technology

特徴	内容
データ構造	データベース └コレクション └JSON
クエリ・インデックス	CRUD、検索、ソート、Limit、正規表現（Mongoクエリ） SQL以上の集計クエリ（アグリゲーションフレームワーク）
API	各言語用ドライバ、簡単なREST
水平分散	マスタ型
レプリケーション	マスタ-スレーブ
その他	ロールベースのアクセス管理 大容量データの取り扱い 多様なインデックス（セカンダリ、ユニーク、マルチキー） 地理空間情報管理 データ圧縮（Ver 3から）
できないこと	マルチマスタ更新 トランザクション

水平分散


アプリ
クラスタマップ
ノード
ノード
ノード
プライマリ
データ
1
4
9
11
16
20
27
バックアップ
データ
20
27
1
4
9
11
16
レプリケーション

Couchbase

オープンソースまるごと OpenStandia™ Open Source Technology

特徴	内容
データ構造	データベース └バケット └JSON、バイナリ
クエリ・インデックス	CRUDとMapReduce ※次期バージョンではSQLライクな「N1QL」が用意される模様
API	HTTP (REST)
水平分散	マルチマスタ （クロスデータセンタにおける複数ノード更新）
レプリケーション	マスタ-スレーブ クロスデータセンタレプリケーション
その他	モバイルに組み込んで、サーバとデータ同期できる GUIコンソールの提供（データ操作、リソース監視） 多様なインデックス（セカンダリ、ユニーク、マルチキー） 地理空間データ管理
できないこと	アドホッククエリ

特徴	内容
データ構造	テーブル └アイテム 型：文字列、数値、バイナリ、ブール値、ヌル、文字列セット、数値セット、バイナリセット、リスト、マップ ※JSONをは文字列として格納されるが、ドライバで吸収することにより、ドキュメント型のような動きになる
クエリ・インデックス	CRUD、検索
API	各言語ドライバ、HTTP (REST)
水平分散	不明（クラウドに吸収されている）
レプリケーション	
その他	AWSの他サービスとの連携 GUIコンソールの提供（データ操作、監視、アラート） セカンダリインデックス
できないこと	集計、ソート、JSONの中の検索 同期的インデックス付与（非同期で6時間毎）？ 無制限セカンダリインデックス（5，5？

Microsoft Azure DocumentDB

オープンソースまるごと OpenStandia™ Open Source Technology

特徴	内容
データ構造	テーブル └コレクション 　└JSON、(添付ファイル)
クエリ・インデックス	SQLライクなクエリ 自動インデックス作成ができる 　同期(一貫した)と非同期(遅延)が選択可能 　ドキュメント内の特定のパスをインデックスから除外可能
API	GUI, HTTP(REST), Javascript
水平分散	ハッシュ、レンジ、クエリベースで分散可能
レプリケーション	不明
その他	トランザクションサポート 型チェック ストアドプロシジャ、トリガ、UDF(JavaScriptのアプリロジック) クエリの一貫性の調整
できないこと	集計機能がない。ソートができない(Preview状態) 複合インデックスがない レンジインデックスは数値のみに限る

MarkLogic™

オープンソースまるごと OpenStandia™ Open Source Technology

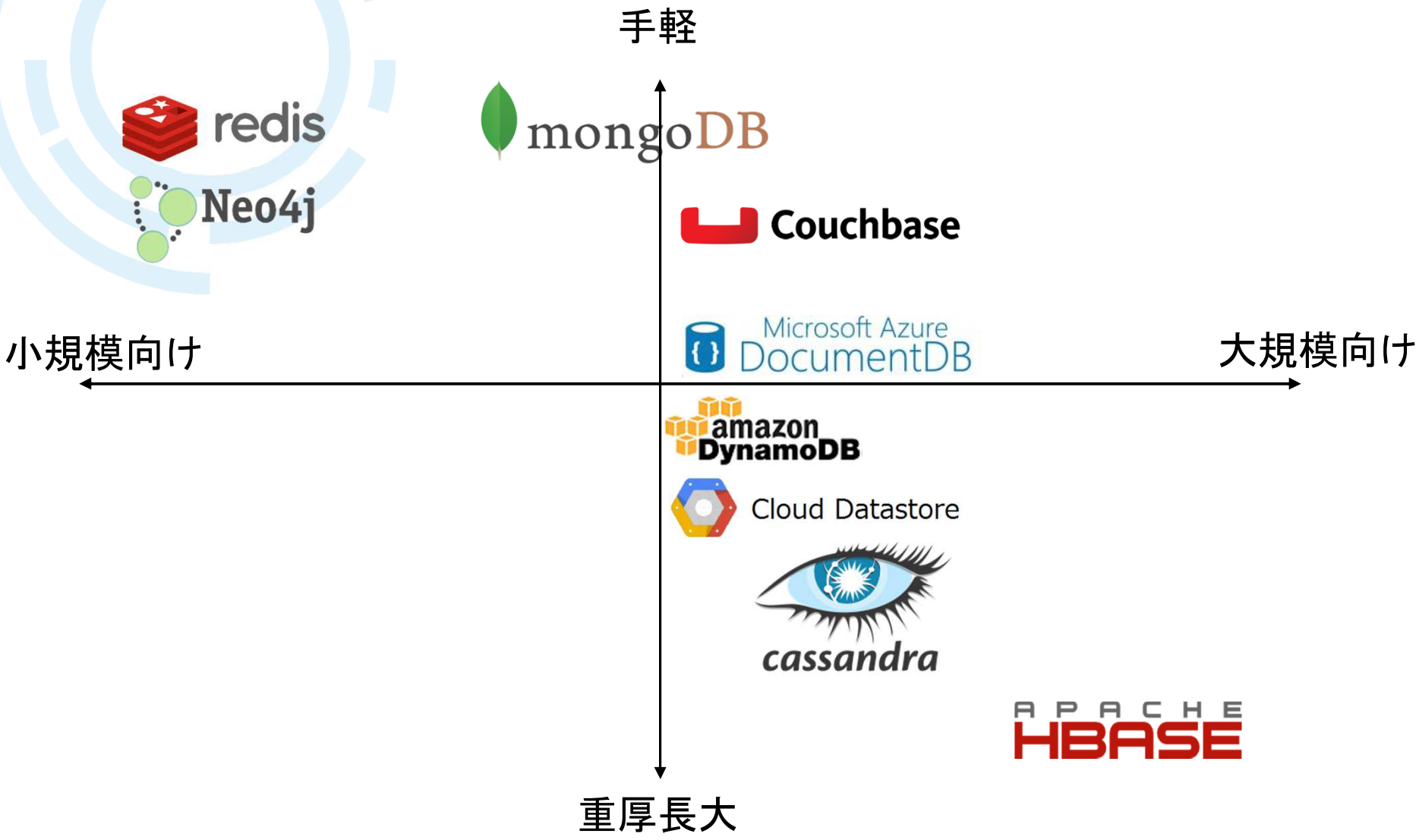
特徴	内容
データ構造	JSON, XML, RDFトリプル, バイナリ, リレーショナルデータ
クエリ・インデックス	JavaScript、XQuery、SPARQL 分類表示（ファセット化）、ドリルダウン
API	JavaScript, XQuery, SQL,
水平分散	シェアードナッシング
レプリケーション	マスター スレーブ
その他	トランザクション アクセス管理、監査、アラート 特徴語抽出、分類器 Hadoop連携
できないこと	？

主要プロダクト紹介　グラフ型

特徴	内容
データ構造	ノード → 関連 → ノード └JSON └JSON └JSON
クエリ・インデックス	Cypher（グラフ構造に特化したクエリ言語） 例）3ホップ先にAというデータが有る元のノードをリストせよ 最短のパスを探索せよ リングを検出せよ
API	Web, HTTP (REST)
水平分散	できない
レプリケーション	マスター-スレーブ
その他	GUIのインターフェースでクエリのチューニングが可能 GUIでグラフデータのアドホックな操作が可能
できないこと	水平分散 コミュニティー版はGPLライセンス

まとめ

今回の調査で感じた印象



NoSQLを見極めるポイント

他のNoSQLと何が違うのかを見極める

- ありがちな謳い文句
  - ✓「ビックデータ、IoTの処理に最適」
  - ✓「安価なハードウェアで利用可能」
  - ✓「無限にスケールする」
  - ✓「柔軟にデータを扱える」
  - ✓「事前にスキーマを定義する必要がなく、高速に開発可能」
  - ✓「メモリで高速に応答できる」
  - ✓「簡単にレプリケーションでき、大事なデータを保護」
- 差別化される要素
  - ✓データ構造
  - ✓アプリケーションからのインターフェース
  - ✓トランザクション
  - ✓セカンダリインデックス、複合インデックス
  - ✓クエリ(ソート、Limit、データの部分更新、集計)
  - ✓マルチマスタレプリケーション
  - ✓アクセス権限管理
  - ✓学習コスト(ドキュメントの量、ノウハウの多さ)

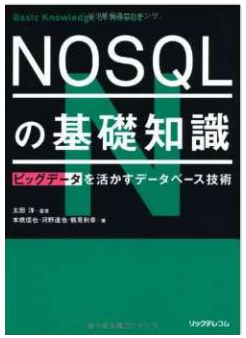
NoSQLを見極めるポイント

- 性能は単純比較できない
  - 性能を決める因子は様々
    - ✓データ量、クエリ、インデックス、メモリの使い方、ロックの粒度、水平分散の方式、結果整合性or強い整合性、etc
  - RDBMSのようにACID保証+SQLというルールの基で比較すれば意味はあるが、**そもそも一貫性やインターフェースが統一されていない**
  - **「NoSQL性能比較レポート」はあまりあてにならない**。人によって得手不得手がある。すべてを完璧にチューニングできる人は少ない。
  - もちろん、**クエリによってはMySQLのほうが速いこともある。**
  - **特性に注目すべき**
    - ✓例えばCouchbaseはマルチマスタで書き込めるから、シングルマスタのNoSQLよりも書き込みが速い
- 最新の公式ドキュメントを見る
  - ✓書籍等では情報が古い事が多く、**常に最新の公式ドキュメントを見ることが重要**
  - ✓半年前にはできなかったことが、できるようになっている事がある。

ご参考

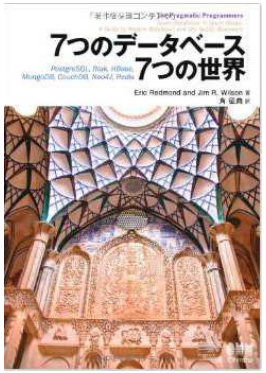
参考にした書籍

- NOSQLの基礎知識
  2014/3/6　本橋信也、河野達也



- NoSQLプログラミング実践活用技法
  2013/6/20　shashank Tiwari、長尾高弘

- 7つのデータベース 7つの世界
  2013/2/26　Eric Redmond、Jim R. Wilson



3000件を超える導入実績

オープンソース・ワンストップサービス

OpenStandia™

おかげさまで 契約件数 3000件 突破 OpenStandia™

OpenStandia（オープンスタンディア）の特徴

- 約50種類のオープンソースを、ワンストップでサポートします。
- 過去バージョンもサポートします。現在お使いのオープンソースをそのままサポートします。
- 大手企業エンタープライズシステムへの導入実績が豊富です。

- **サポート中**

mongoDB redis

- **サポート準備中**

Neo4j

本日の資料は、OpenStandiaのホームページに掲載しています。

宣伝

エンタープライズ/ジンさんで連載やってます

EL エンタープライズ/ジン データテクノロジー/企業セキュリティ ITリーダー向け専門メディア

Developers Summit 2015 ～関西ITエンジニアの

ホーム ニュース 記事 Security Online DB Online 主催イベント ホワイトペーパー

テーマ別に探す マネジメント クラウド モバイル ビッグデータ インフラ ソフト開発

「脱初心者！MongoDB中級編」連載一覧

2件中1～2件を表示

2015/08/03

第2回 MongoDBの単体性能（後編）

第1回ではMongoDBの単体性能の重要性、そして「何が遅いのか」の調べ方について説明してきました。第二回では「なぜ遅いのか」について考えてきましょう。

DB NoSQL/Hadoop MongoDB

2015/07/22

第1回 MongoDBの単体性能（前編）

「初心者から中級者になるためのMongoDB講座」は、MongoDBをインストールして実際にアプリケーションを作っている方、MongoDBのレプリケーションを運用しているが仕組みはあまりわからない方、シャーディングに手を出そうとしているが不安な方、そんなMongoDB初心者のみなさんへ向けた連載で

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推進室へ

ossc@nri.co.jp

http://openstandia.jp/

NRI 未来創発

Dream up the future.