TOSHIBA
Leading Innovation

ビッグデータ×IoT時代のデータベースのアーキテクチャとメカニズムの比較

２０１７年９月１８日

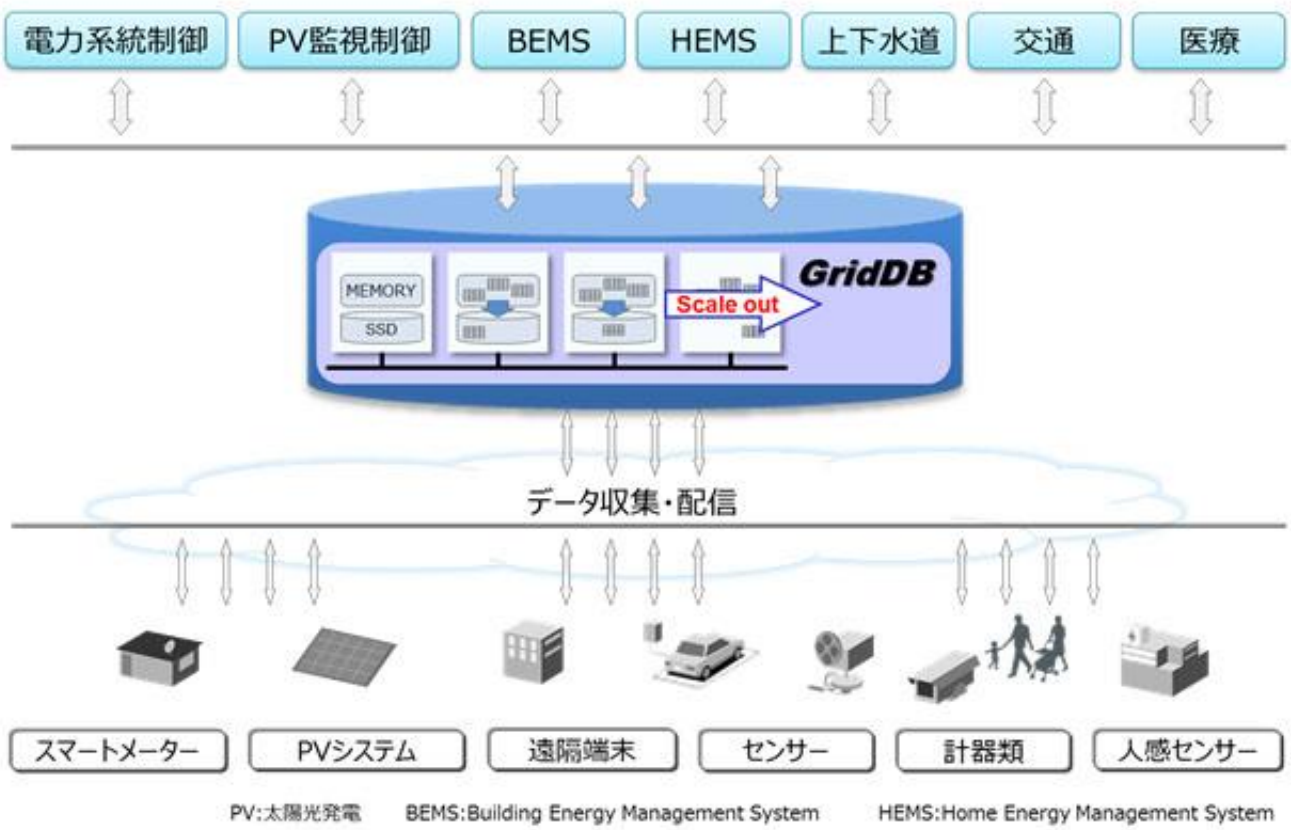
東芝デジタルソリューションズ株式会社 ＆ 株式会社東芝
野々村 克彦

発表内容

1. スケールアウト型データベースGridDB
2. アーキテクチャとメカニズムについて代表的なNoSQL DB（Cassandra、MongoDB）との比較
   - データモデル、クラスタ管理、一貫性と可用性について
   - クラスタ管理のメカニズムについて
3. まとめ

GridDBとは

- ビッグデータ／IoT向けのスケールアウト型データベース
- 開発(２０１０年)、製品化(２０１３年)、オープンソース化(２０１６年)
- 社会インフラを中心に、高い信頼性・可用性が求められるシステムで使われている



GridDB ４つの特長

IoT指向のデータモデル

- データ集計やサンプリング、期限解放、データ圧縮など、時系列データを効率よく処理・管理するための機能を用意
- データモデルはユニークなキーコンテナ型。コンテナ内でのデーター貫性を保証

高性能
High Performance

- メモリを主、ストレージを従としたハイブリッド型インメモリーDB
- メモリやディスクの排他処理や同期待ちを極力排除したオーバヘッドの少ないデータ処理により高性能を実現

スケーラビリティ
High Scalability

- データの少ない初期は少ないサーバで初期投資を抑え、データが増えるにしたがってサーバを増やし性能・容量を高めるスケールアウト型アーキテクチャ
- コンテナによりサーバ間通信を少なくし、高いスケーラビリティを実現

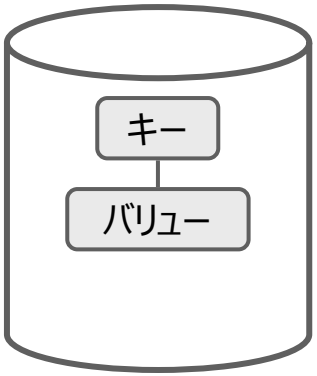
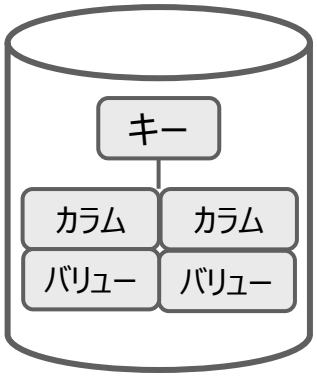
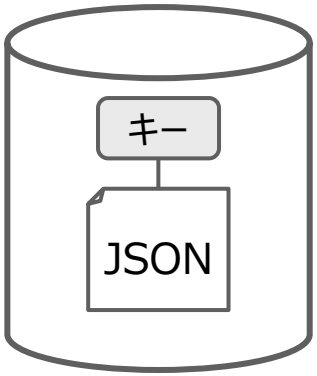
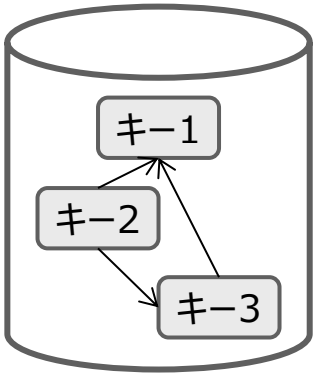
高い信頼性と可用性
High Availability

- データ複製をサーバ間で自動的に実行し、サーバに障害が発生しても、システムを止めることなく運用を継続することが可能

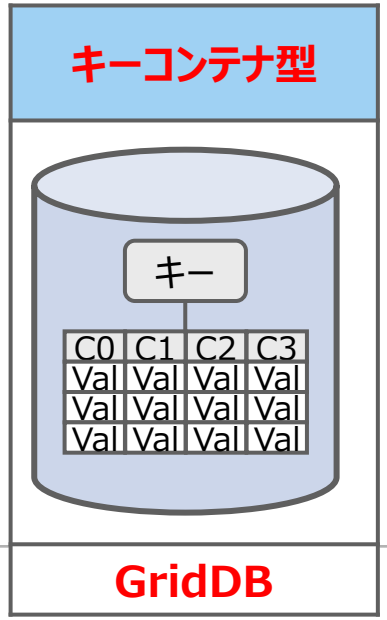
発表内容

1. スケールアウト型データベースGridDB
2. アーキテクチャとメカニズムについて代表的なNoSQL DB（Cassandra、MongoDB）との比較
   - データモデル、クラスタ管理、一貫性と可用性について
   - クラスタ管理のメカニズムについて
3. まとめ

①データモデル

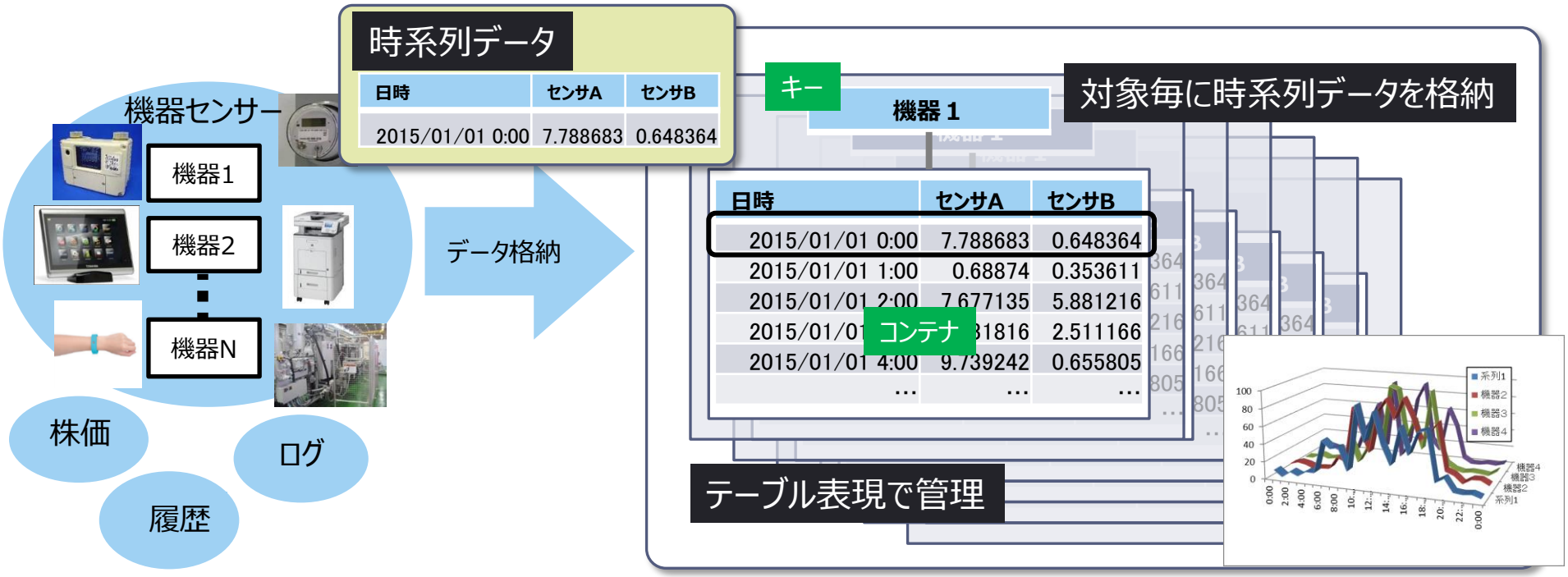
データモデル	キーバリュー型	カラム型	ドキュメント型	グラフ型
				
NoSQLの例	Riak、Redis	Cassandra	MongoDB	Neo4j

GridDBはキーコンテナ型

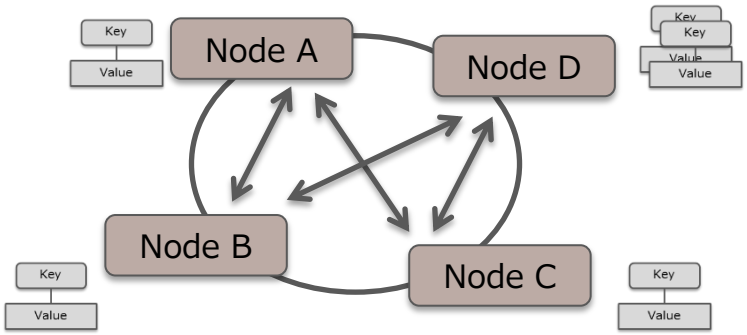
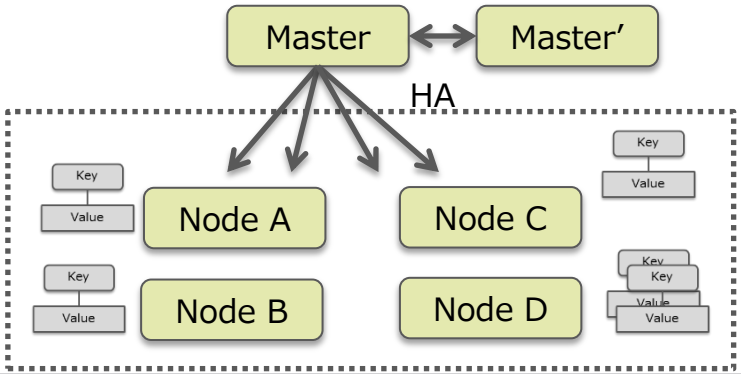


キーコンテナ型のデータモデル

- データをグループ化するコンテナ（テーブル）
  - ✓ コレクションコンテナ：レコードデータ管理用
  - ✓ 時系列コンテナ：時系列データ管理用。サンプリング、時系列圧縮、期限解放など時系列特有の機能がある
- コンテナ単位でACID保証



②クラスタ管理

P2P(Peer to Peer)方式	マスタスレーブ(Master Slave)方式
	
○ノード追加でのデータ再配置が容易 ×一貫性維持のためのノード間通信のオーバヘッドが大⇒一貫性と処理速度がトレードオフ	○一貫性の維持は容易 ×マスタノードが単一障害点(SPOF) ×ノード追加でのデータ再配置が難しい

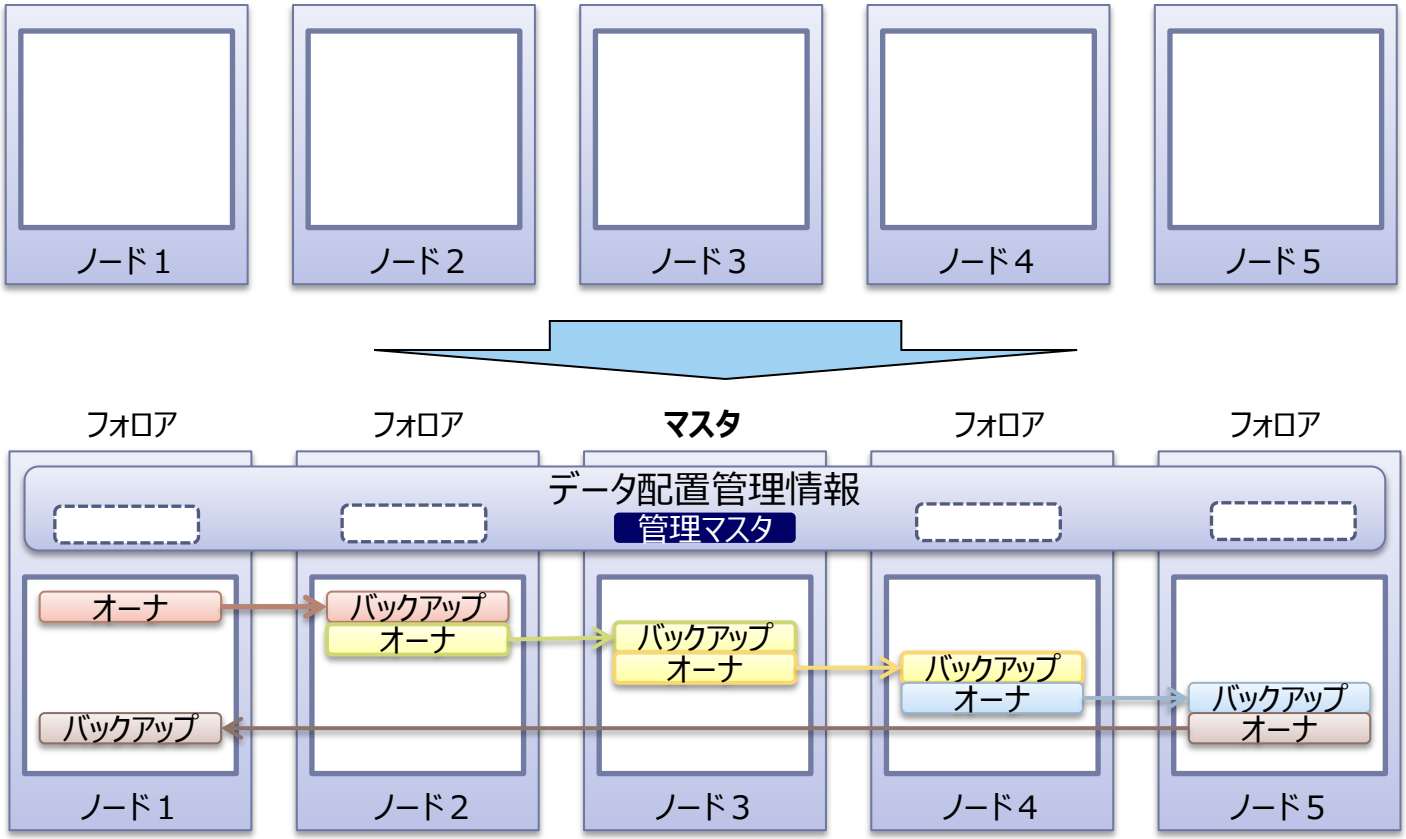
例：**Cassandra**

例：**MongoDB**

②クラスタ管理

GridDBはハイブリッド型

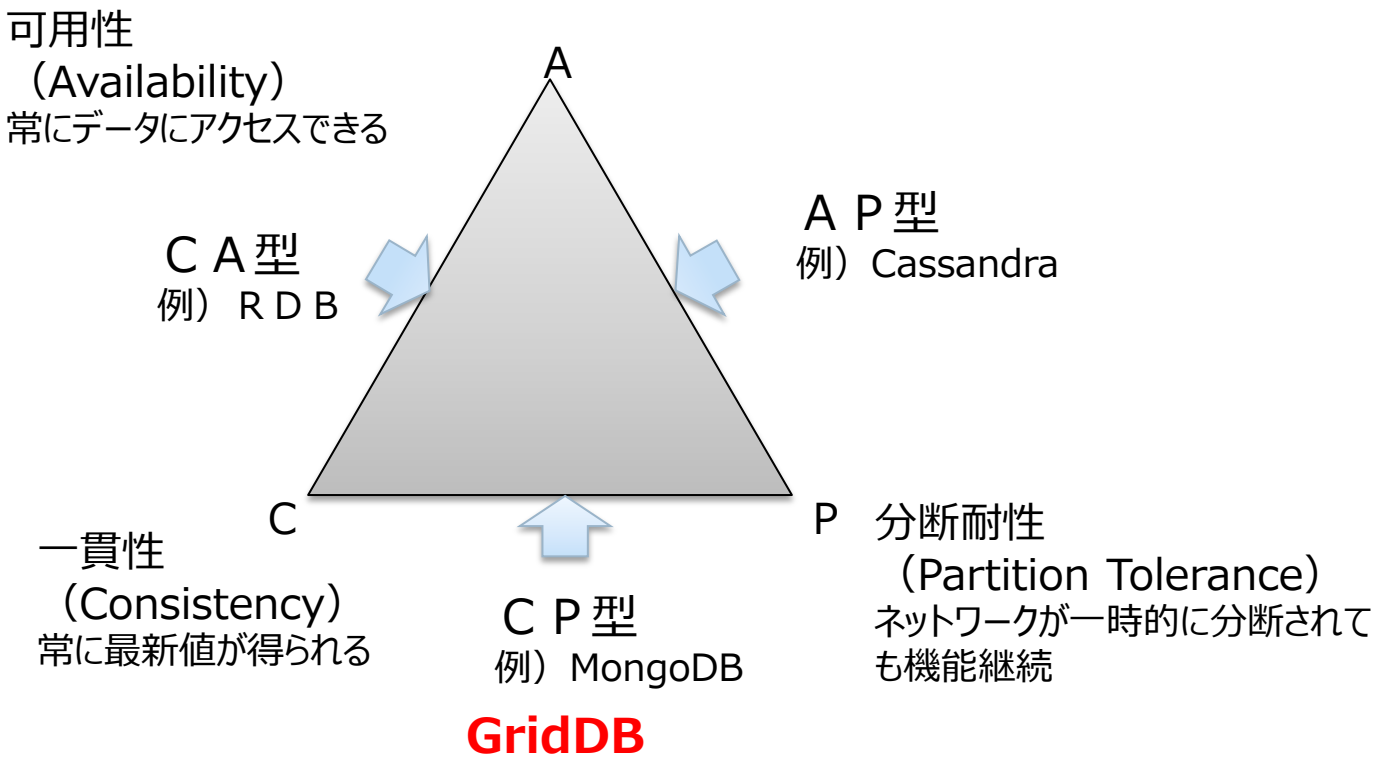
- ノード間で自律的、動的にマスタノードを決定。単一故障点を排除
- マスタがデータ配置（オーナ/バックアップ）を決定



③一貫性と可用性

GridDBはCP型

- **CAP定理:** E. Brewer, "Towards Robust Distributed Systems“[1]



[1] Proc. 19th Ann. ACM Symp.Principles of Distributed Computing (PODC 00), ACM, 2000, pp. 7-10;

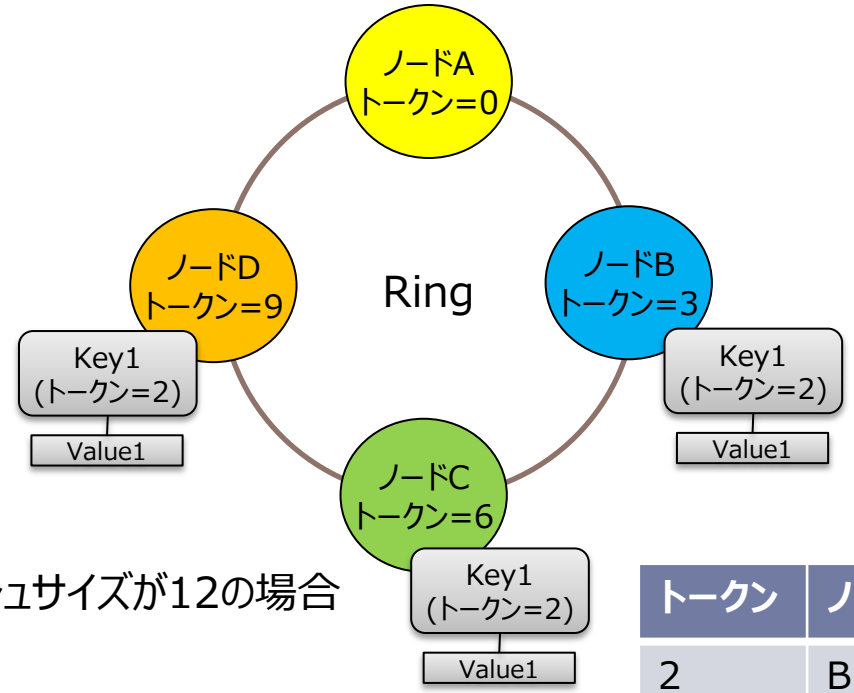
発表内容

1. スケールアウト型データベースGridDB
2. アーキテクチャとメカニズムについて代表的なNoSQL DB（Cassandra、MongoDB）との比較
   - データモデル、クラスタ管理、一貫性と可用性について
   - クラスタ管理のメカニズムについて
     - データ分散方法、レプリケーション、フェイルオーバ
3. まとめ

（A）Cassandra

- **単一トークンによるコンシステントハッシュ法**

データ分散の例：SingleStrategy
まずハッシュ関数で決められたノードに配置。
続いて時計回りでリング状の次のノードに配置。



※ハッシュサイズが12の場合

トークン	ノード
2	B, C, D

データモデル	カラム型
クラスタ管理	P2P方式
CAP	AP型
データ分散方法	コンシステント・ハッシュ
レプリケーション	プライマリ/セカンダリの区別なし
フェイルオーバ	P2P方式

最新バージョン：3.11

一貫性の調整（ONE、ALL）

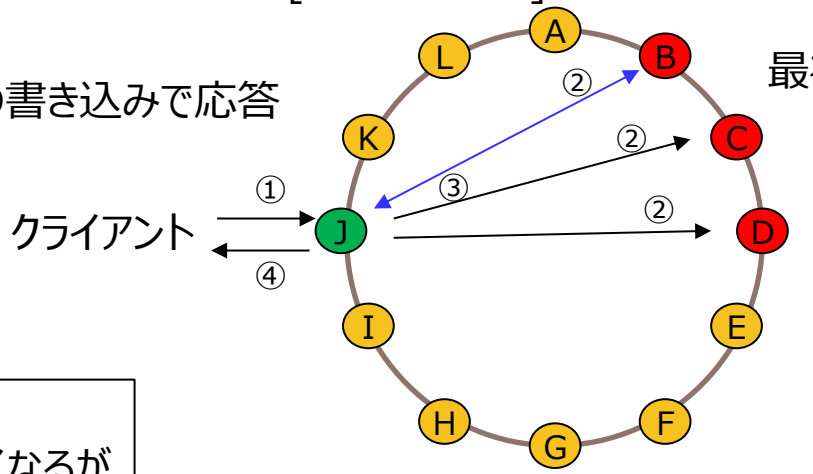
一貫性と可用性・性能はトレードオフ

※レプリカ数が３の場合

(ONE)

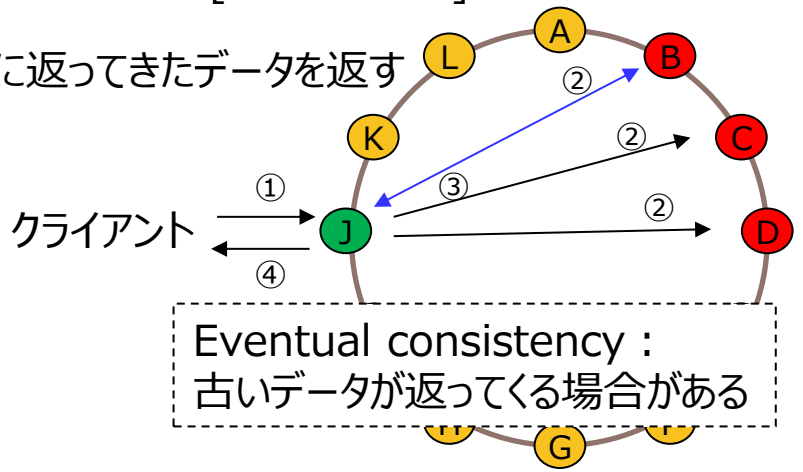
１ノードの書き込みで応答

[書込みケース]



[読込みケース]

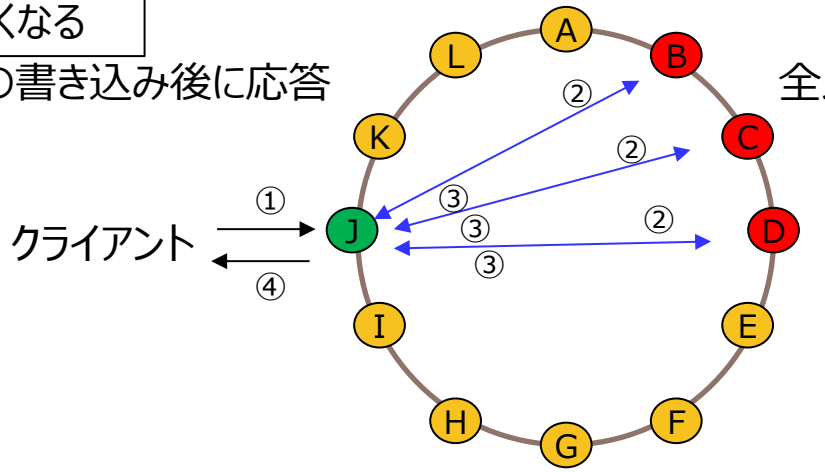
最初に返ってきたデータを返す



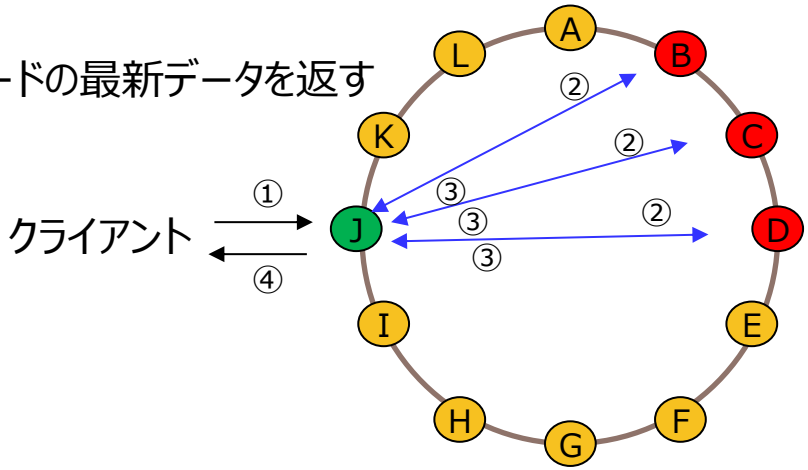
Eventual consistency：
古いデータが返ってくる場合がある

(ALL)
一貫性は高くなるが
可用性は低くなる

全ノードの書き込み後に応答



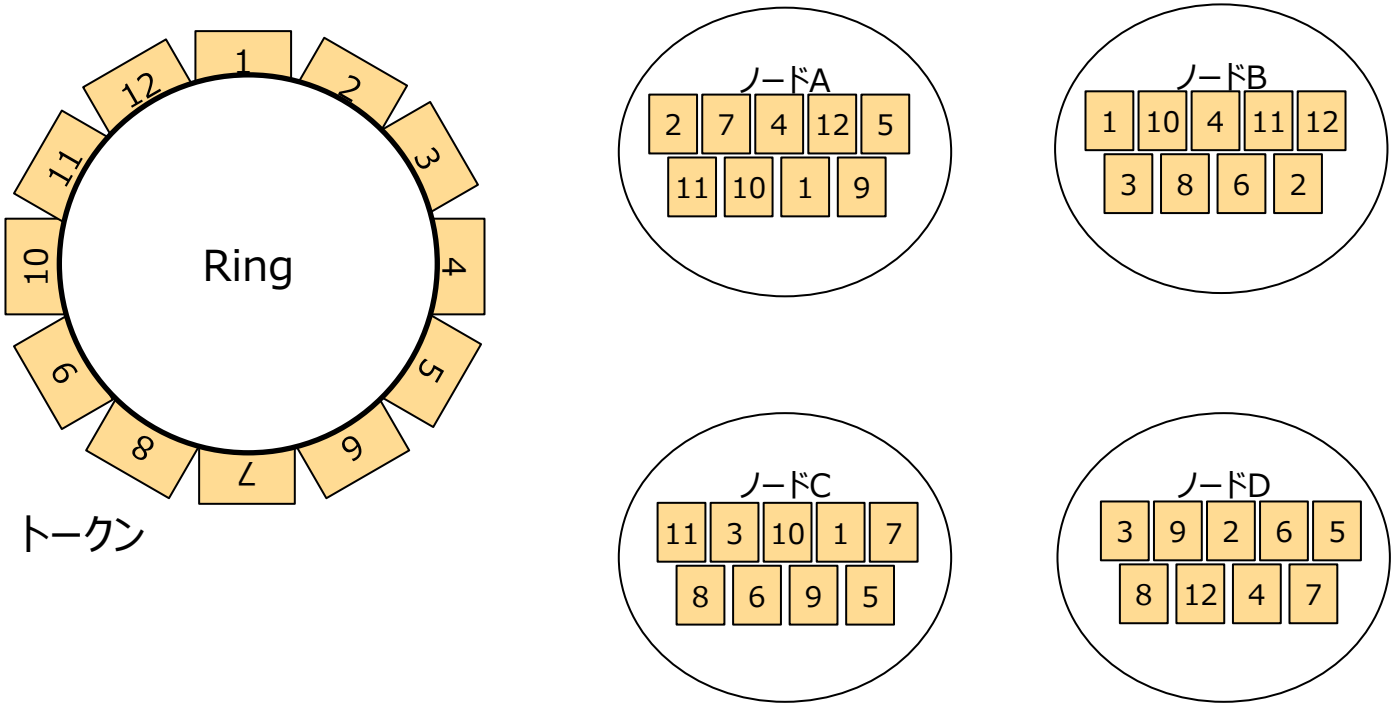
全ノードの最新データを返す



①稼働状態

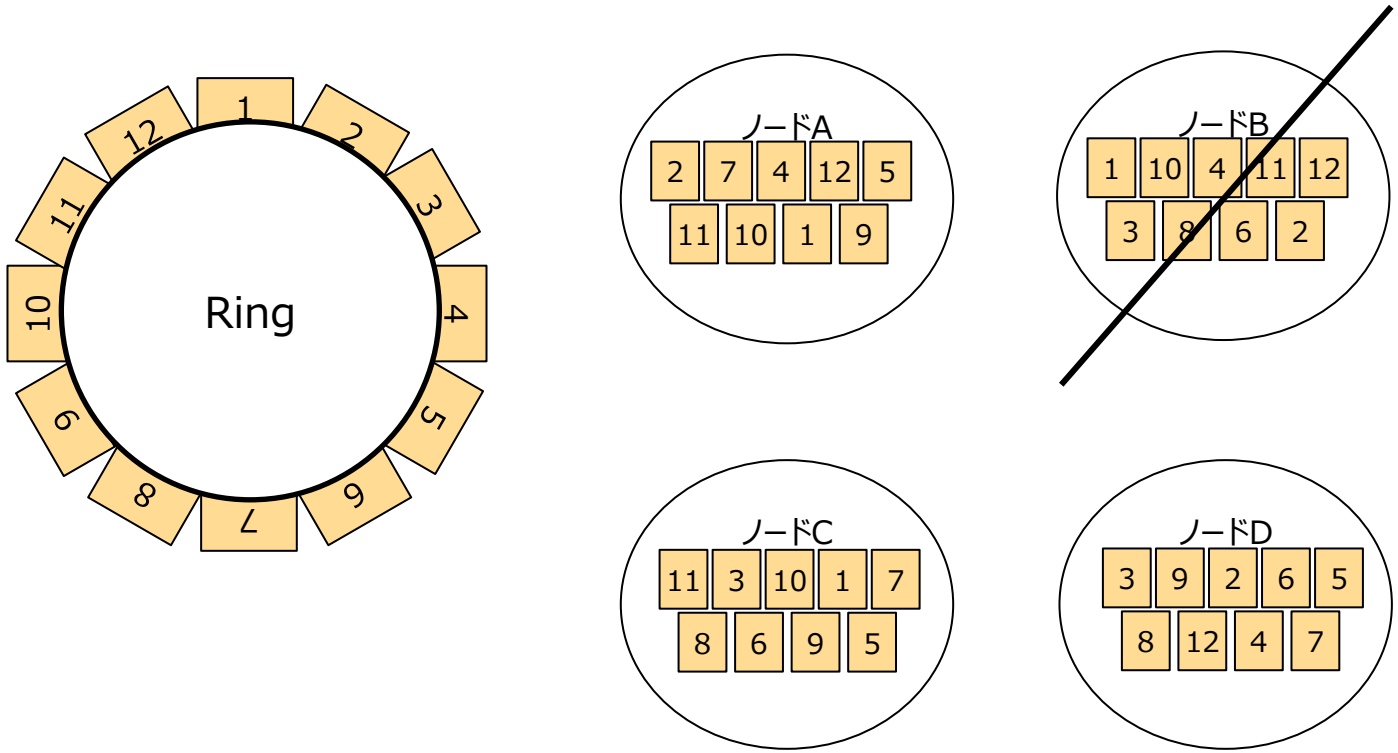
- **仮想ノード(V1.2～)によるコンシステントハッシュ法**

※レプリカ数が３の場合



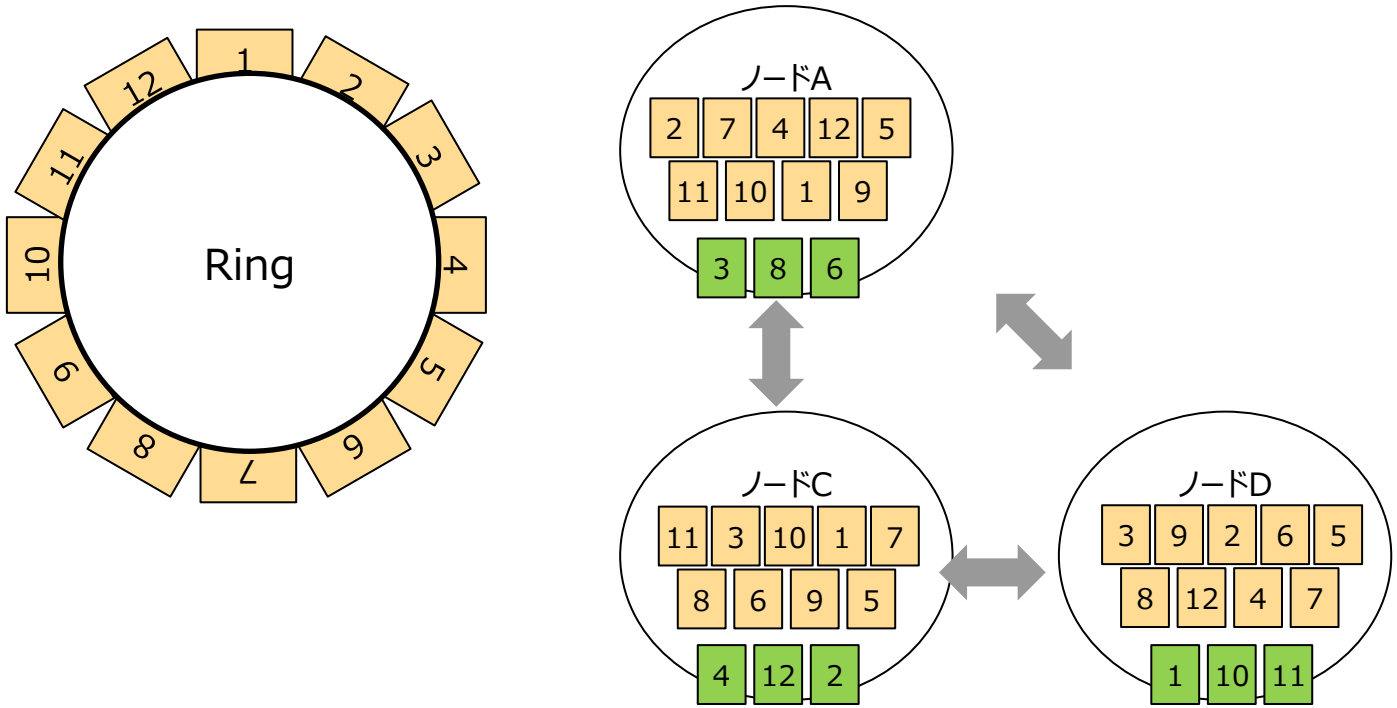
１つのノードに複数のランダムなトークン値を割当。担当範囲を複数に分割

②ノードBがノードダウン



③安定状態

P2Pにより自動バランシング

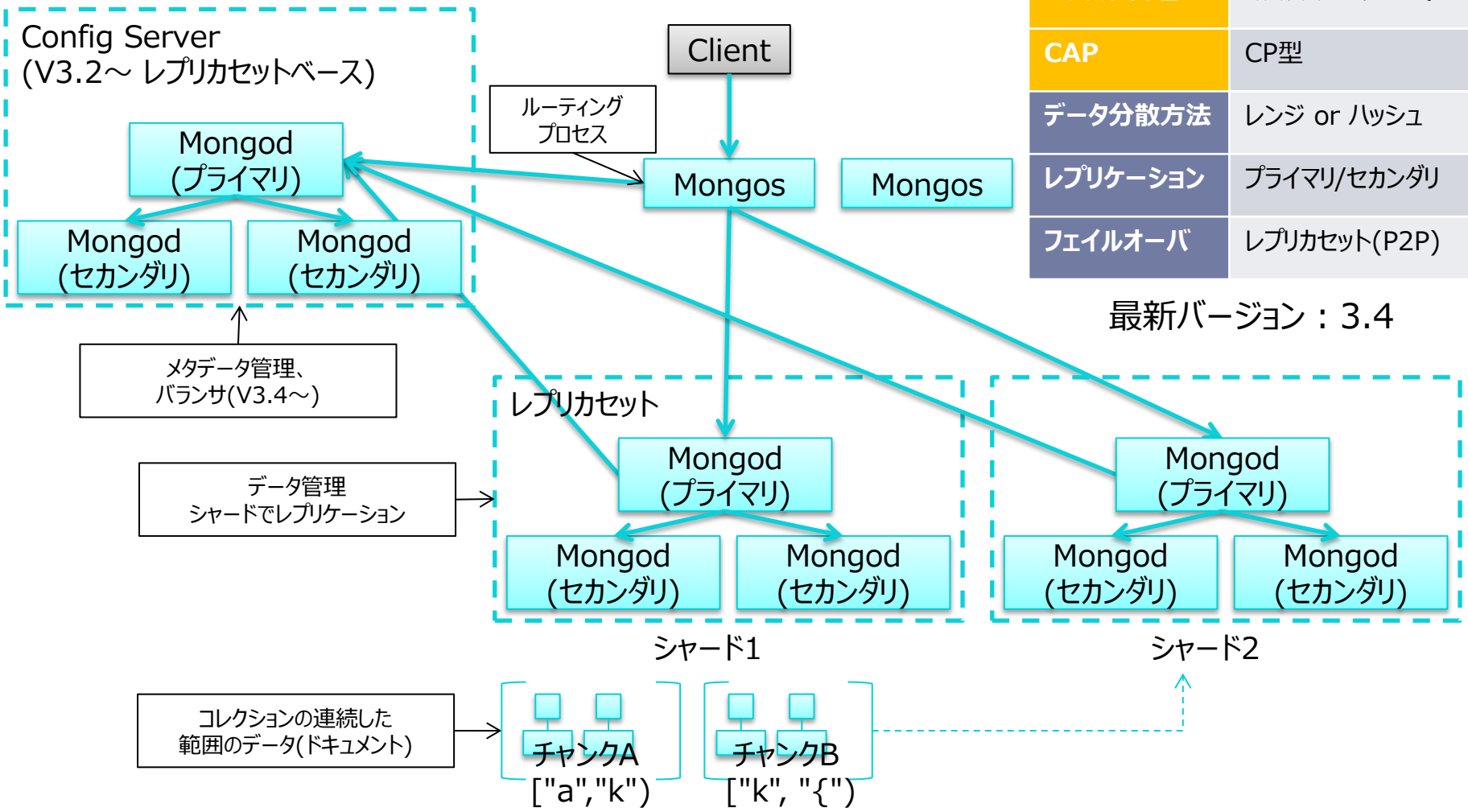


P2Pでトークンを再配置してバランシング

（B）MongoDB

データモデル	ドキュメント型
クラスタ管理	マスタスレーブ方式
CAP	CP型
データ分散方法	レンジ or ハッシュ
レプリケーション	プライマリ/セカンダリ
フェイルオーバ	レプリカセット(P2P)

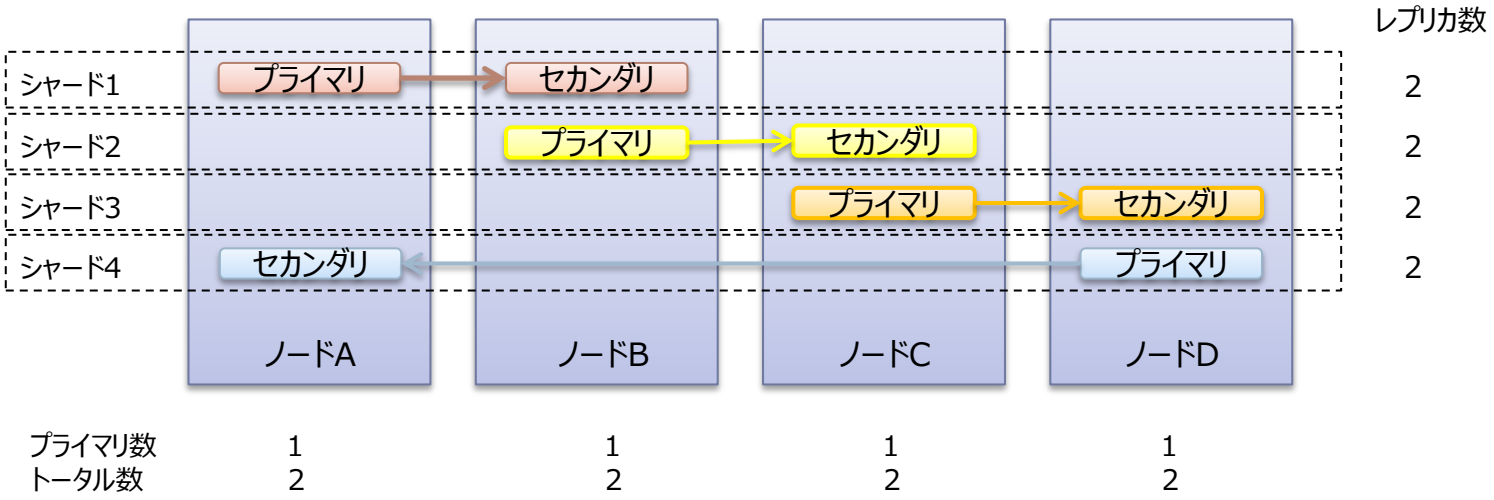
最新バージョン：3.4



①稼働状態

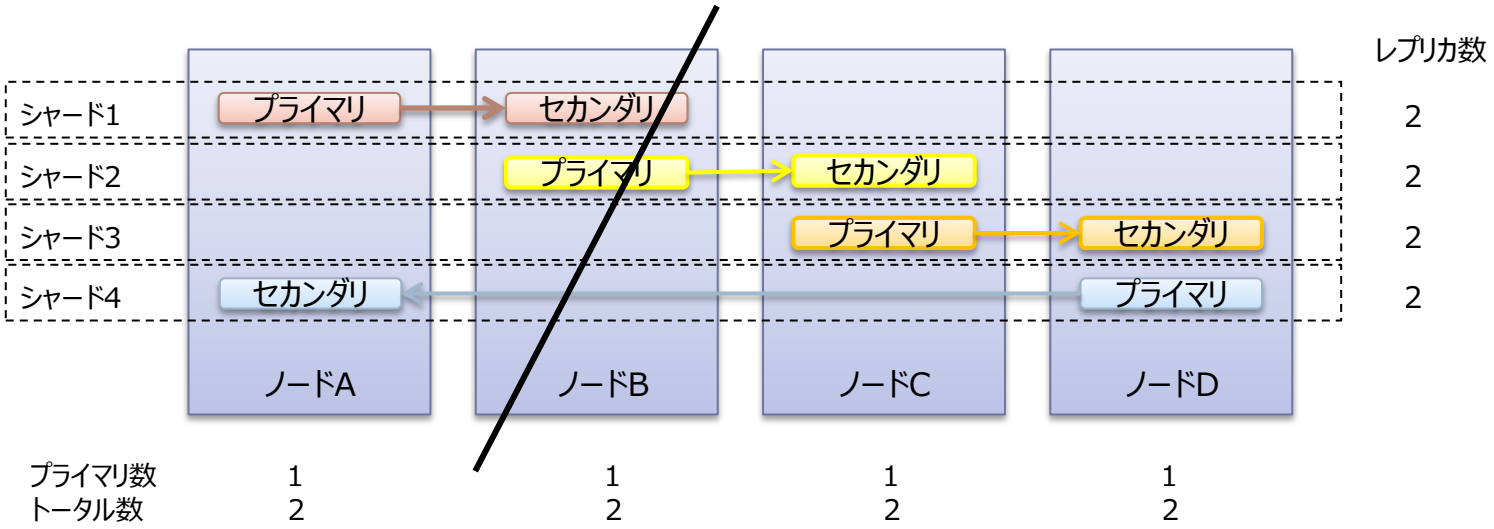
サーバ(プロセス)数＝シャード数×レプリカ数

※Config Server以外のMongodサーバを同一ノード内に混在させた場合
※レプリカ数が2の場合



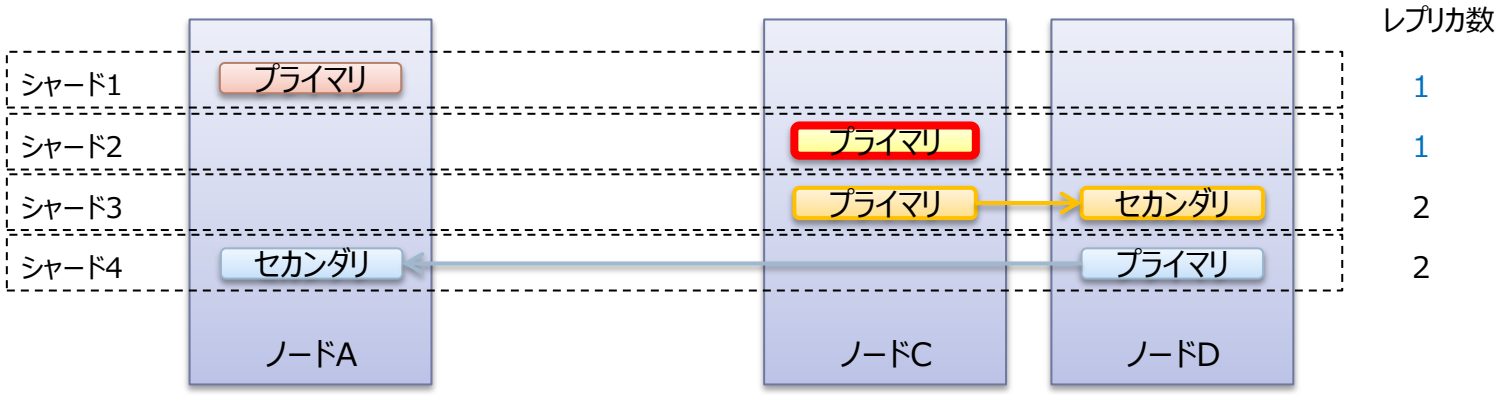
4ノード、8サーバ(プロセス)、4シャード
各ノードのプライマリは1個、セカンダリは1個
レプリカ数はすべて２個

②ノードBがノードダウン



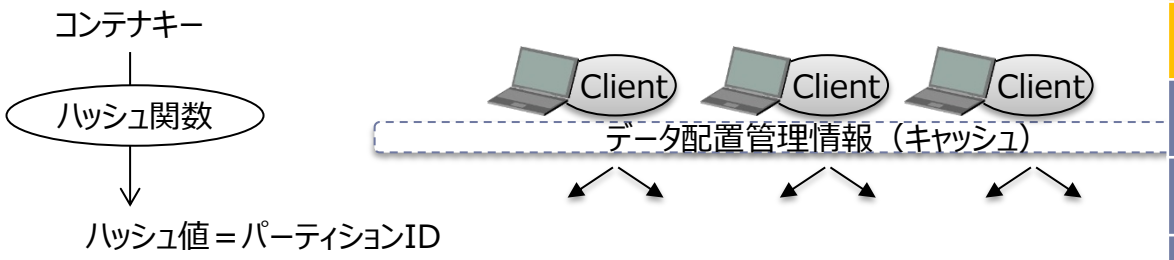
③フェイルオーバ

P2Pでセカンダリがプライマリに自動切り替え

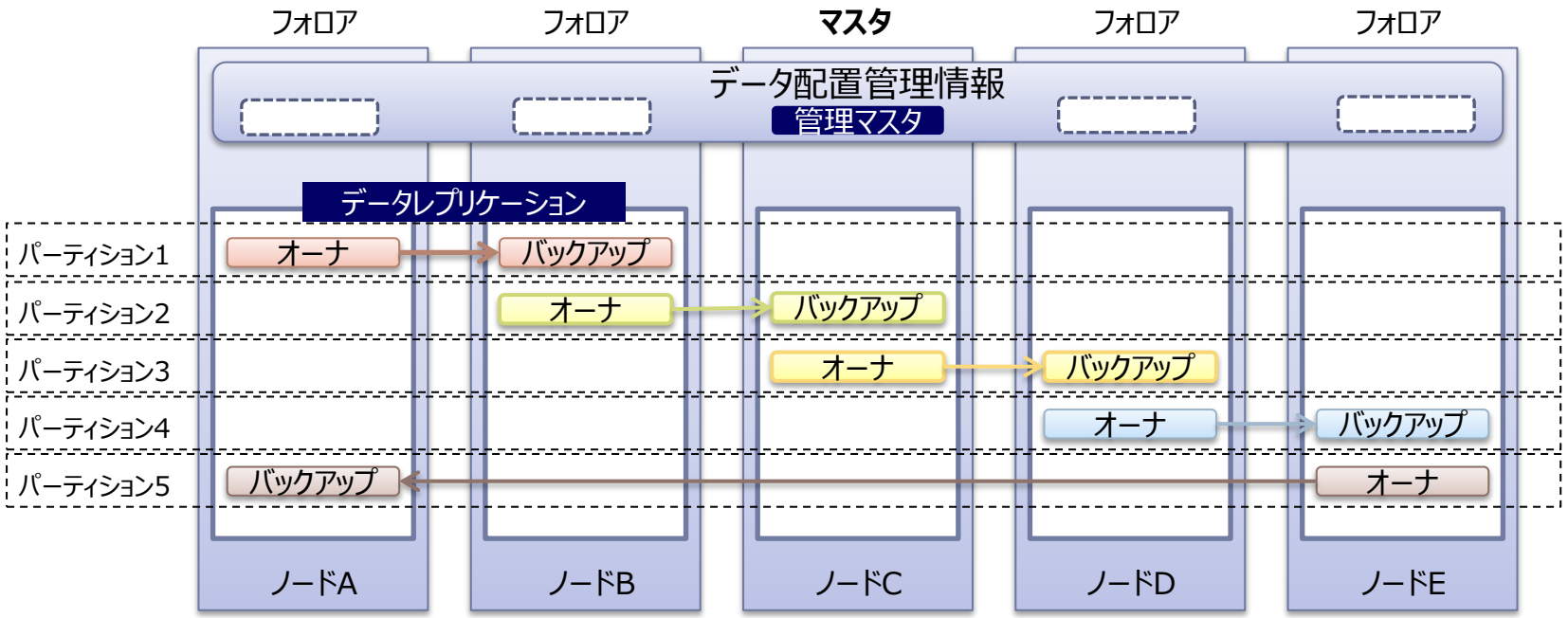


レプリカ数の少ないものが発生
・・・サーバの手動追加が必要
プライマリ数の偏りが発生
・・・プライマリの手動切替が必要

（C）GridDB



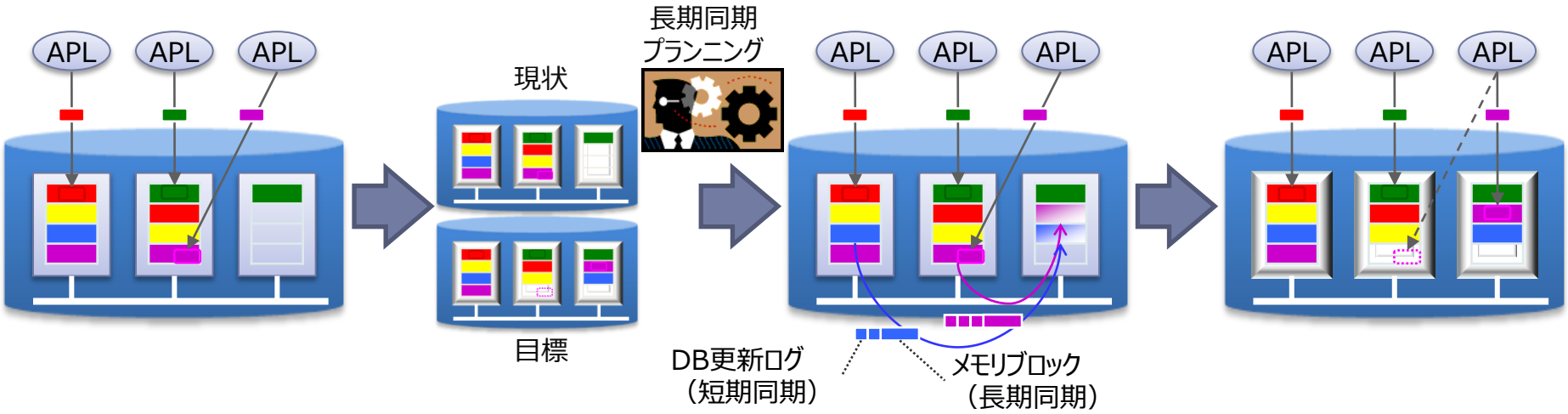
データモデル	コンテナ型
クラスタ管理	ハイブリッド型
CAP	CP型
シャーディング	ハッシュ
レプリケーション	オーナ/バックアップ
フェイルオーバ	ハイブリッド型



自律データ再配置技術（ADDA）

ADDA：Autonomous Data Distribution Algorithm

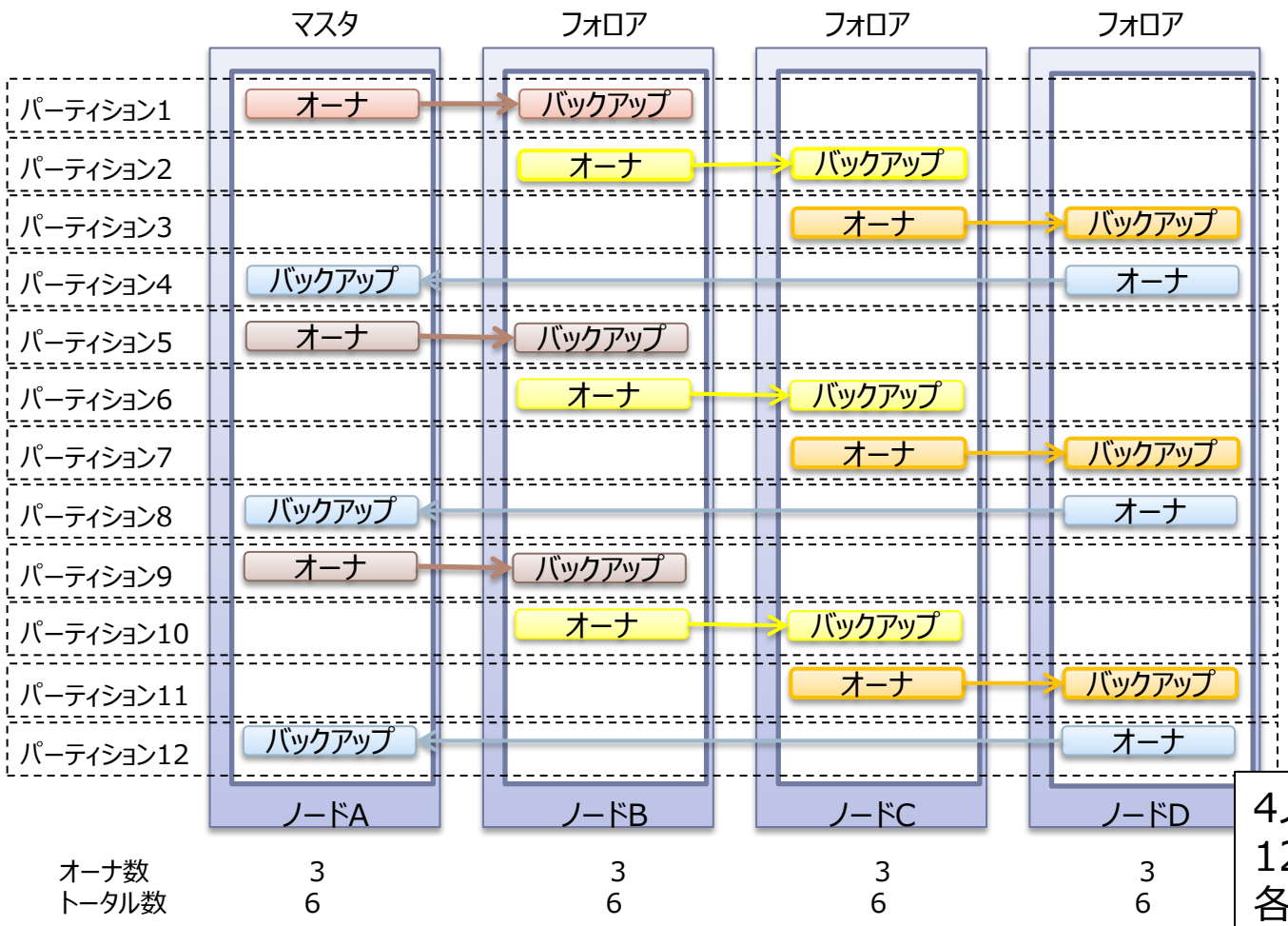
- インバランス状態を検知、長期同期プランニング
- ２種類のデータを使ってバックグラウンド高速同期、完了後切替
  - ✓ DB更新ログ、メモリブロック



①負荷インバランス検知　②長期同期プランニング　③長期同期実行　④アクセス切替

①稼働状態

ノード、パーティション単位にデータ分散配置

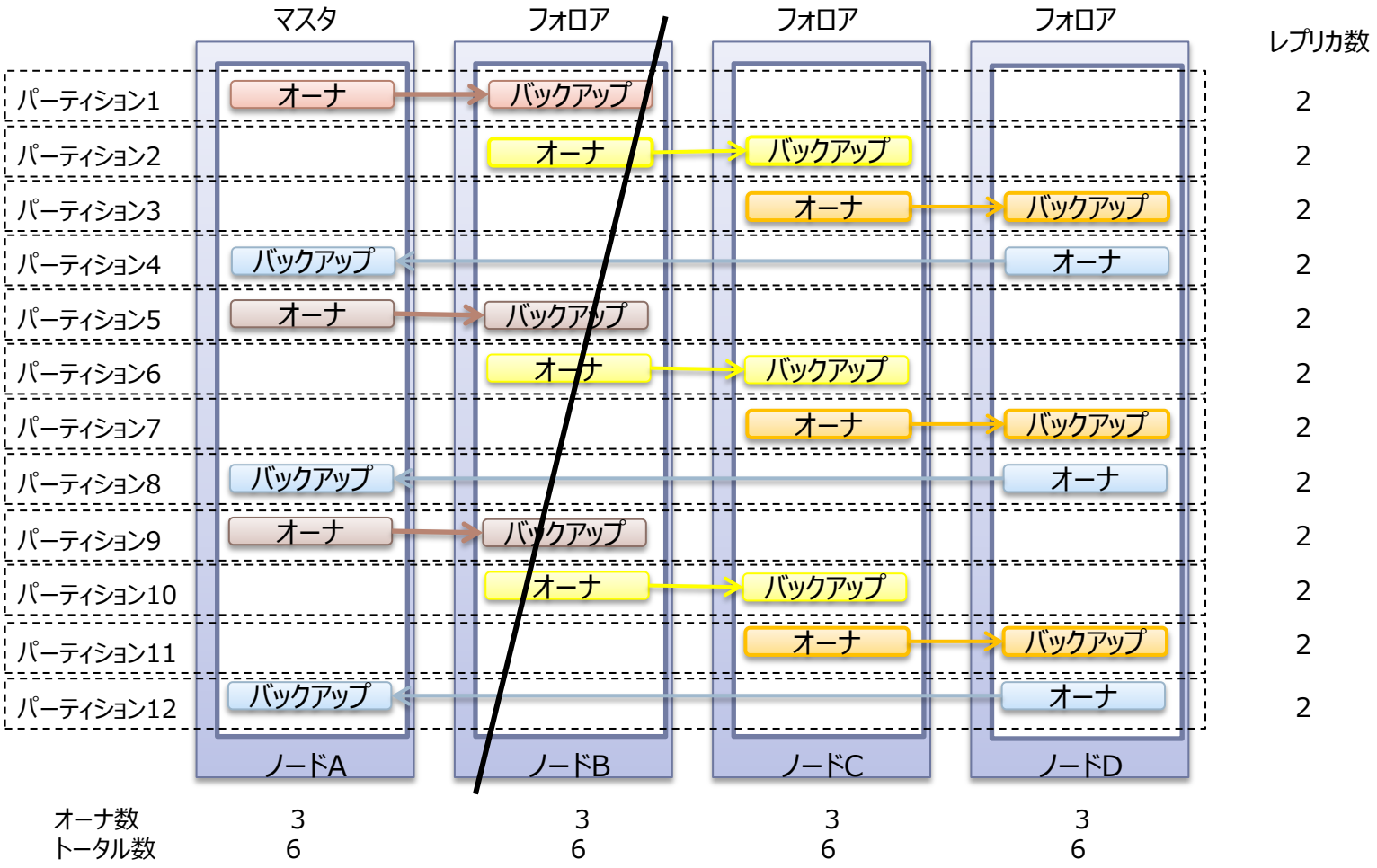


※レプリカ数が2の場合

レプリカ数
2
2
2
2
2
2
2
2
2
2
2
2

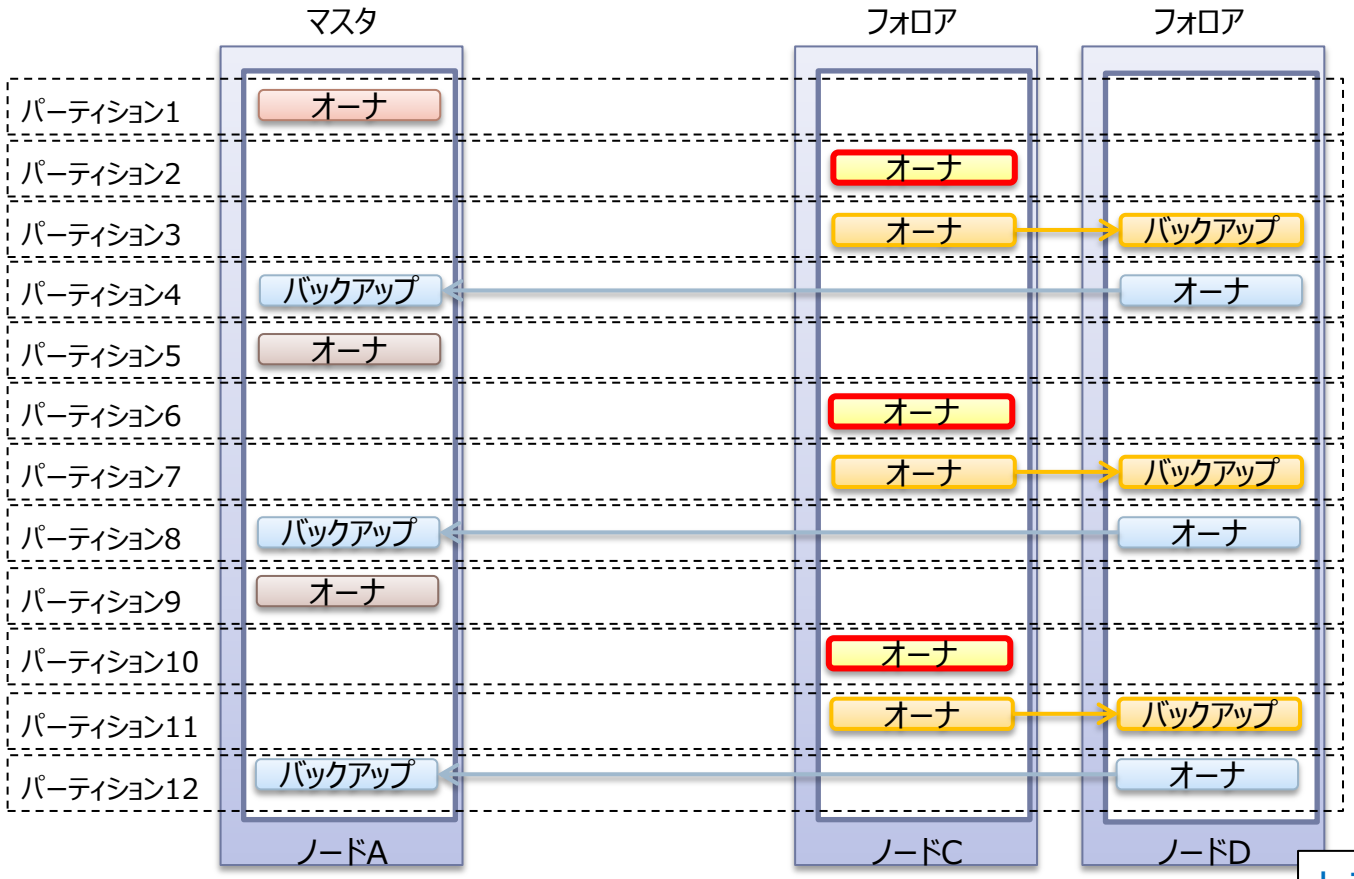
4ノード、4サーバ、
12パーティション
各ノードのオーナは3個、
バックアップは3個
レプリカ数はすべて２個

②ノードBがノードダウン



③フェイルオーバ

マスタが即座にバックアップをオーナに自動切り替え



	レプリカ数	レプリカ数
パーティション1	2	1
パーティション2	2	1
パーティション3	2	2
パーティション4	2	2
パーティション5	2	1
パーティション6	2	1
パーティション7	2	2
パーティション8	2	2
パーティション9	2	1
パーティション10	2	1
パーティション11	2	2
パーティション12	2	2

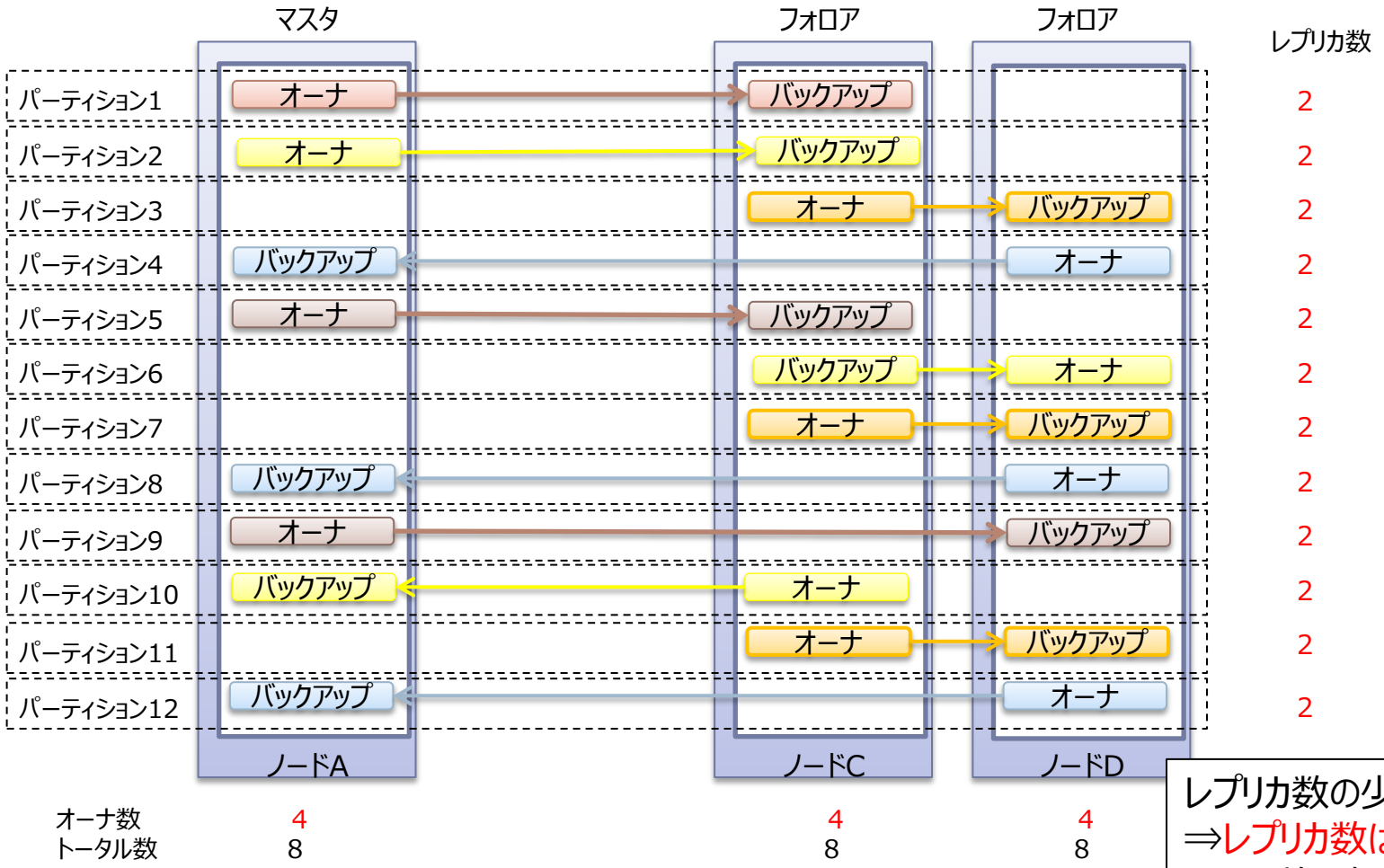
	ノードA	ノードC	ノードD
オーナ数	3	6	3
トータル数	6	6	6

レプリカ数の少ないものが発生

オーナ数の偏りが発生

④安定状態

ADDAにより自動バランシング



レプリカ数の少ないものが発生
⇒レプリカ数はすべて2個に
オーナ数の偏りが発生
⇒オーナ数はすべて4個に

アーキテクチャ・メカニズムの比較（まとめ）

	Cassandra	MongoDB	GridDB
データモデル	ワイドカラム型	ドキュメント型	**キー・コンテナ型**
クラスタ管理	P2P方式	マスタスレーブ方式	**ハイブリッド型** （P2Pによるマスタ自動選出）
CAP	AP型	CP型	CP型
シャーディング	コンシステント・ハッシュ	ハッシュ or レンジ	ハッシュ
レプリケーション	プライマリ/セカンダリ の区別なし	プライマリ/セカンダリ (サーバ単位)	オーナ/バックアップ (パーティション単位)
フェイルオーバ	P2P方式	P2P方式	ハイブリッド型

一貫性(C)/可用性(A)を高める仕掛け	調整可能な一貫性レベル (一貫性アップ)	P2Pによるレプリカセット (可用性アップ)	**ADDA** (可用性アップ)
一貫性	性能とトレードオフ	常に最新データ	常に最新データ
可用性、スケーラビリティ	P2Pによる自動バランシング	レプリカ数は減ったまま ⇒レプリカ追加操作が必要	マスタによる自動バランシング(ADDA)

まとめ

- **ビッグデータ×IoT向けスケールアウト型データベースGridDBを代表的なNoSQL DBであるCassandra、MongoDBと比較しました。**
  - ①データモデル、②クラスタ管理、③一貫性と可用性、について
  - クラスタ管理のメカニズム（データ分散方法、レプリケーション、フェイルオーバ）について
- **GridDBは高い一貫性と可用性をもつIoTに適したデータベースです。**

GridDBのオープンソース版がありますので、是非とも使ってみてください。

GridDBに関する情報

- GridDB 製品情報
  http://www.toshiba.co.jp/cl/pro/bigdatapf/
- GridDB デベロッパーズサイト：https://griddb.net/
- GridDB OSSサイト：https://github.com/griddb/griddb_nosql/
- OSSを利用したビッグデータ分析環境　GridData Analytics Cloud
  https://www.griddata-analytics.net/
- AWS Marketplace: GridDB Community Edition (CE)
  https://aws.amazon.com/marketplace/pp/B01N5ASG2S
- AWS Marketplace: GridDB Standard Edition (SE)
  https://aws.amazon.com/marketplace/pp/B01N9QMCMF/
- Twitter　http://twitter.com/GridDBCommunity/
- Facebook　http://fb.me/griddbcommunity/

デベロッパーズサイト(https://griddb.net/)の主なコンテンツ一覧

ホワイトペーパ：

- GridDB®とは
- GridDB と Cassandra のパフォーマンスとスケーラビリティ
  - Microsoft Azure 環境における YCSB パフォーマンス比較
- GridDB Reliability and Robustness

など

ブログ：

- CAP 定理と GridDB
- Raspberry Piチュートリアル：KairosDBコネクタを介してGridDBに温度データを送信する
- Docker上でGridDBを実行する
- IoT産業におけるGridDB導入事例
- GridDB Azureクラスタの構築
- YCSB向けGridDBコネクタを使ってみよう

など


TOSHIBA
Leading Innovation