TOSHIBA

もうSQLとNoSQLを選ぶ必要はない！？
～ 両者を備えたスケールアウトデータベースGridDB ～

東芝デジタルソリューションズ株式会社
ソフトウェア＆AIテクノロジーセンター　技監
服部　雅一

GridDB™
Highly Scalable Database for IoT

自己紹介

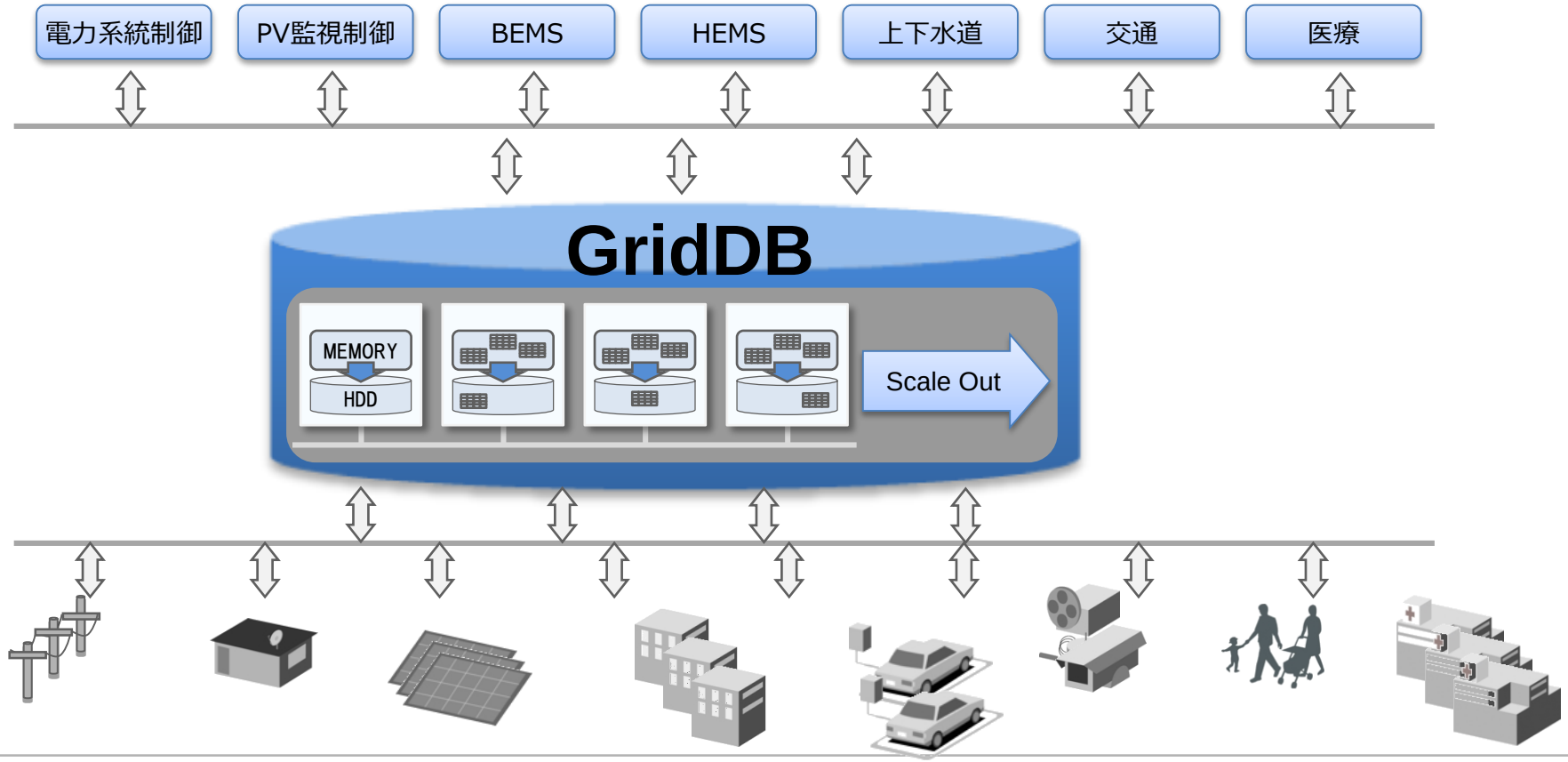
- 東芝デジタルソリューションズ株式会社
  ソフトウェア＆AIテクノロジーセンター　技監
  服部 雅一　masakazu.hattori@toshiba.co.jp

- 東芝入社　機械設計自動化、プラント監視自動化、電力系統最適化、
  などのAIシステムの研究開発、および実用化
- その後　XMLデータベースなどデータベース関連の研究開発、および製品化
- 2011年　ビッグデータ向けデータベースのチーフアーキテクトとして研究開発を開始
- 2013年　GridDB製品化

- 平成21年(社)情報処理学会　喜安記念業績賞、
- 第55回(財)電気科学技術奨励会電気科学技術奨励賞(電気科学技術奨励賞委員会会長)、等

スケールアウト型データベースGridDB

- ビッグデータ／IoT向けスケールアウト型データベース
- 製品化(2013年)、OSS化(2016年)、V4.1(2018年10月)
- 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に適用中



ご紹介内容

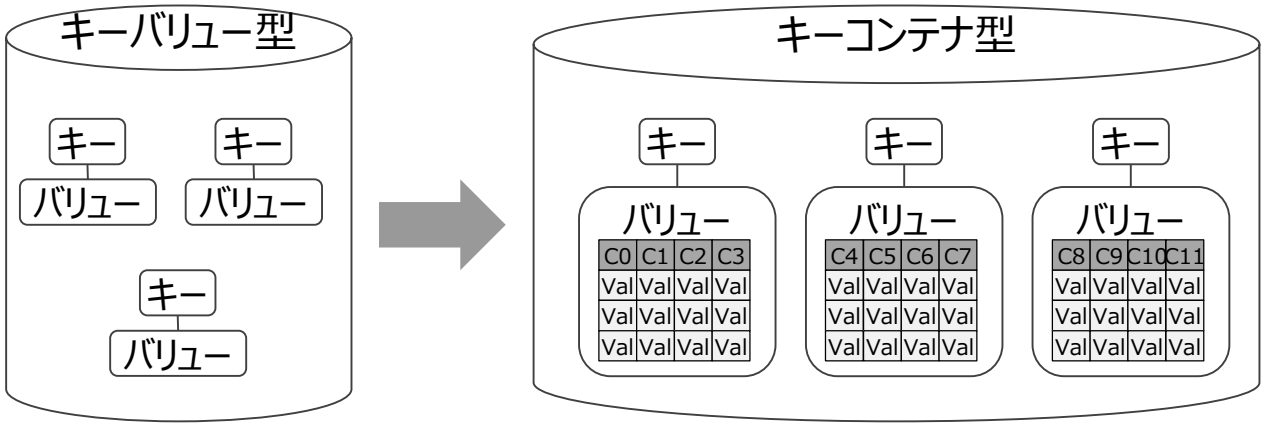
- GridDBの特長1
  - IoT指向のデータモデル
  - 高い信頼性と可用性
  - スケーラビリティ
- GridDBの特長2
  - NoSQL+SQL
  - SQLにおける分散並列処理
- 適用事例

GridDBの特長1

IoT指向のデータモデル	• データモデルはキー・コンテナ。コンテナ内でのデーター貫性を保証 • 時系列データ管理する特別な機能 • 過去データをコールド保存する長期アーカイブ機能
高い信頼性と可用性	• データの複製をノード間で自動的に実行 • ノード障害があってもフェールオーバによりサービス継続 • 数秒から数十秒の切替え時間
高いスケーラビリティ	• 少ないノード台数で初期投資を抑制 • 負荷や容量の増大に合わせたノード増設が可能 • 自律データ再配置により、高いスケーラビリティを実現
高性能なNoSQL+SQL	• 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DB • メモリやディスクの排他処理や同期待ちを極力排除 • SQLにおける分散並列処理

データモデル

- NoSQL型でよく採用されているキー・バリューを拡張
- 順序に関係無くレコードが格納されるコレクションコンテナ
- 時間順にレコードが格納される時系列コンテナ
  時系列レコードを圧縮する機能や期限解放する機能
- コンテナ内でのデーター貫性を保証



"siteA_equip"

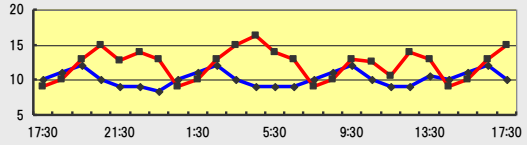
コレクション(Collection)

id	name	specification
equip001	変圧器1	xxx変圧器
equip002	変圧器2	yyy変圧器
equip003	遮断機1	xxx遮断機
equip004	遮断機2	yyy遮断機
equip005	ケーブル1	zzzケーブル
…	…	…

"siteA_s012"

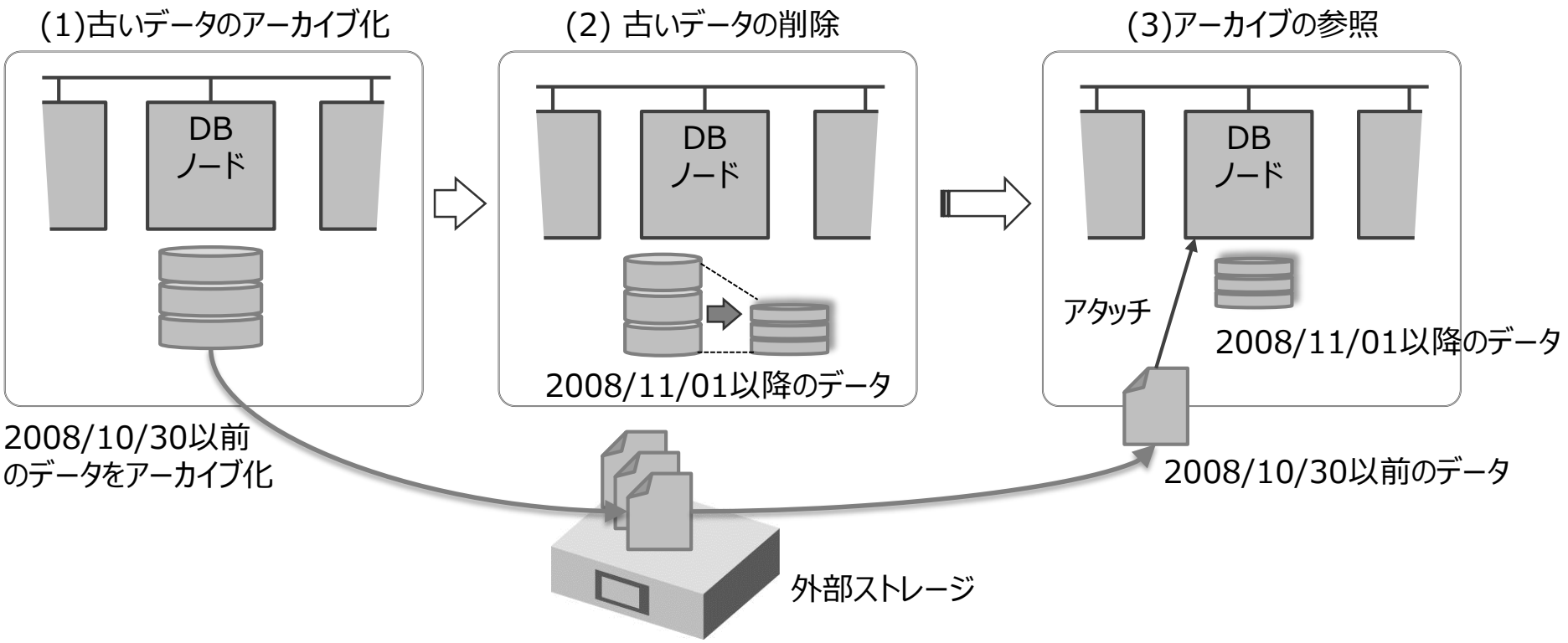
時系列(TimeSeries)

timestamp	heat_rate	temperature
4/28/20011 …	78.3	47.9
4/28/20011 …	82.9	63.4
…	…	…



長期アーカイブ機能

- 適切に容量を抑えつつ、長期間データをコールド保存
- 効率の良いアーカイブ処理

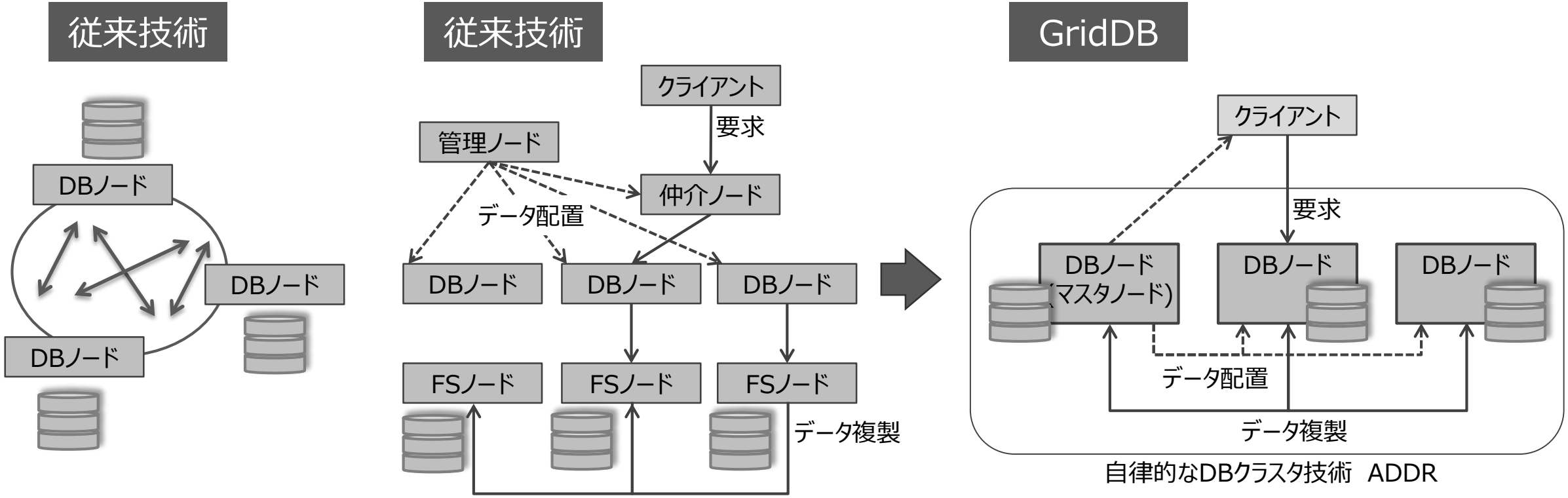


GridDBの特長1

IoT指向のデータモデル	• データモデルはキー・コンテナ。コンテナ内でのデーター貫性を保証 • 時系列データ管理する特別な機能 • 過去データをコールド保存する長期アーカイブ機能
高い信頼性と可用性	• データの複製をノード間で自動的に実行 • ノード障害があってもフェールオーバによりサービス継続 • 数秒から数十秒の切替え時間
高いスケーラビリティ	• 少ないノード台数で初期投資を抑制 • 負荷や容量の増大に合わせたノード増設が可能 • 自律データ再配置により、高いスケーラビリティを実現
高性能なNoSQL+SQL	• 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DB • メモリやディスクの排他処理や同期待ちを極力排除 • SQLにおける分散並列処理

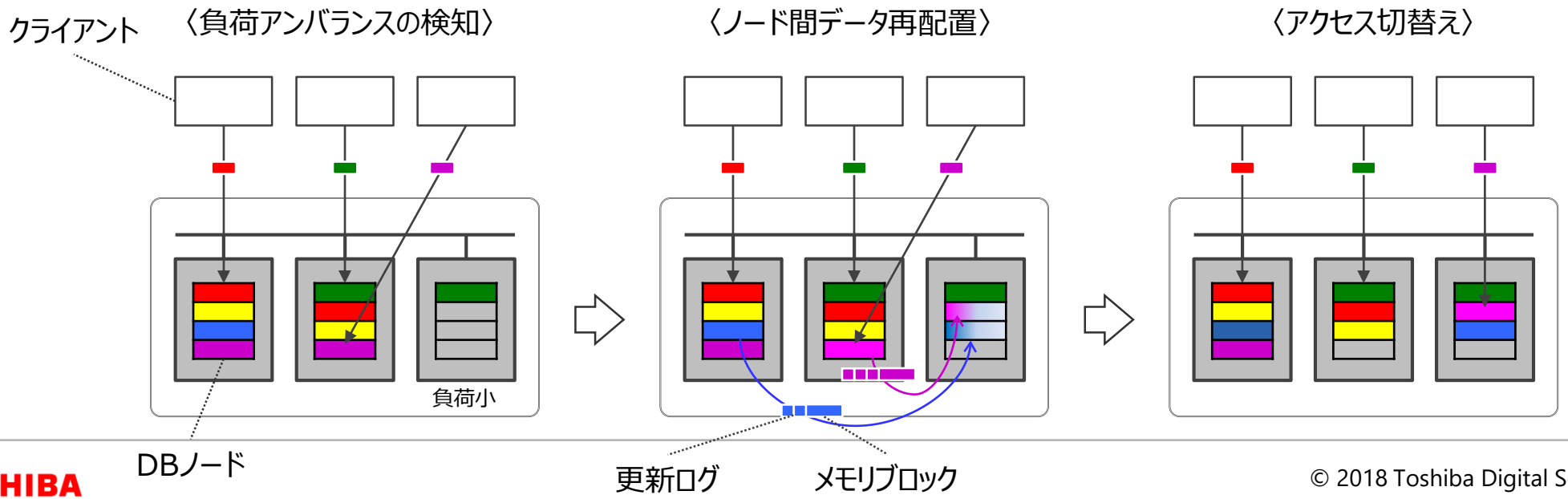
DBクラスタ

- 自律データ再配置技術(ADDA：Autonomous Data Distribution Algorithm)
- データを分散化するゆえにデータの一貫性が弱くなり、
  一貫性やスケール性を求めるとパフォーマンスが落ちる、という大きな欠点を解決



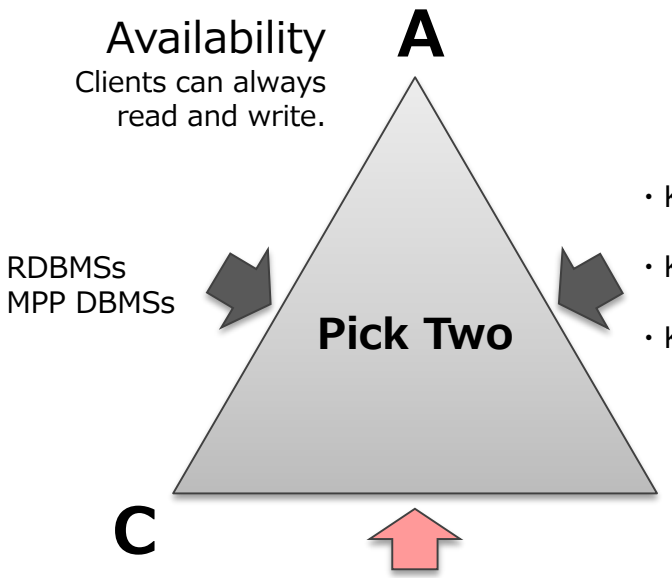
具体的な挙動

- マススレーブモデルの改良
  - ノード間でマスタノードを自動選択。管理サーバがクラスタ内に存在せず、SPOFを完全排除
  - ノード過半数を占めたサブクラスタのみがサービス可能となるクオーラムポリシーにより、スプリットブレインを完全排除
- 自律データ再配置技術の開発
  - (マスターノードが)ノード間アンバランス、レプリカ欠損を検知⇒バックグラウンドでデータ再配置
  - 2種類のレプリカデータを使って高速同期、完了後切替え



CAP定理

- GridDBはCP。強い一貫性を持ち、古いデータが見えてしまうことは無い。



BASE
- Basically Available
- Soft State
- Eventually Consistency(結果整合)

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

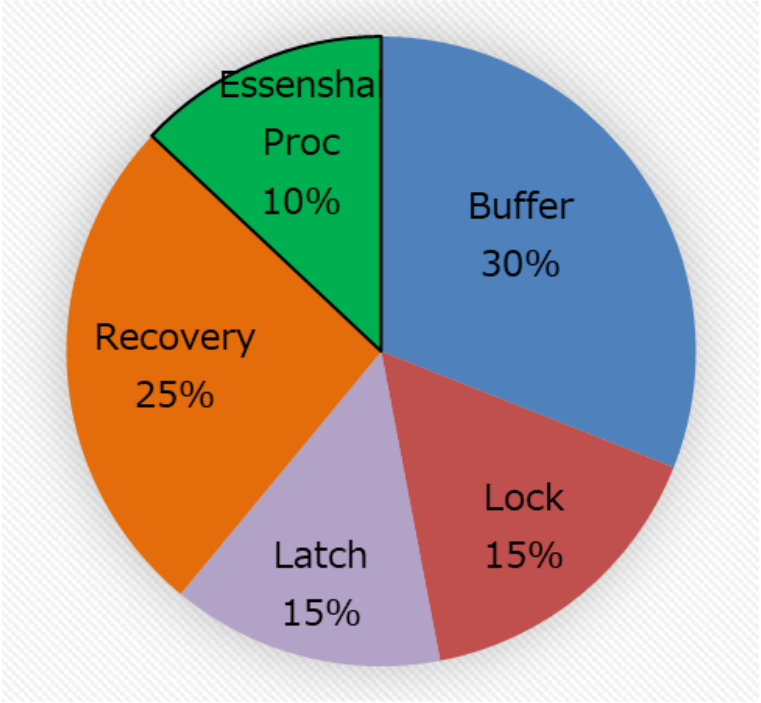
GridDBの特長2

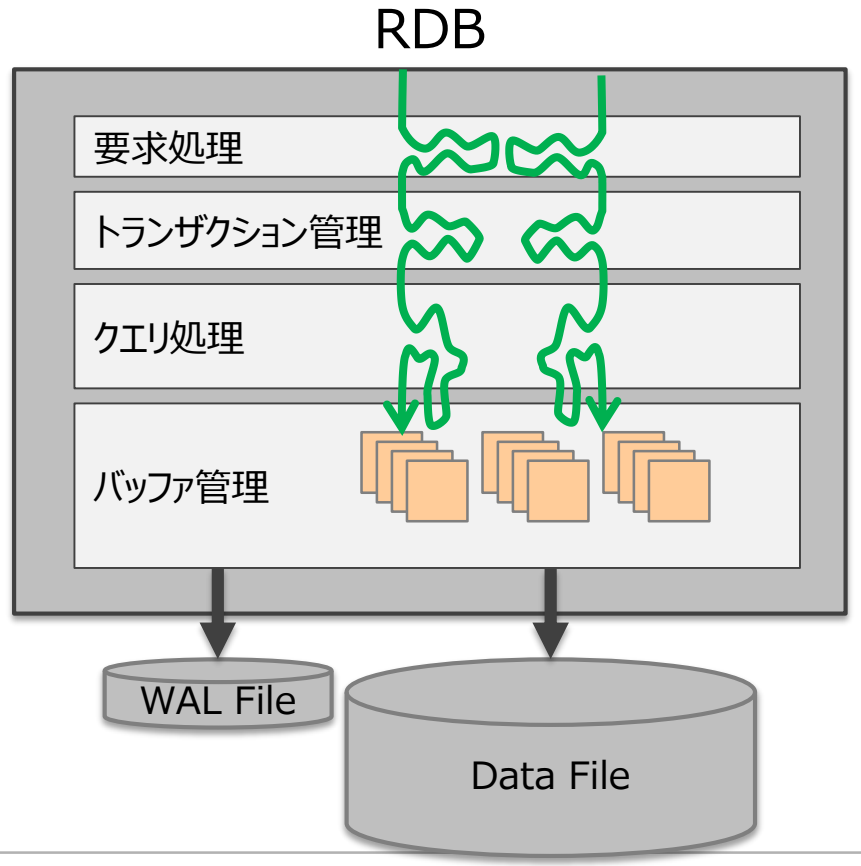
IoT指向のデータモデル	• データモデルはキー・コンテナ。コンテナ内でのデーター貫性を保証 • 時系列データ管理する特別な機能 • 過去データをコールド保存する長期アーカイブ機能
高い信頼性と可用性	• データの複製をノード間で自動的に実行 • ノード障害があってもフェールオーバによりサービス継続 • 数秒から数十秒の切替え時間
高いスケーラビリティ	• 少ないノード台数で初期投資を抑制 • 負荷や容量の増大に合わせたノード増設が可能 • 自律データ再配置により、高いスケーラビリティを実現
高性能なNoSQL+SQL	• メモリを主、ストレージを従としたハイブリッド型インメモリDB • メモリやディスクの排他処理や同期待ちを極力排除 • SQLにおける分散並列処理

DBMSにおけるCPU使途

- 本質的なデータ処理に費やすCPU使用率は10%強
- バッファ管理、ラッチ、ロック、リカバリ、等で大半のCPUを消費

Harizopoulos, S. et al, “OLTP Through the Looking Glass, and What We Found There”, SIGMOD 2008



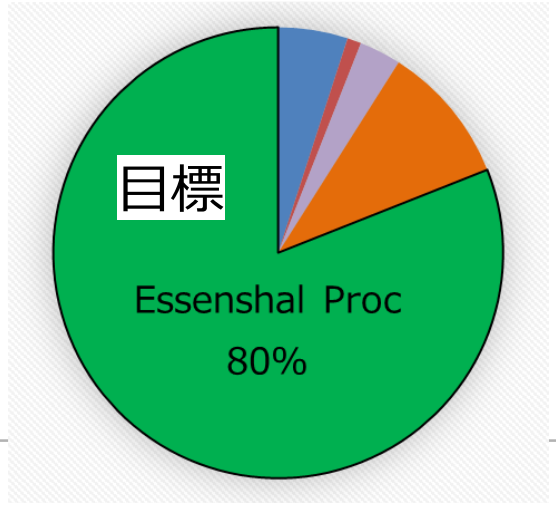


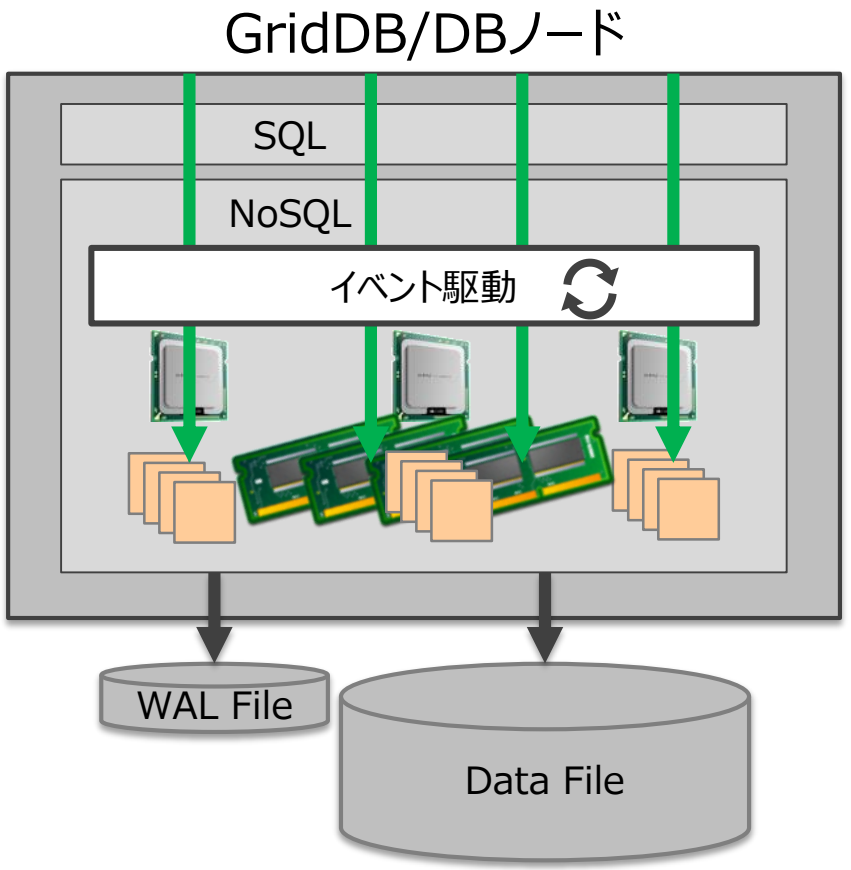
GridDBの考え方

- 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DB

ラッチ	• 大半のラッチを無くし • 時分割的な挙動で代用
メモリ	• 適材適所のメモリプール • それ以前にメモリコピーを減らす
バッファ管理	• ブロックサイズをかなり大きく • メモリ指向のバッファ管理
リカバリ	• WAL(REDO LOG)を踏襲しつつ • リカバリ処理を軽量化

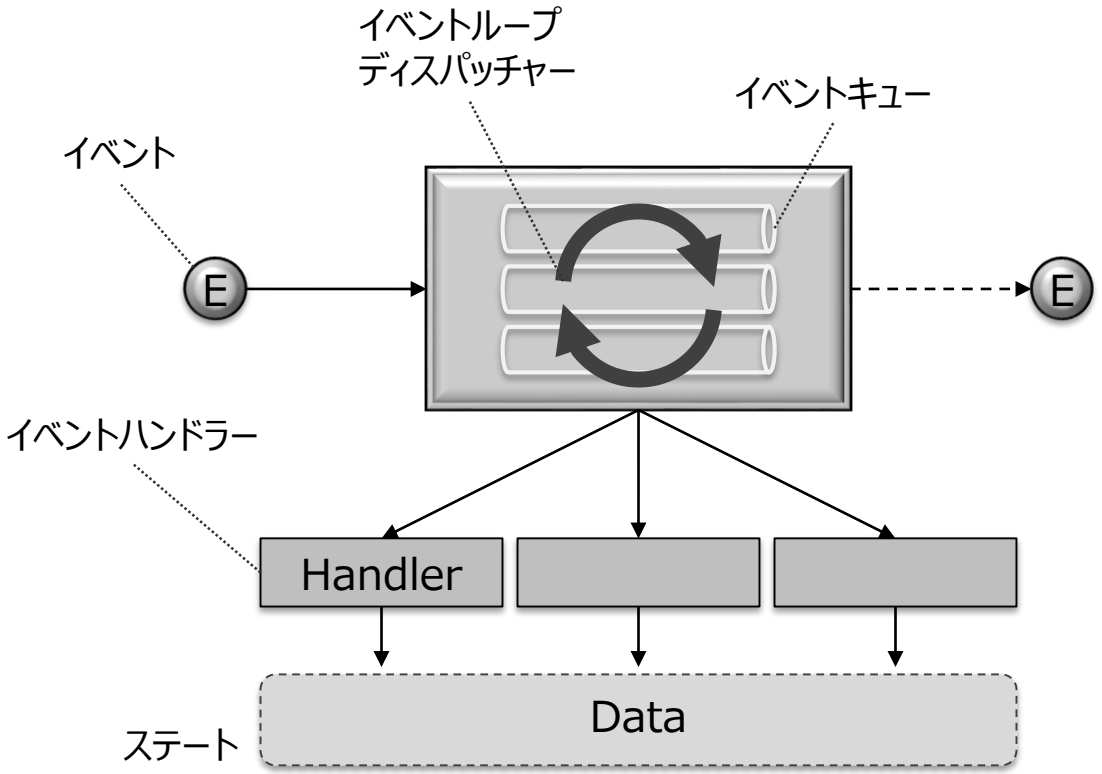
パワーを無駄なく、
イベント駆動モデル





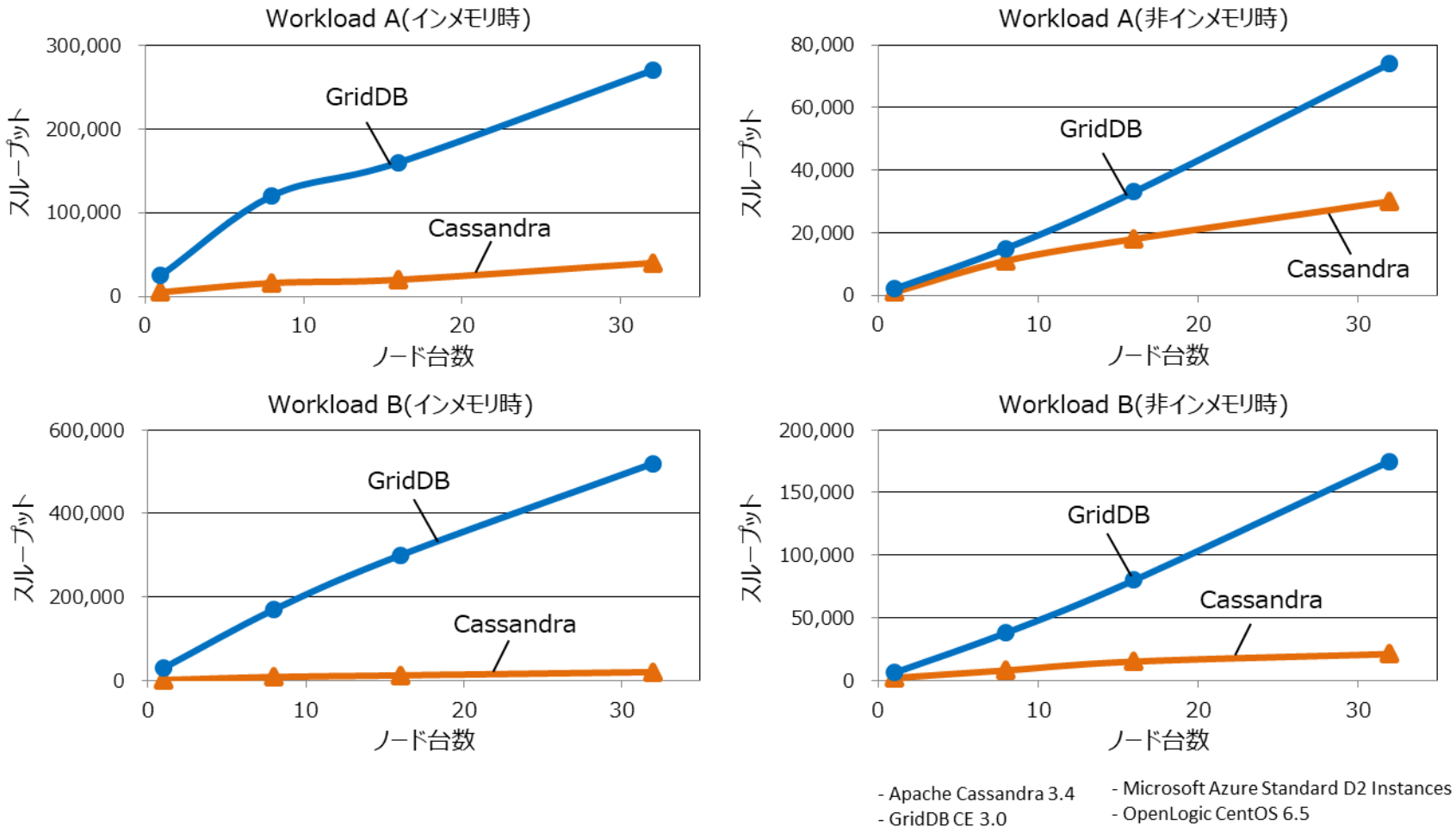
イベント駆動モデル

- イベント駆動(Event-Driven)
  - イベント＝ユーザや他システムからの要求や事象
  - そのイベントに応じて、設定された関数やサービスを起動



NoSQL性能

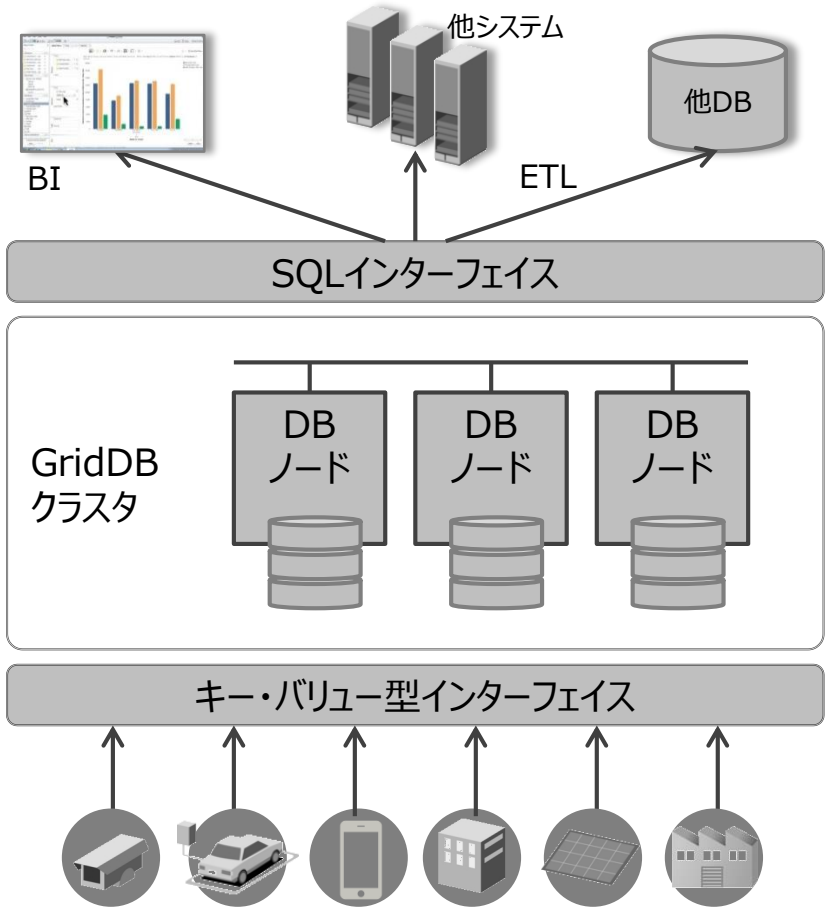
- YCSB (Yahoo! Cloud Serving Benchmark)



- Apache Cassandra 3.4
- GridDB CE 3.0

- Microsoft Azure Standard D2 Instances
- OpenLogic CentOS 6.5

NoSQLとSQLのデュアルインターフェイス



SQLインターフェイス

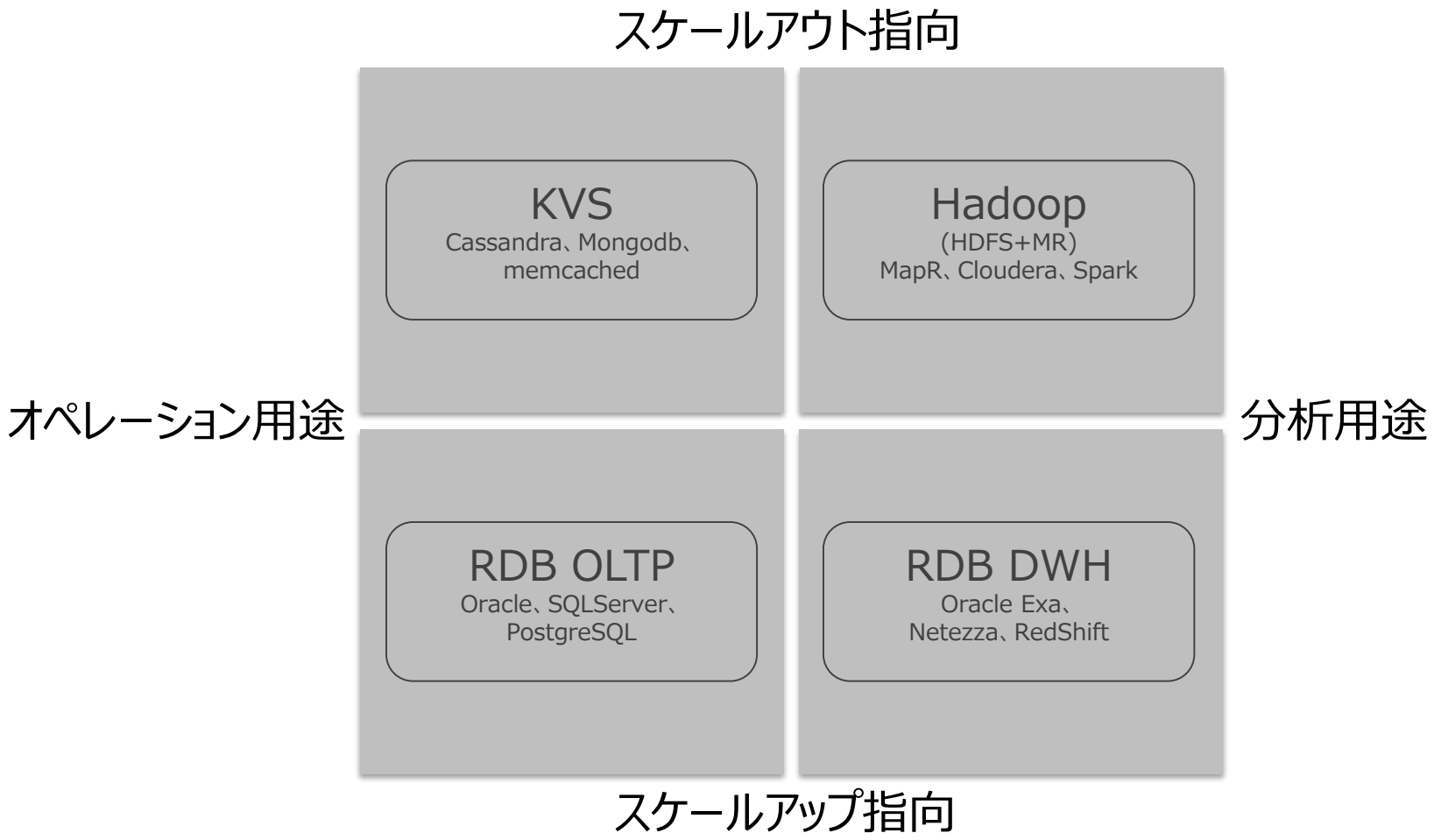
- 分散並列SQLデータベース
- 巨大コンテナに対するコンテナパーティショニング
- ジョインなど複数コンテナ(テーブル)に対するSQL
- JDBC/ODBCドライバー

NoSQL(キー・バリュー型)インターフェイス

- 高可用、高スループット指向のKVS
- キーコンテナに対するCRUD
- Java/C/Python/Rubyドライバー

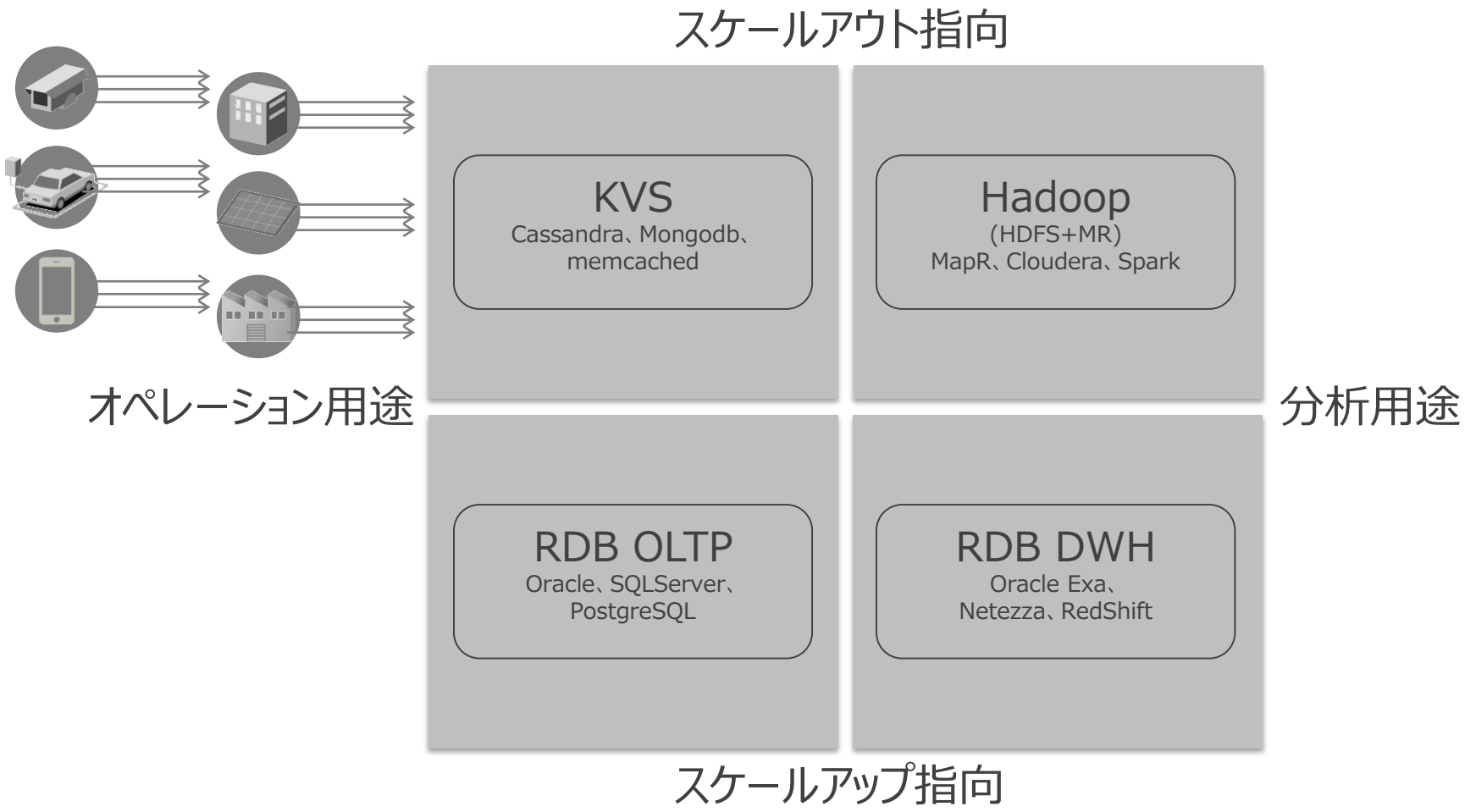
DB分類

- いつもの4象限マトリクス



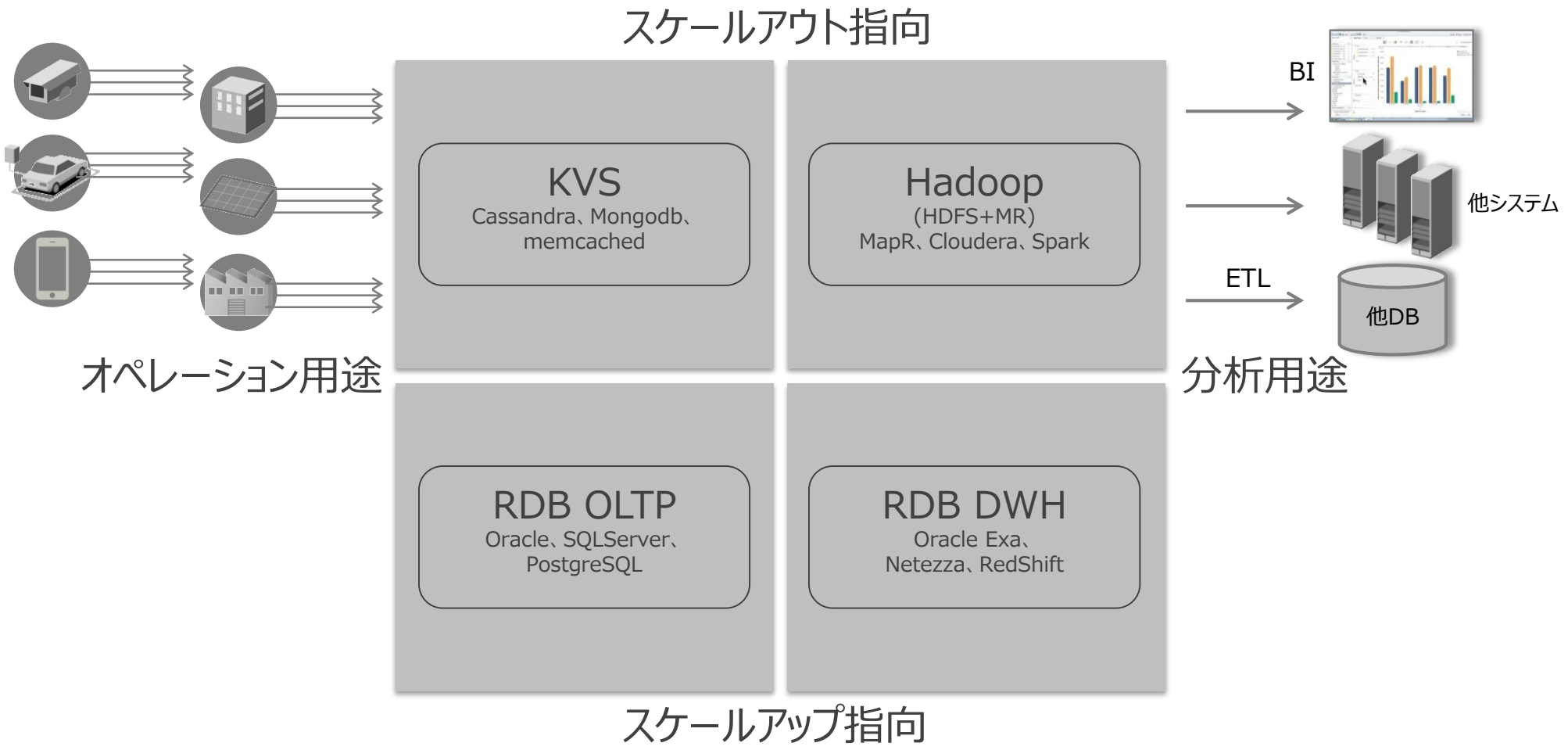
IoTでは

- 将来が見通せないので、スケールアウトするストア



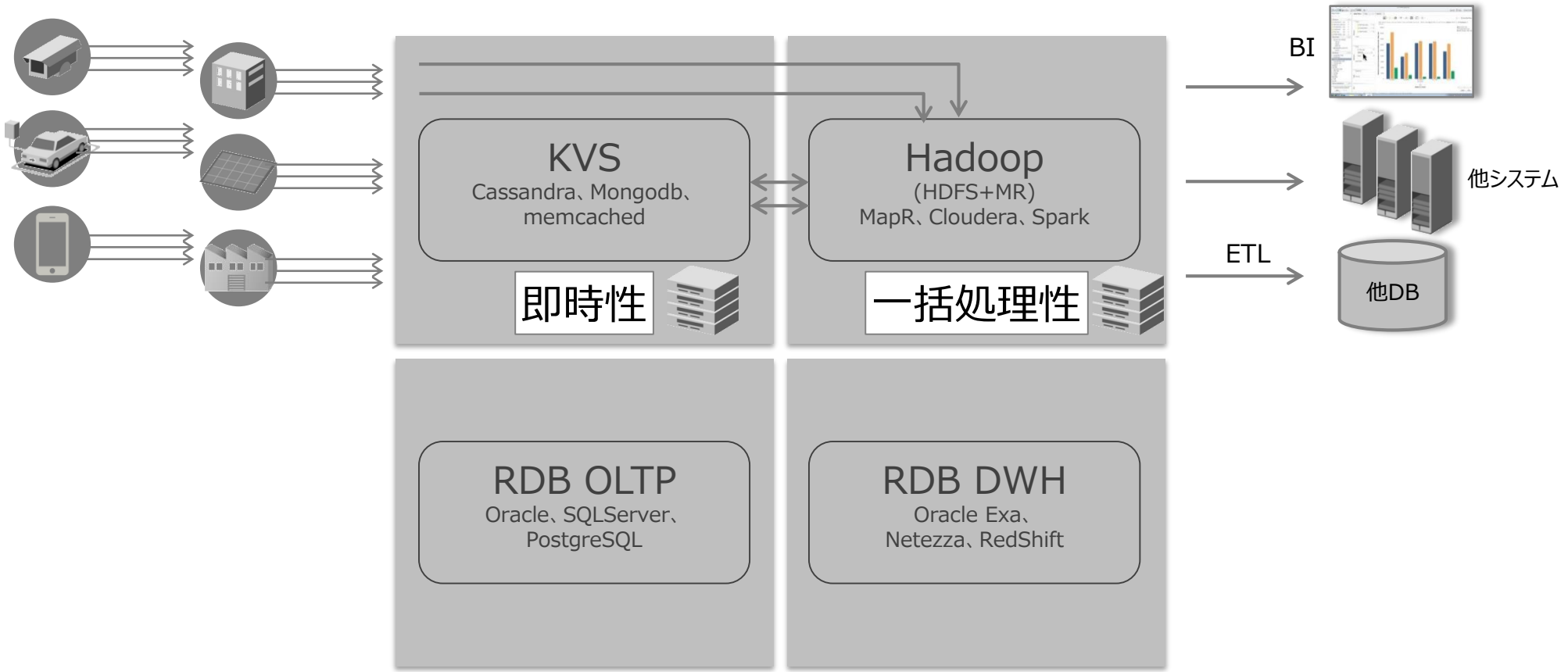
IoTでは

- 貯めていくうちに分析もしたくなる



ラムダアーキテクチャ

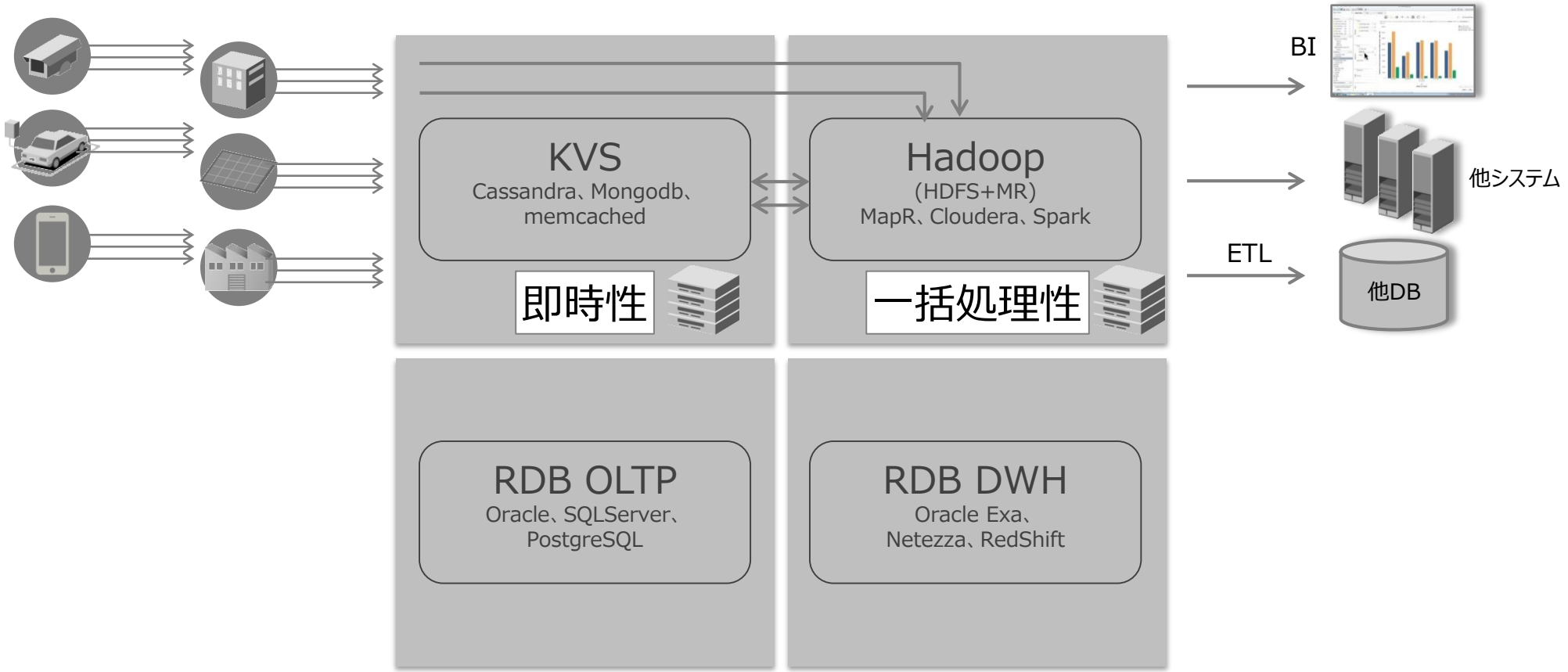
- 「ファストデータとビッグデータ」 適材適所、組み合わせの妙



http://horicky.blogspot.jp/2014/08/lambda-architecture-principles.html

ラムダアーキテクチャ

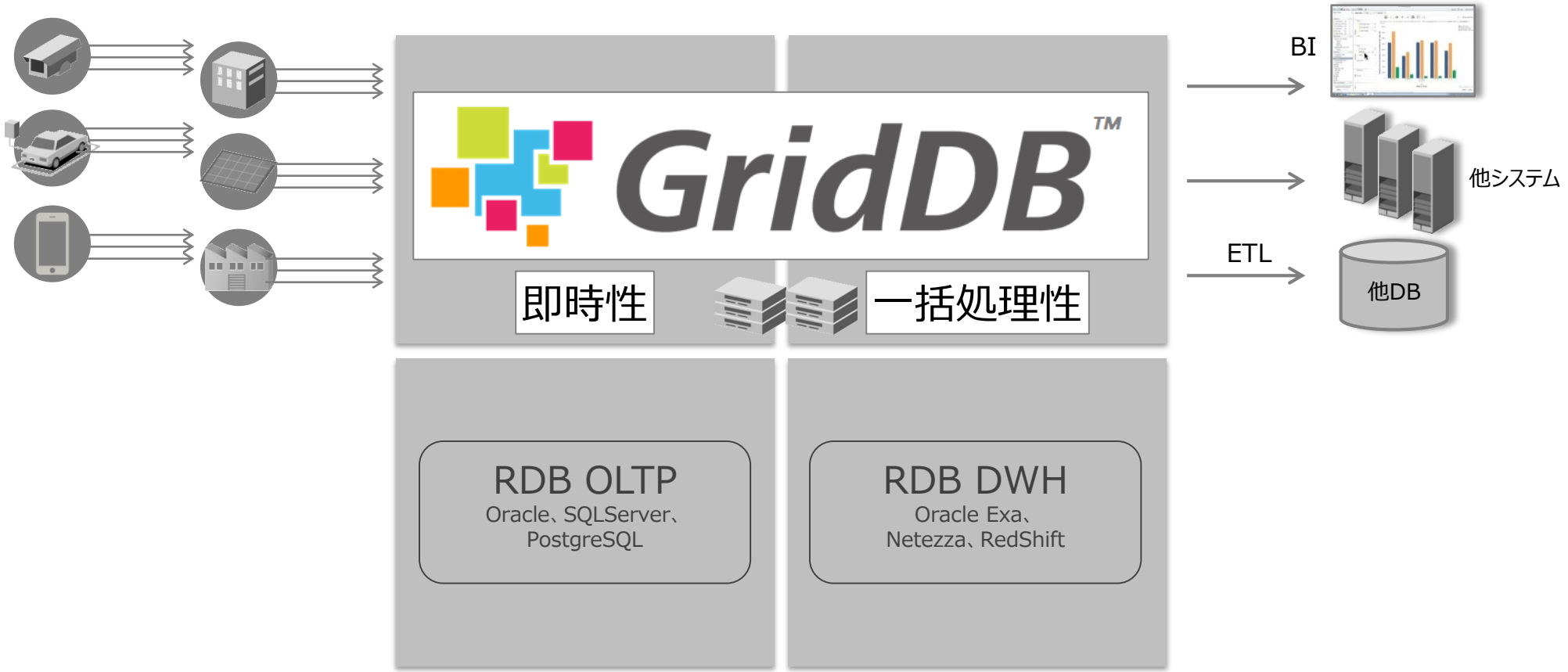
- 「ファストデータとビッグデータ」 適材適所、組み合わせの妙
- 構築コスト、運用コスト、やや古いデータが気になる…



http://horicky.blogspot.jp/2014/08/lambda-architecture-principles.html

ユーザ要求

- 条件さえ合えば、One Data、One Place。One DB
- あまり激しいことをやらなければ…

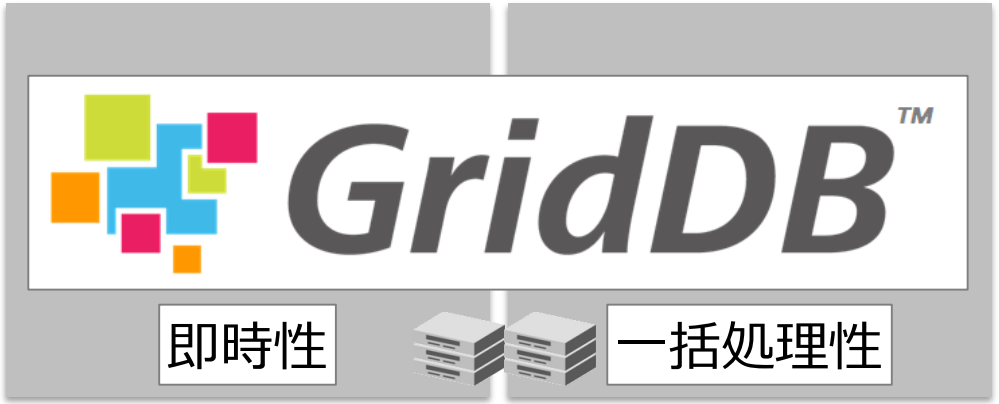


キーワード

SQL-NoSQLマッピング

分散並列処理(MPP)

NoSQL(KVS)



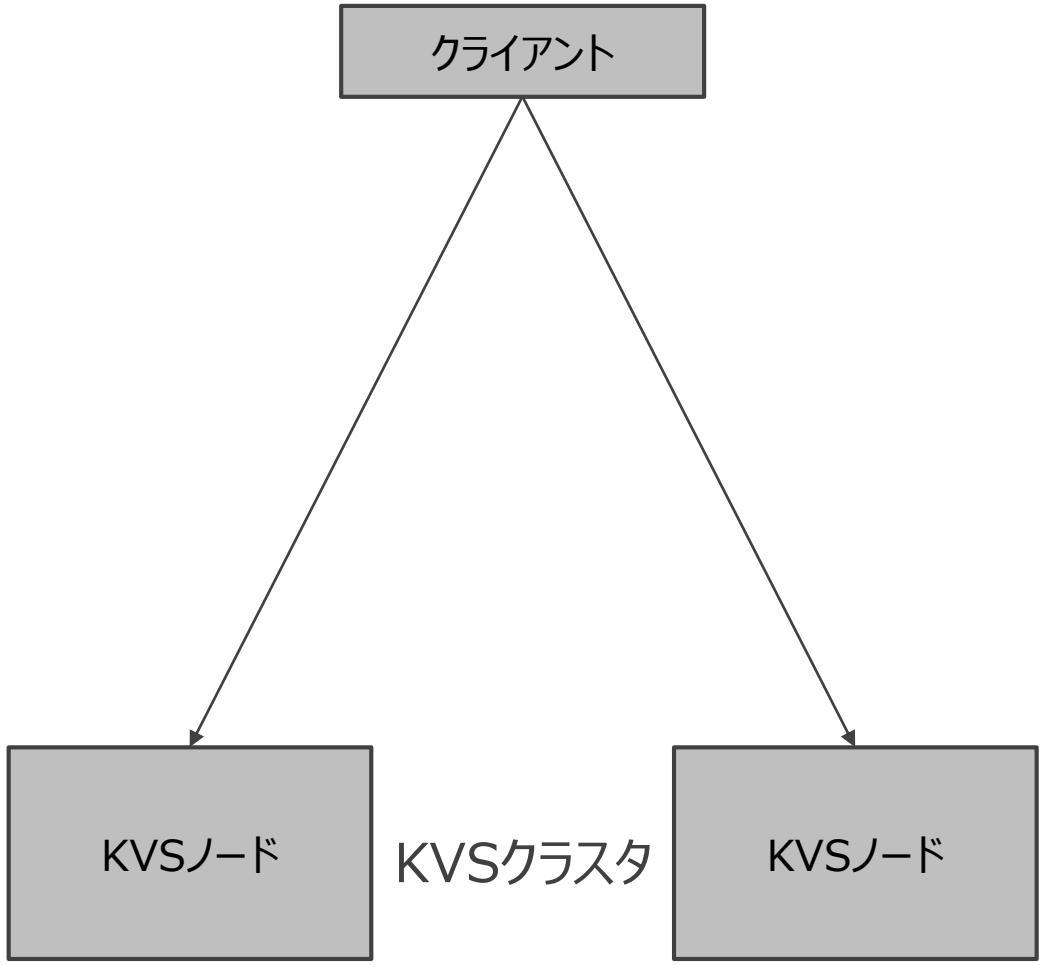
SQL

アーキテクチャ

イベント駆動モデル

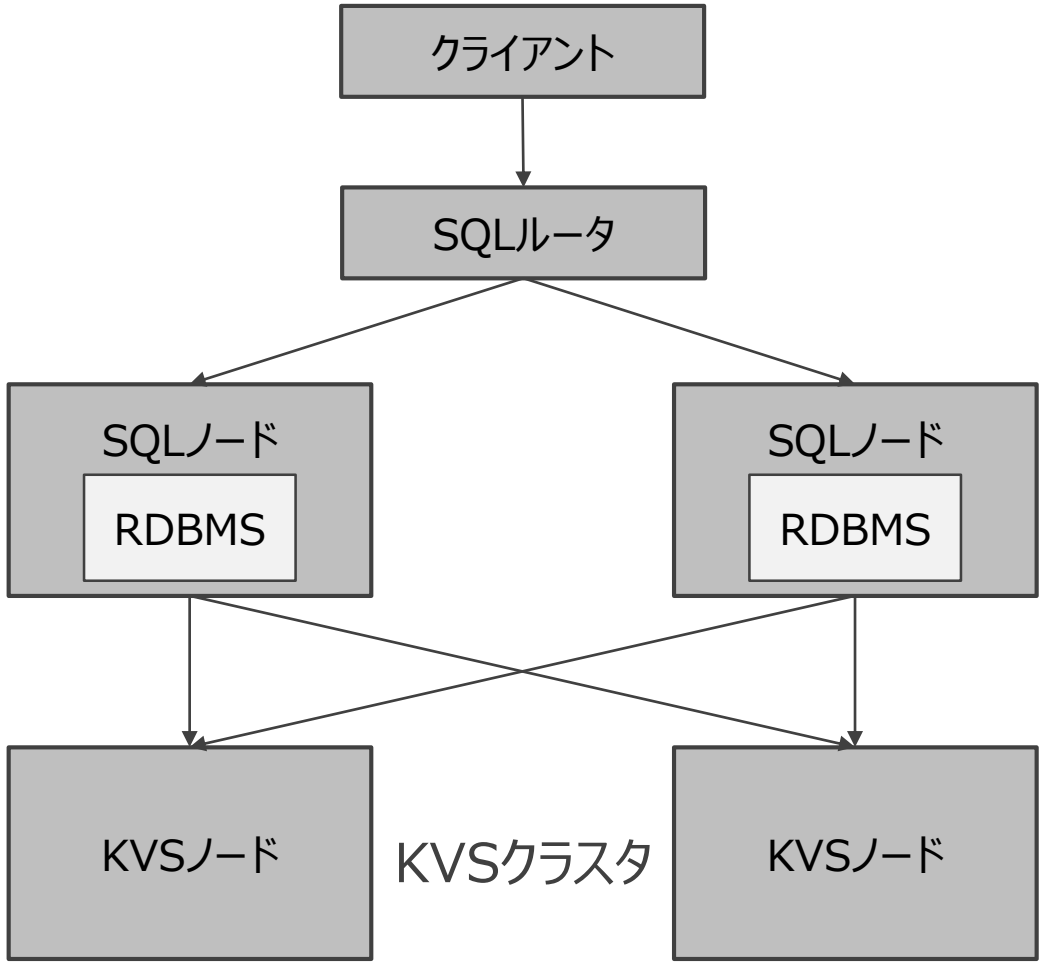
NoSQL+SQLですが…

- NoSQL(KVS)の上に…



NoSQL+SQLですが…

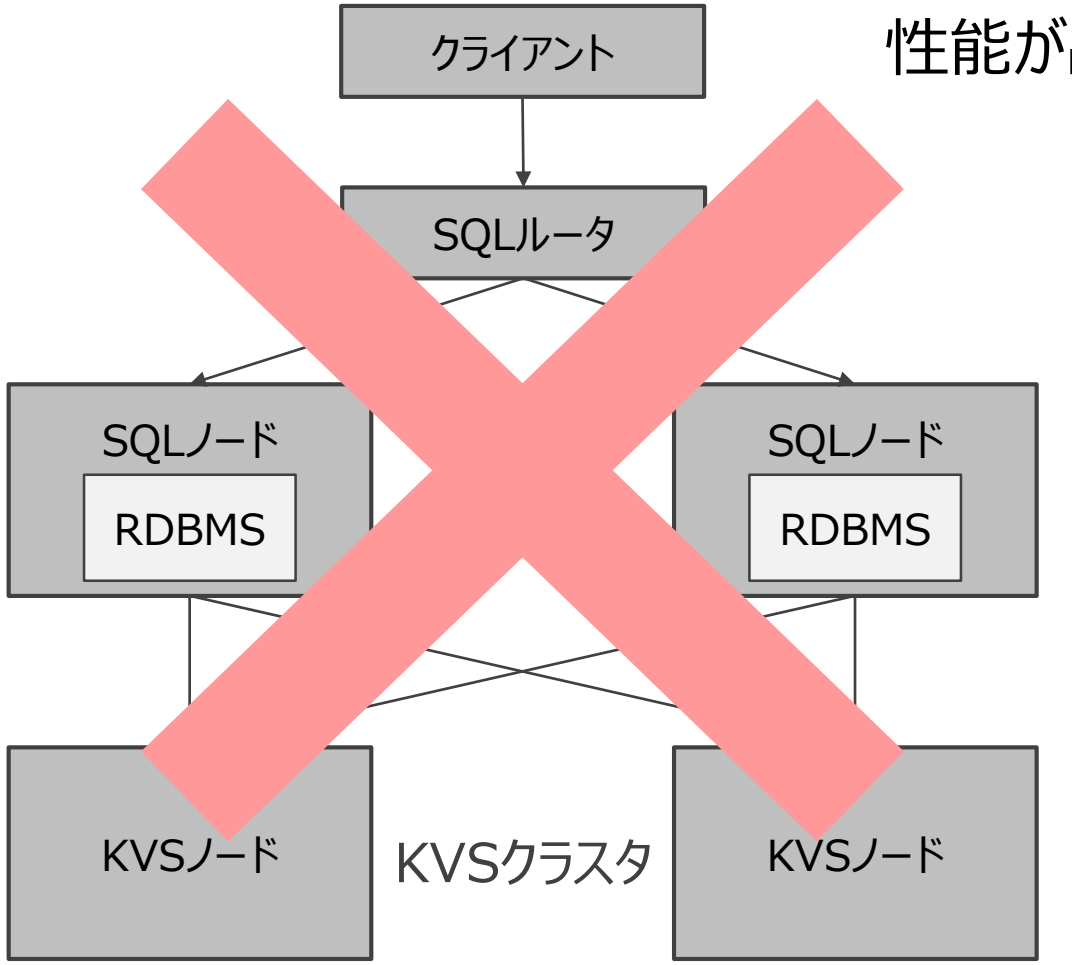
- SQLを載せる、よくある構成



NoSQL+SQLですが…

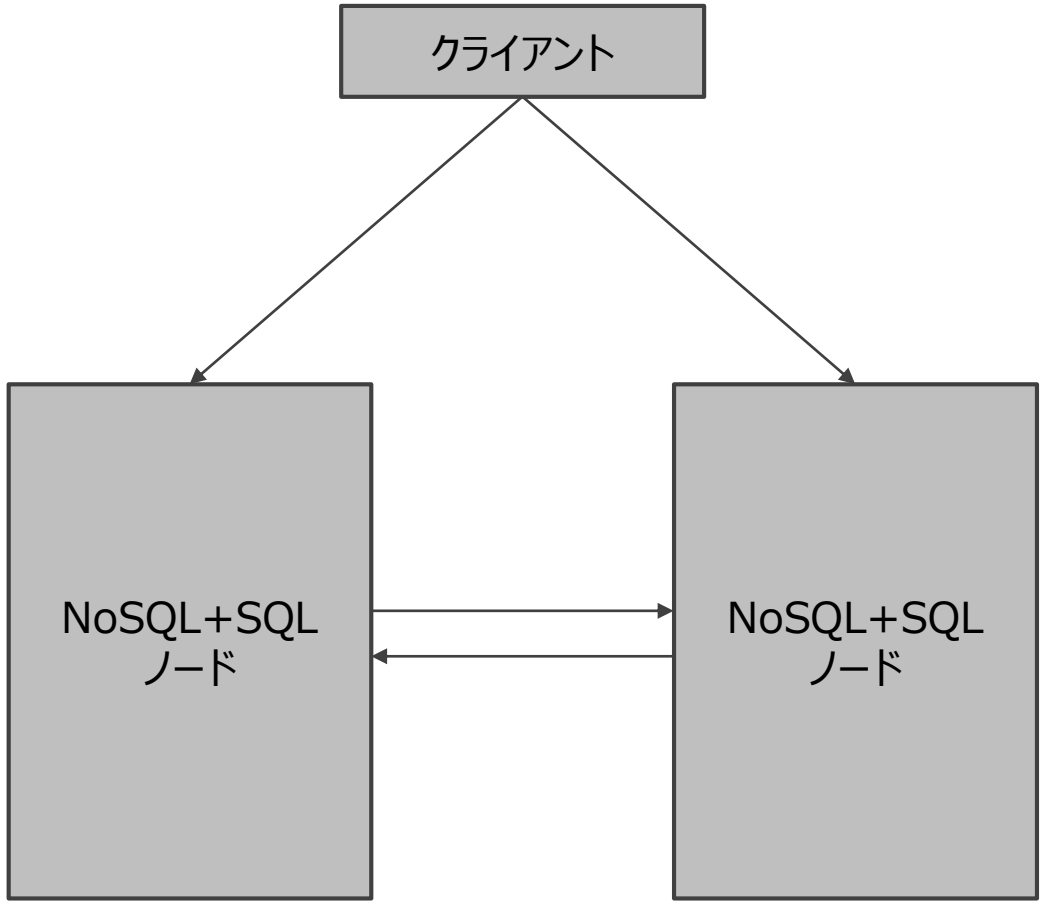
- このようなものではないです！

昔、試したこともありましたが、
性能が出ませんでした。



NoSQL+SQLですが…

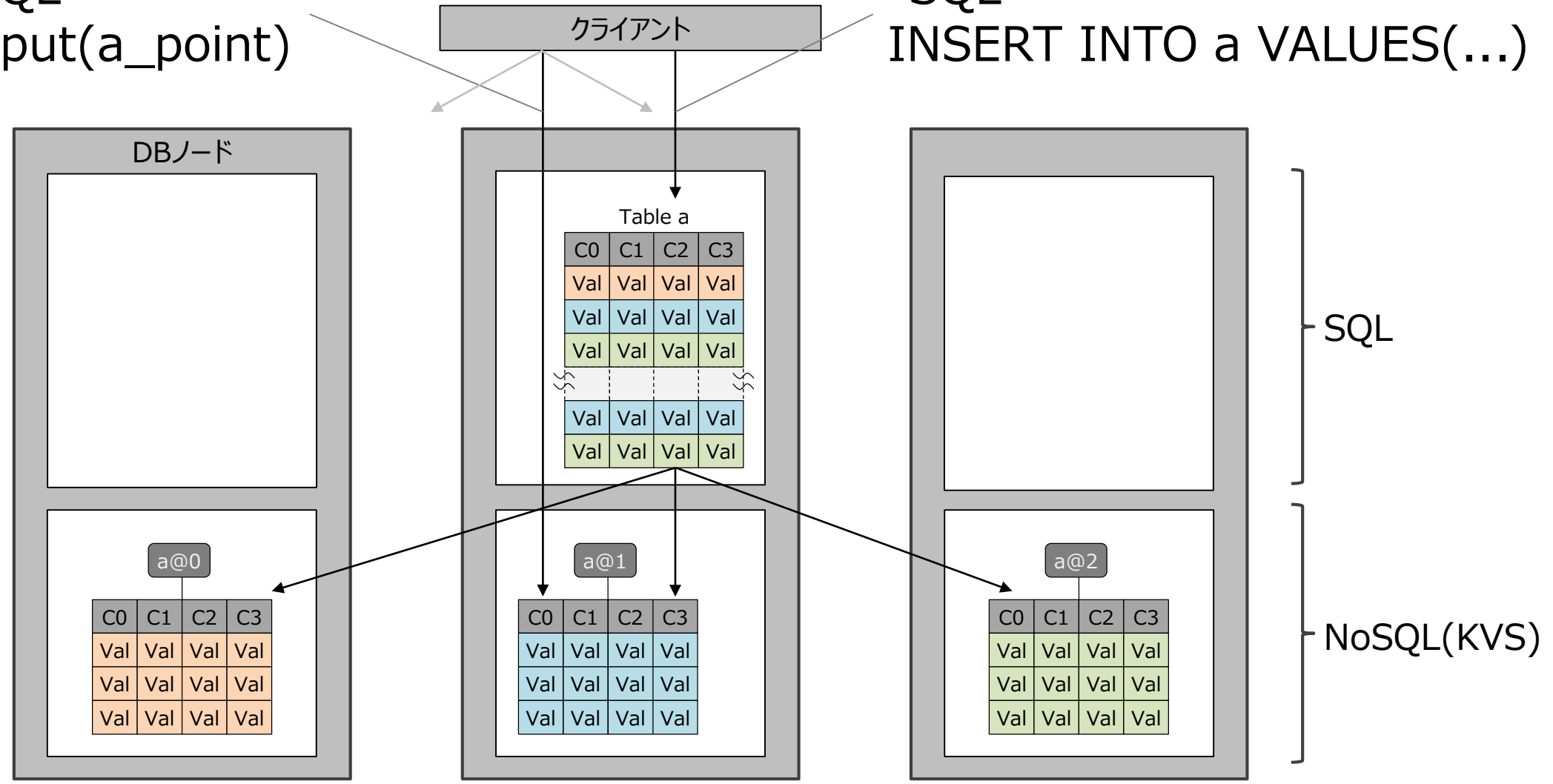
- どちらかというとこのような感じ



SQL-NoSQLのマッピング

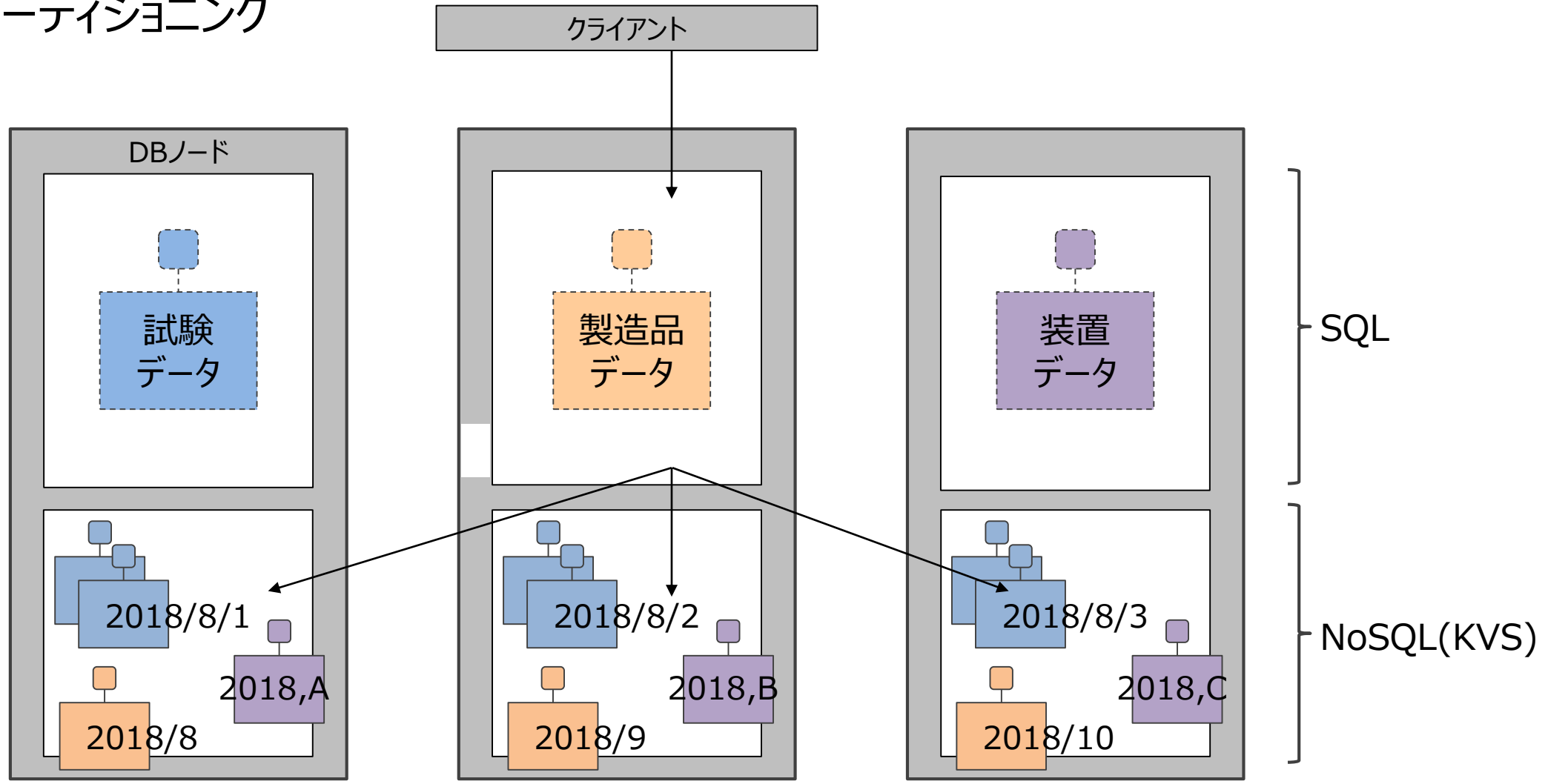
“NoSQL”
cont.put(a_point)

“SQL”
INSERT INTO a VALUES(...)



IoTデータの散らし方

- テーブルパーティショニング



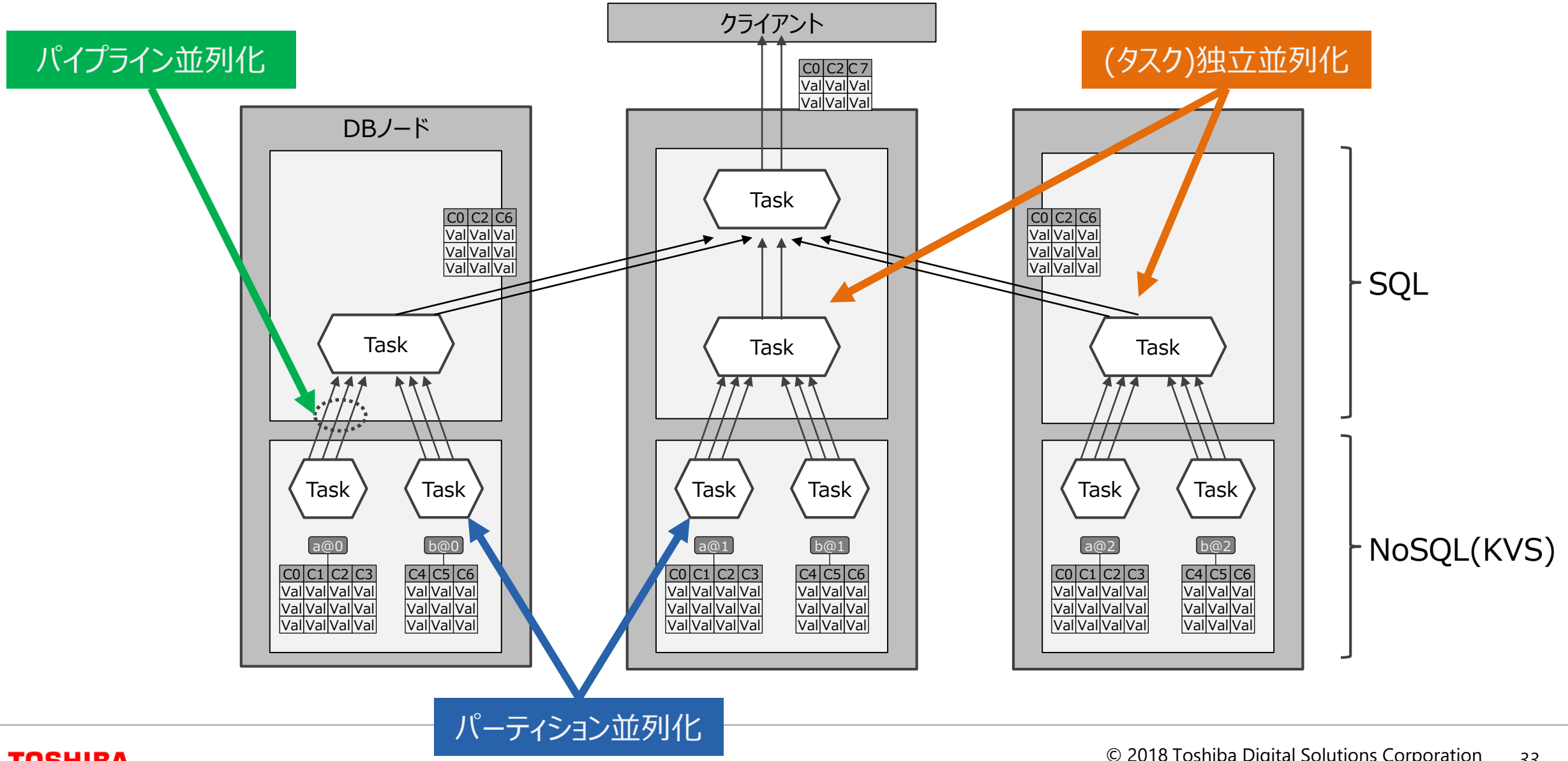
テーブルパーティショニング

```
-- インターバル
CREATE TABLE a1 (code INT, ts TIMESTAMP NOT NULL, dest STRING)
    PARTITION BY INTERVAL(ts) EVERY(1,DAY)

-- レンジ
CREATE TABLE a2 (code INT NOT NULL, ts TIMESTAMP, dest STRING)
    PARTITION BY RANGE(code) EVERY(1000)

-- ハッシュ
CREATE TABLE a3 (code INT, ts TIMESTAMP, dest STRING NOT NULL)
    PARTITION BY HASH(dest) PARTITIONS 10

-- インターバルハッシュ/レンジハッシュ
CREATE TABLE a4 (code INT NOT NULL, ts TIMESTAMP, dest STRING)
    PARTITION BY RANGE(code) EVERY(1000)
    SUBPARTITION BY HASH(dest) SUBPARTITIONS 2
```

テーブルパーティショニングのPros. / Cons.

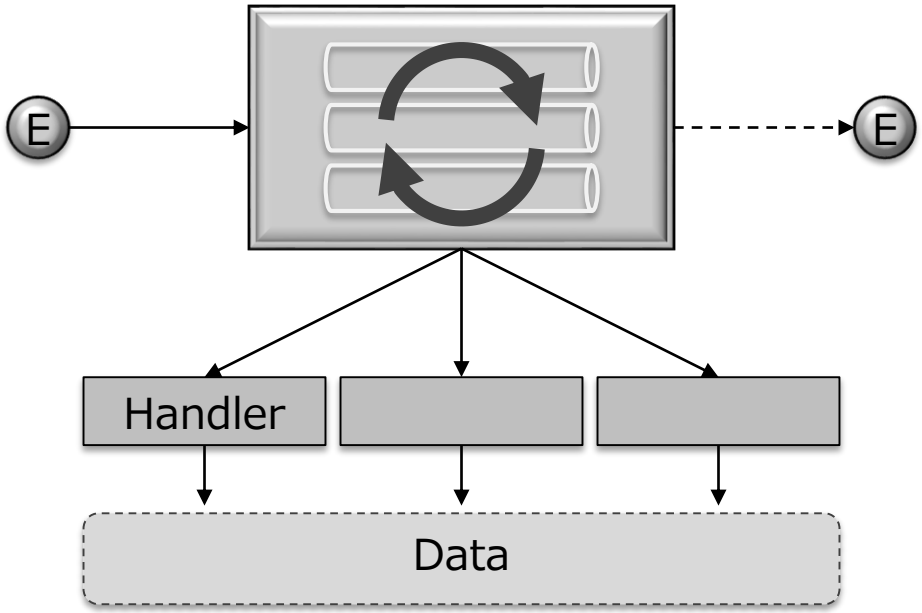
- Pros.
  - 分割されたテーブルを並列処理。大規模なデータかつ並列化しやすいSQLでは効果大。
  - 分割によるメモリアクセスが局所化する場合はI/O量削減。ランダムにアクセスするインデックス
- Cons.
  - 分割されたテーブルをまとめる処理は低速化。少量テーブルに対するJoinやScanなど
  - 分割されたテーブル間でコミットできない
- ベストケースとして
  - 分割されたテーブル内で処理が閉じている。
  - 分割されたテーブルおよびインデックスがうまくメモリに乗っている
- 各タイプの選択基準
  - ハッシュパーティショニング…散らすべきキーにランダム性が高く、キーの間に処理上の関連性が無い場合
  - インターバルパーティショニング…散らすべきキーの数値的な範囲で散らしたい場合
  - インターバルハッシュパーティショニング…インターバルパーティショニングでは力不足の場合

SQLにおける分散並列処理

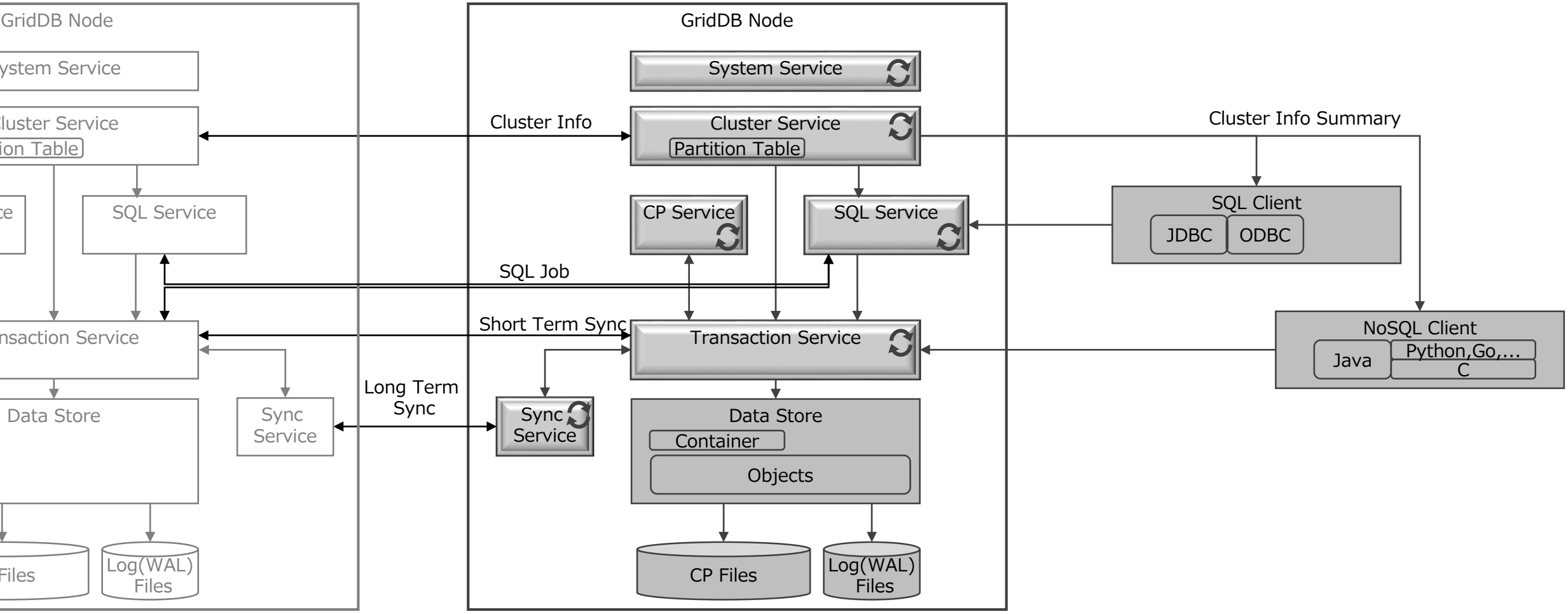


(再び)イベント駆動モデル

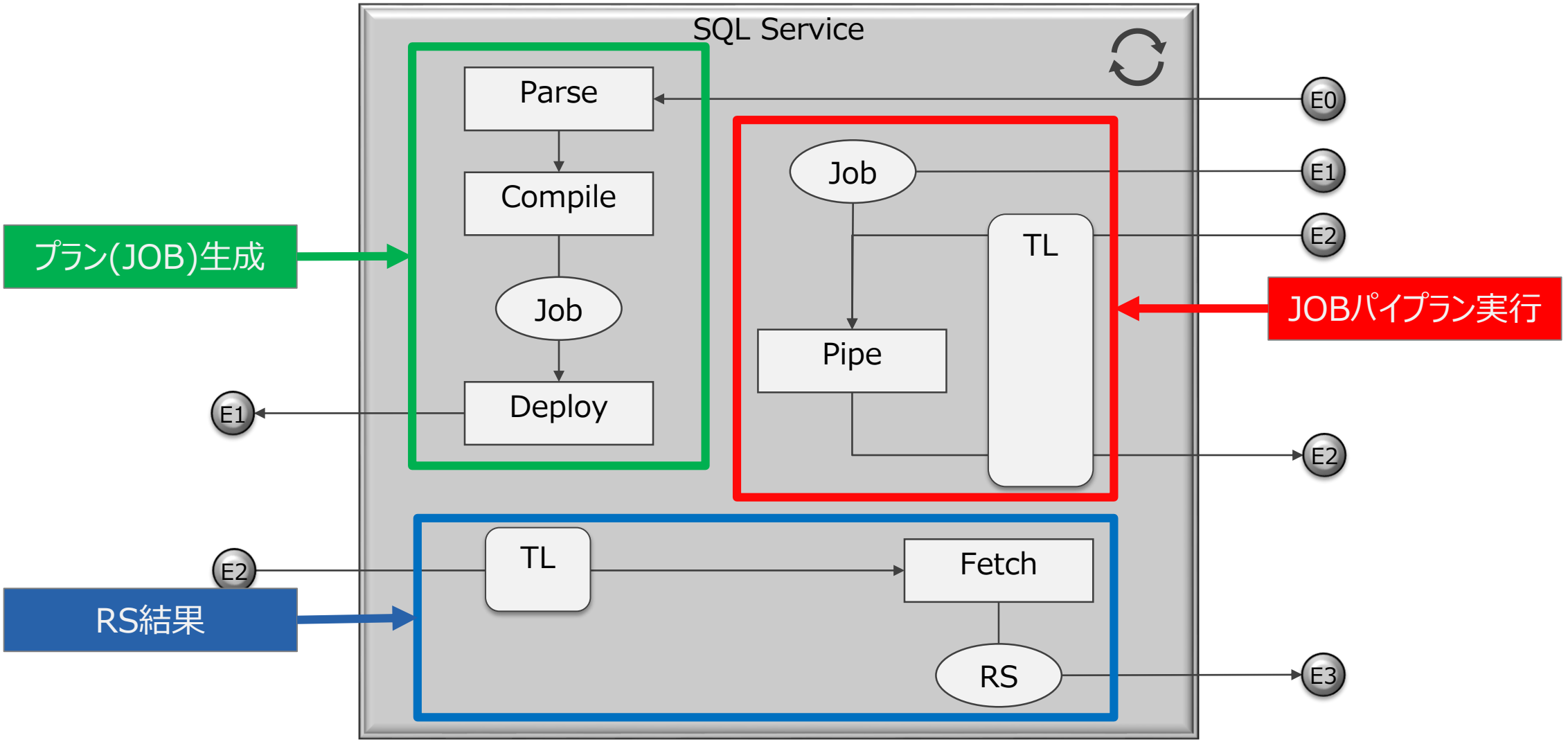
- 分散並列処理のベースとなっているのがイベント駆動モデル
- GridDBではサービスと呼ぶ



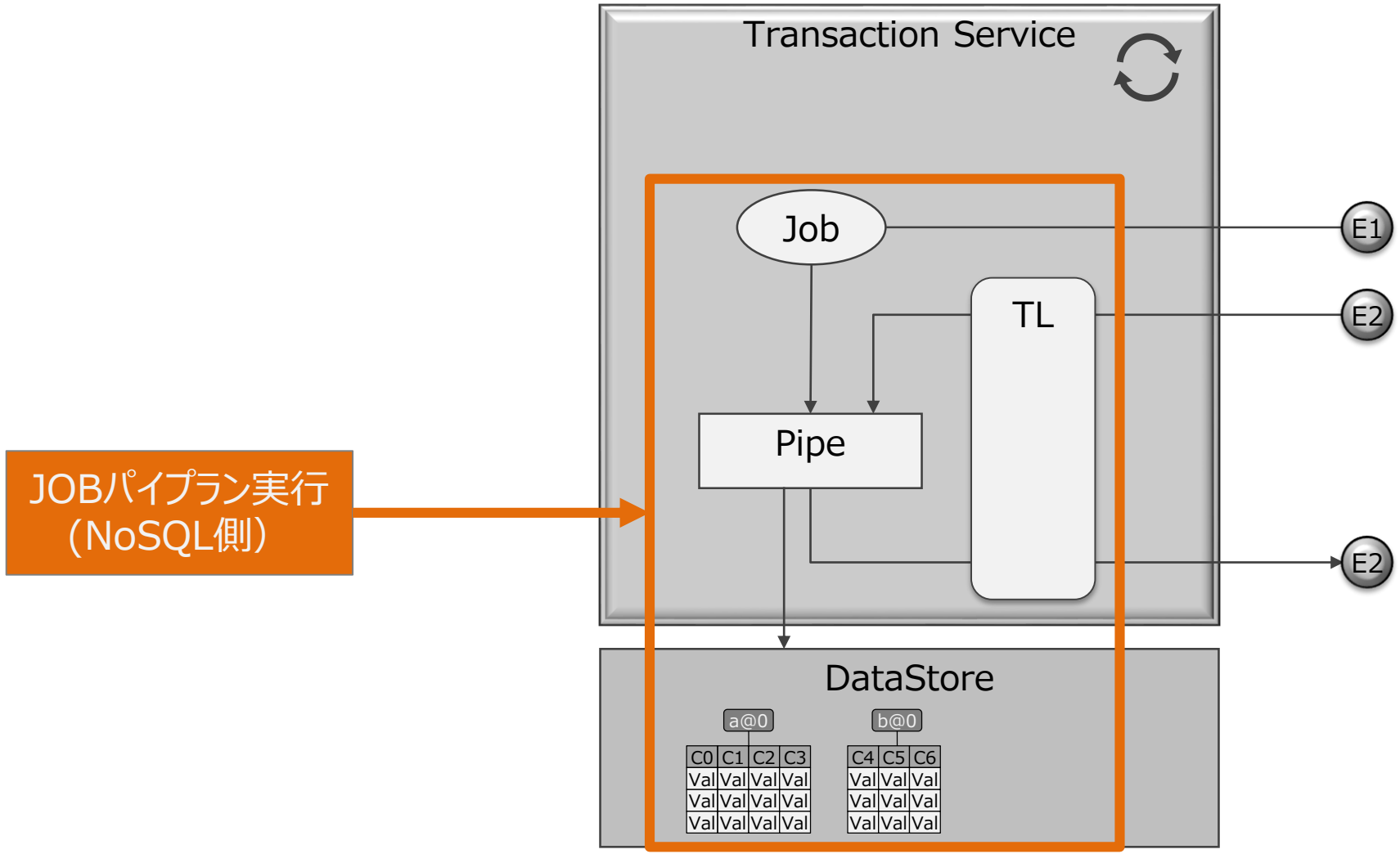
Logical Architecture of GridDB



Internal Structure of SQL Service



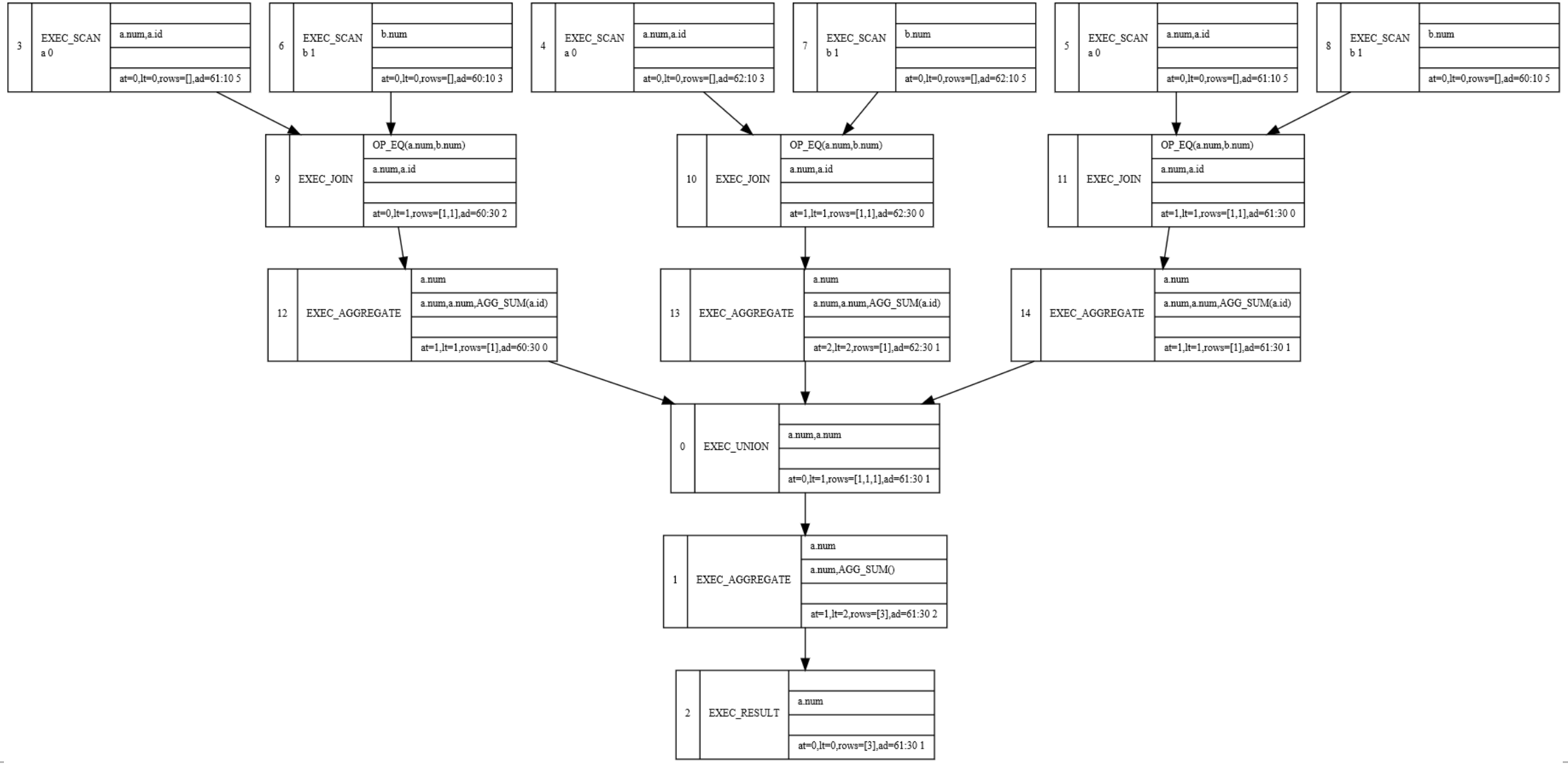
Internal Structure of Transaction Service



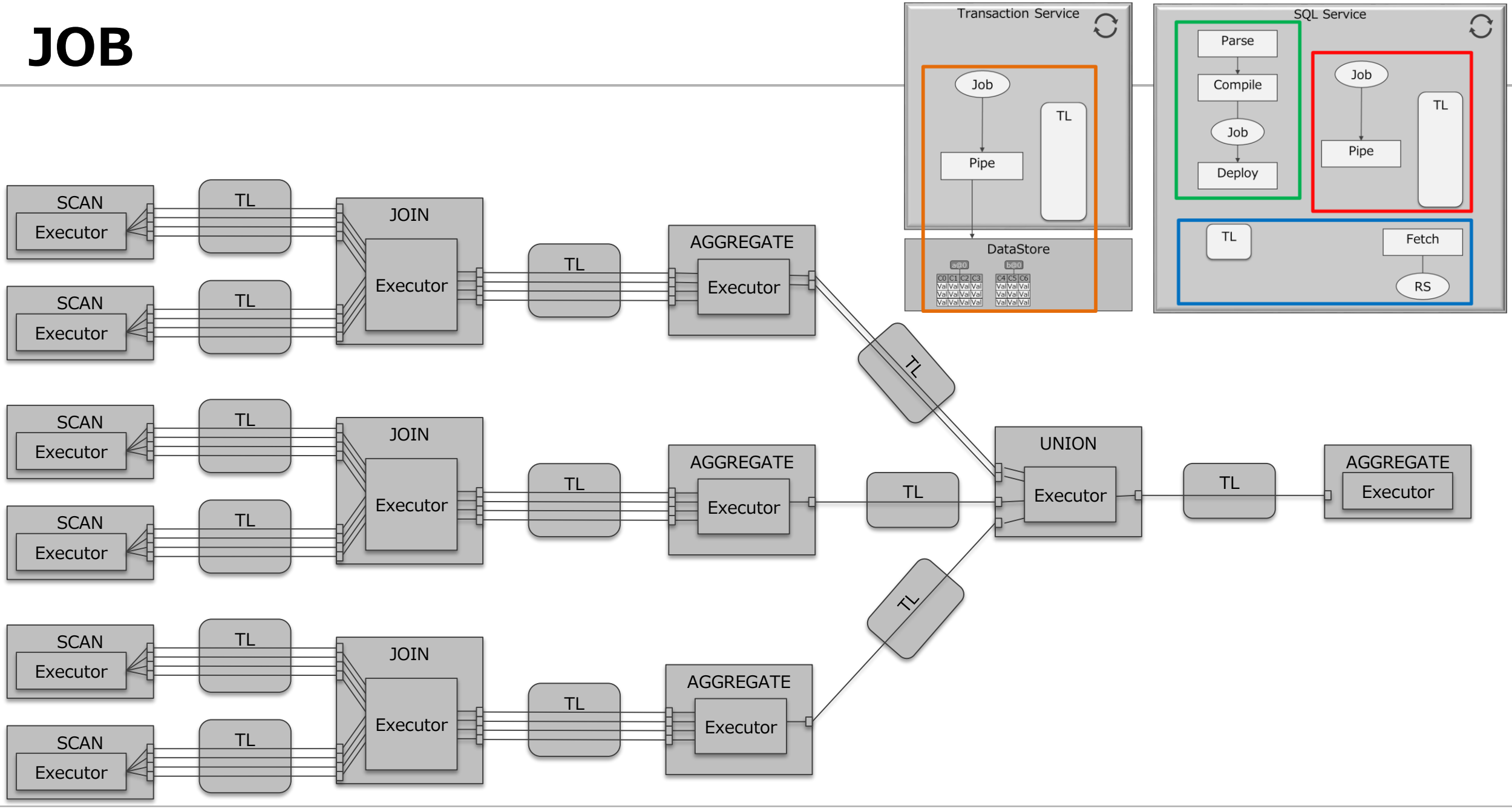
サンプルSQL

```
CREATE TABLE A (num INT NOT NULL, id INT, ...)
PARTITION BY HASH(num) PARTITIONS 3

SELECT A.num, SUM(A.id)
FROM A, B
WHERE A.num = B.num
GROUP BY A.num
```

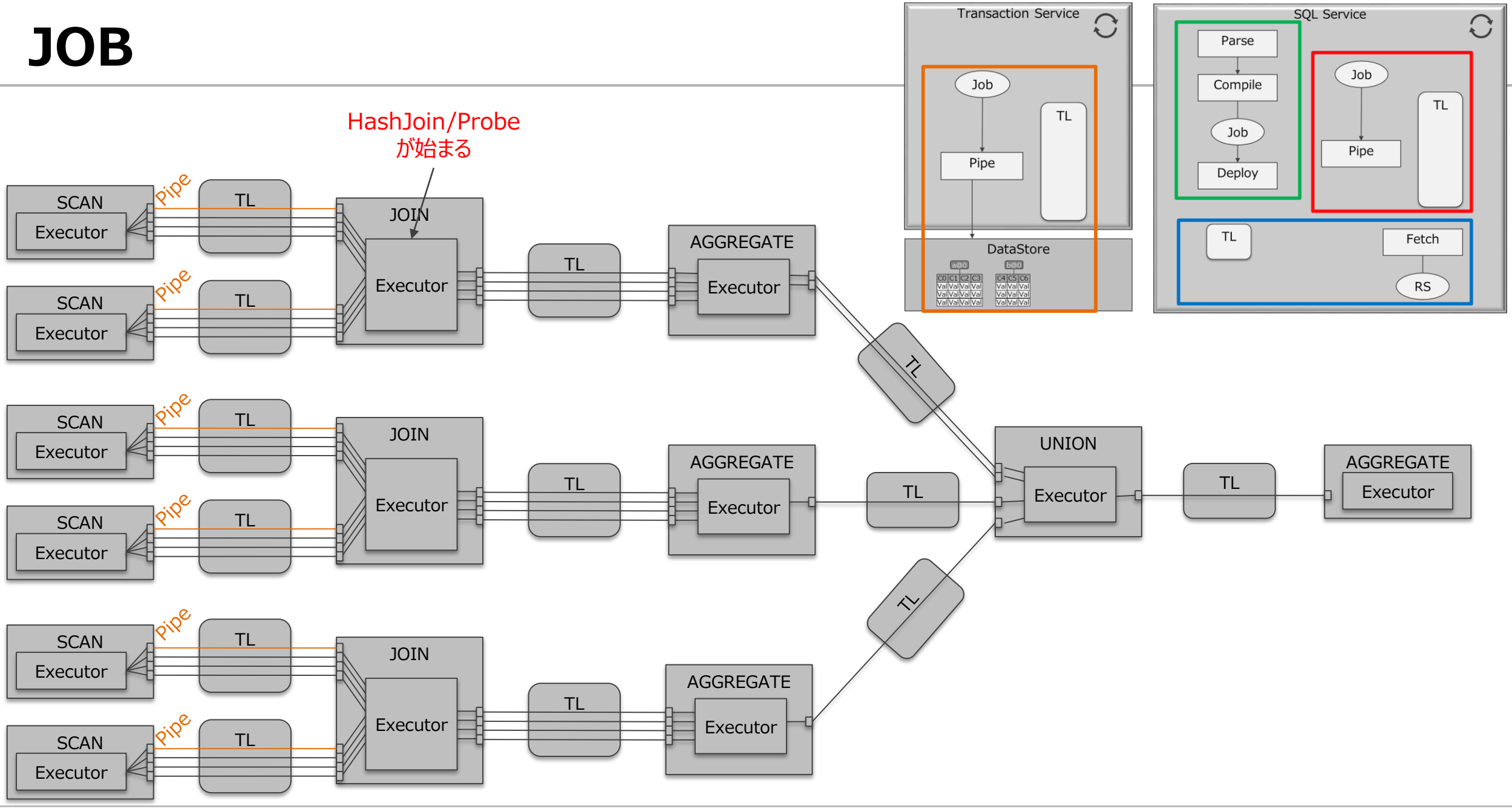
JOB(EXPLAIN ANALYZE)



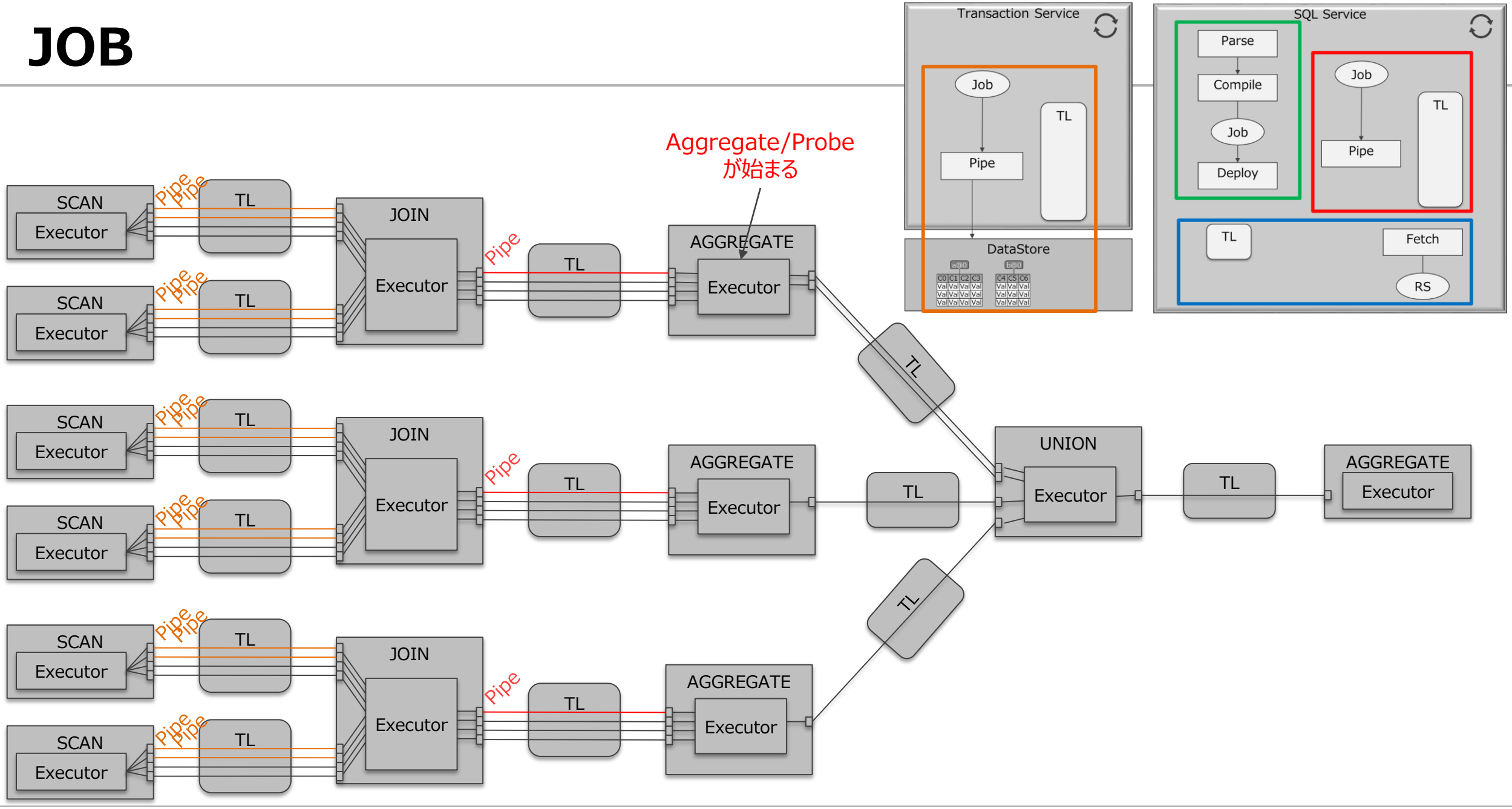
JOB



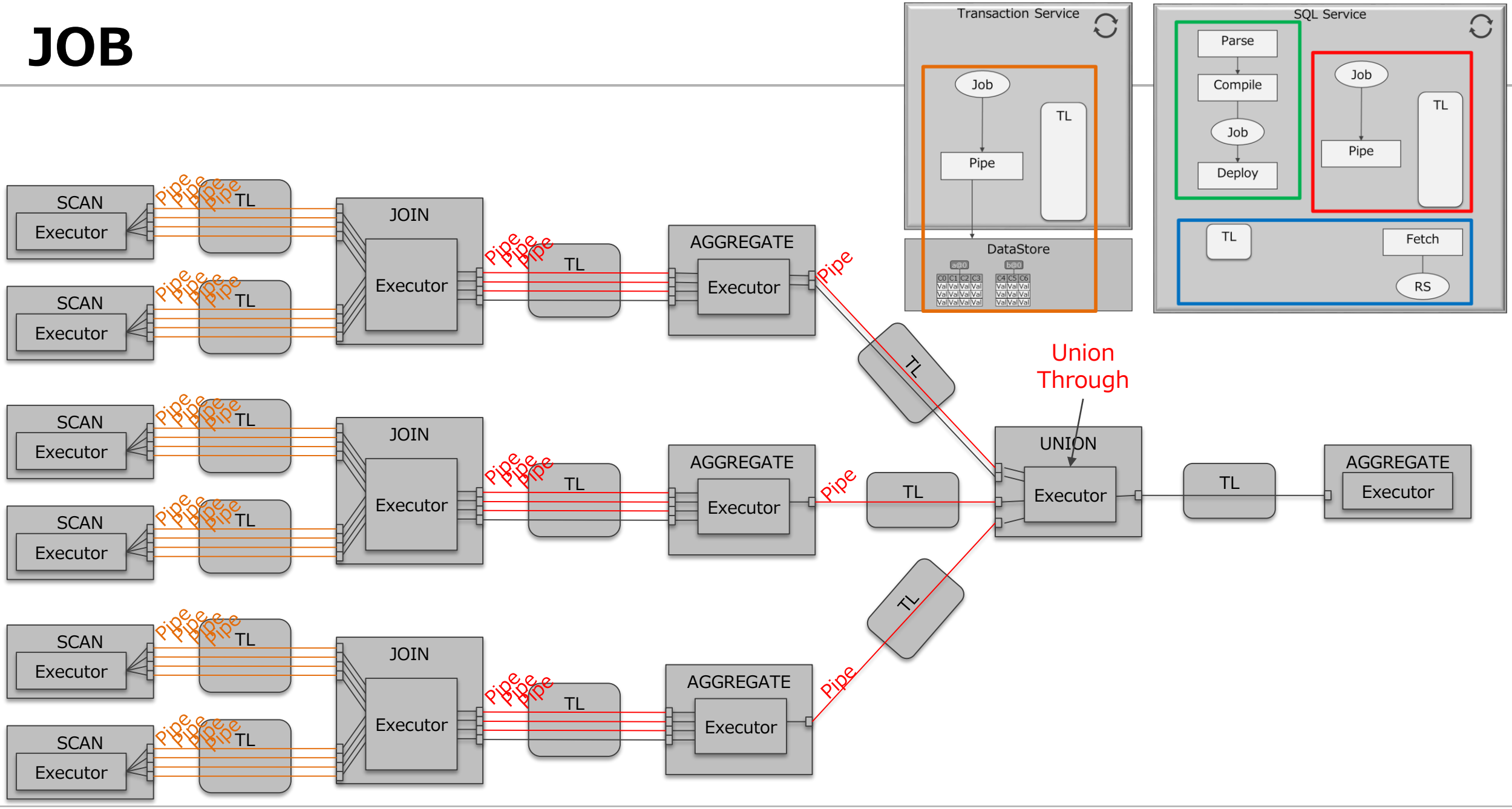
JOB



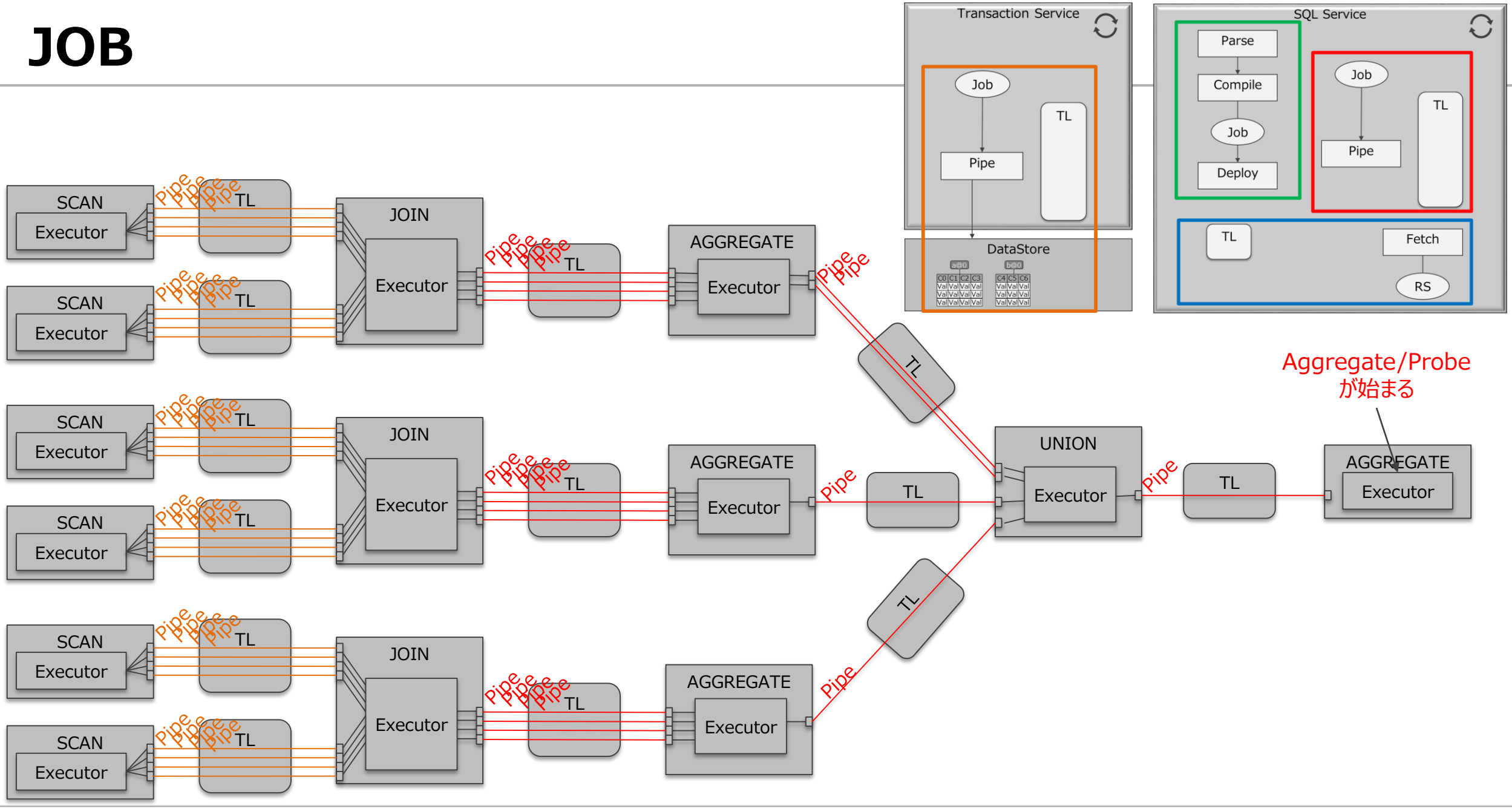
JOB



JOB

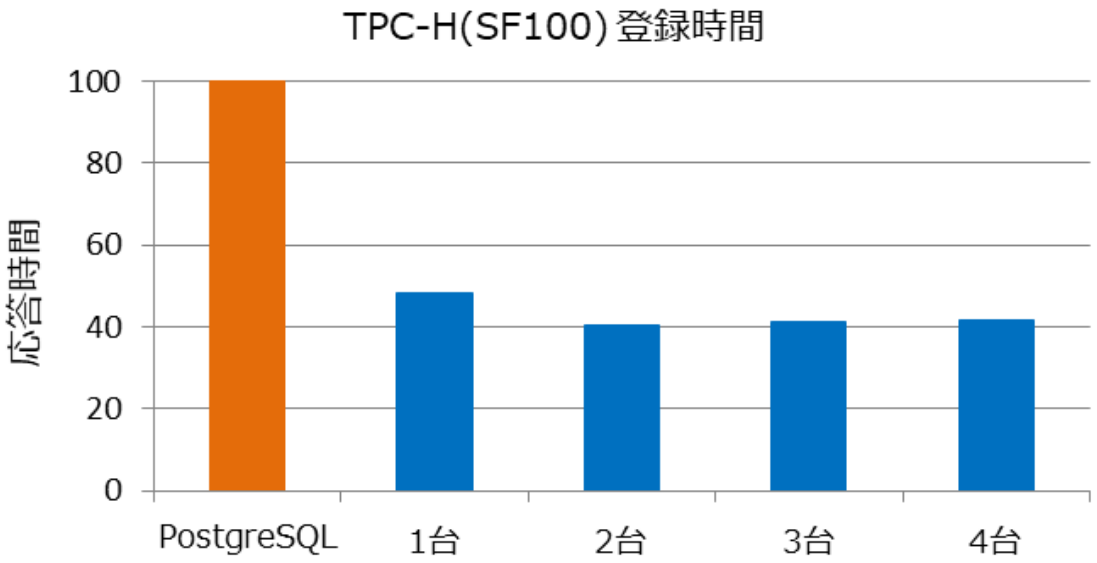


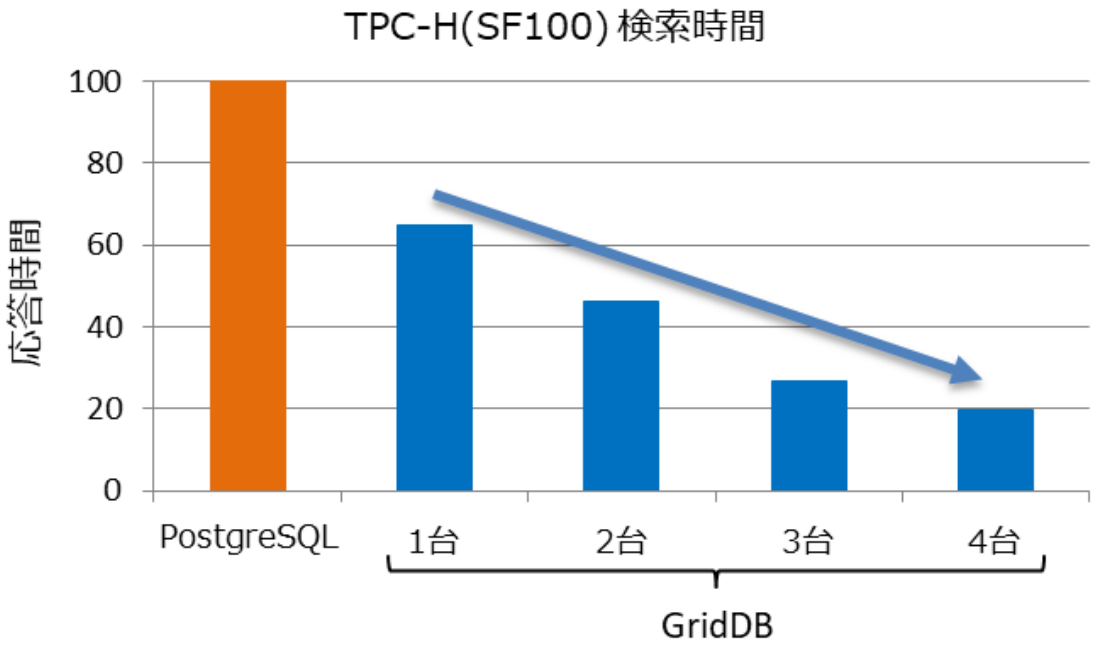
JOB



SQL性能

- TPC-H(Transaction Processing Performance Council)
- SQLのスケールアウト効果





- PostgreSQL 9.6
- GridDB AE 4.0
- CPU : 8-core Intel® Xeon® E5-2620 v4 2.10GHz
- Memory : 64GB
- HDD : SAS 12TB
- OS : CentOS 7 with kernel 3.10.0-514.el7.x86_64
- Network : 1Gb Ethernet
- Dataset : TPC-H(SF 100), Q1-Q8

適用事例

- 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に適用中

・フランス リヨン 太陽光発電 監視・診断システム
　　発電量の遠隔監視、発電パネルの性能劣化を診断
・クラウドBEMS
　　ビルに設置された各種メータの情報の収集、蓄積、分析
・石巻スマートコミュニティ プロジェクト
　　地域全体のエネルギーのメータ情報の収集、蓄積、分析
・電力会社 低圧託送業務システム
　　スマートメータから収集される電力使用量を集計し、需要量と発電量のバランスを調整
・神戸製鋼所 産業用コンプレッサ稼働監視システム
　　グローバルに販売した産業用コンプレッサをクラウドを利用して稼働監視
・東芝機械 IoTプラットフォーム
　　工作機器、射出成形、ダイカストマシン、など膨大な製造データを管理
・DENSO International Americaの次世代の車両管理システム
　　https://griddb.net/ja/blog/griddb-automotive/
・クラウド型IoTソリューション
….

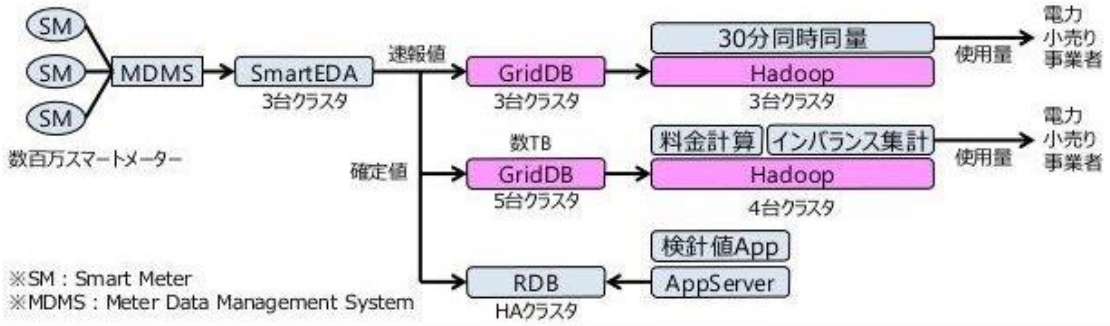
- 東芝IoTアーキテクチャー「SPINEX」の構成ソリューション

適用事例

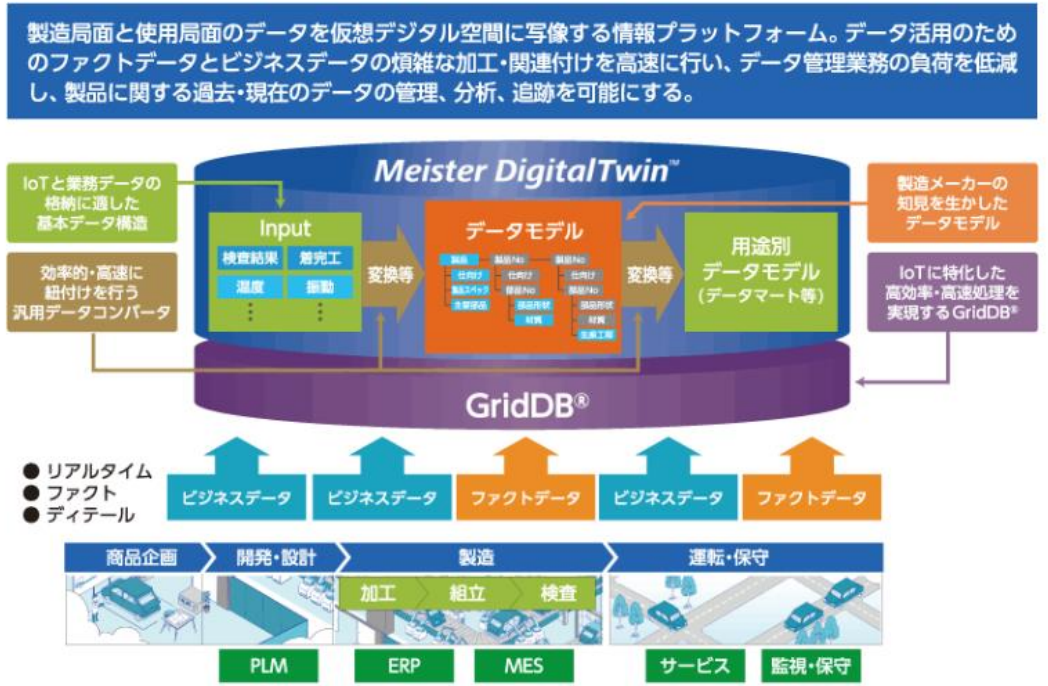
低圧託送業務

- 電力の小売全面自由化スマートメーター数の増加など要件の変化
- RDBを使った従来システムと比較して、数十倍のパフォーマンス向上と高い信頼性が必要

- **数百万台のスマートメータから30分おきに送られてくるメータデータ3ヶ月分をGridDBに蓄積（データサイズ：数百億レコード、数TB）**
- **HadoopのMapReduceを使って使用量を計算**
- **2016年4月の運用開始以来、安定稼働**



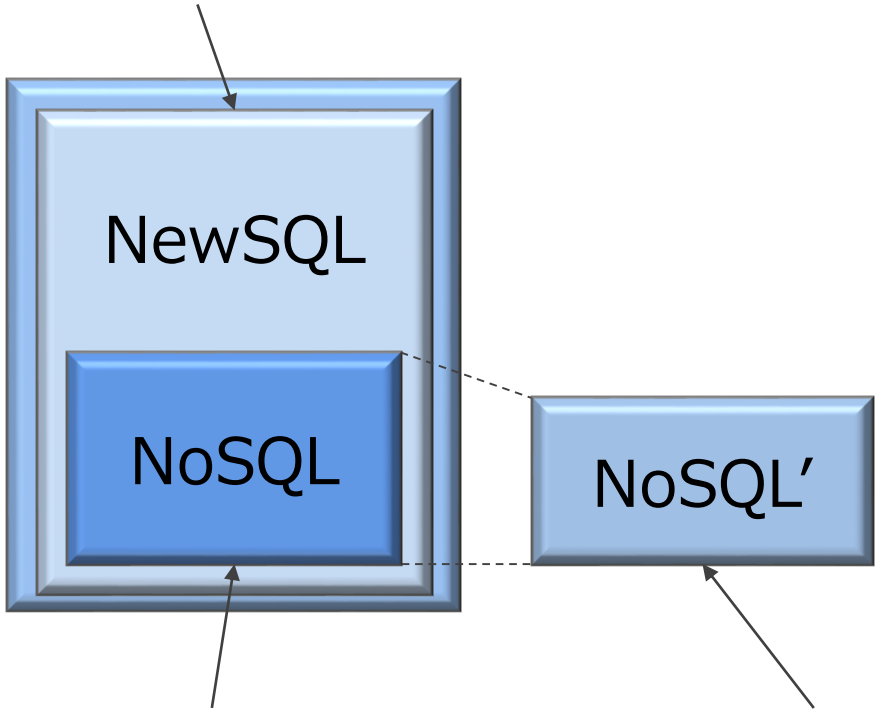
ものづくり情報プラットフォーム Meister DigitalTwin



日経BP、”明日”をつむぐテクノロジー、
「製造と使用の両面でモノを捉え最適化と価値創造を目指す」
http://special.nikkeibp.co.jp/atclh/tomorrowtech/factory_iot/

製品ラインアップ

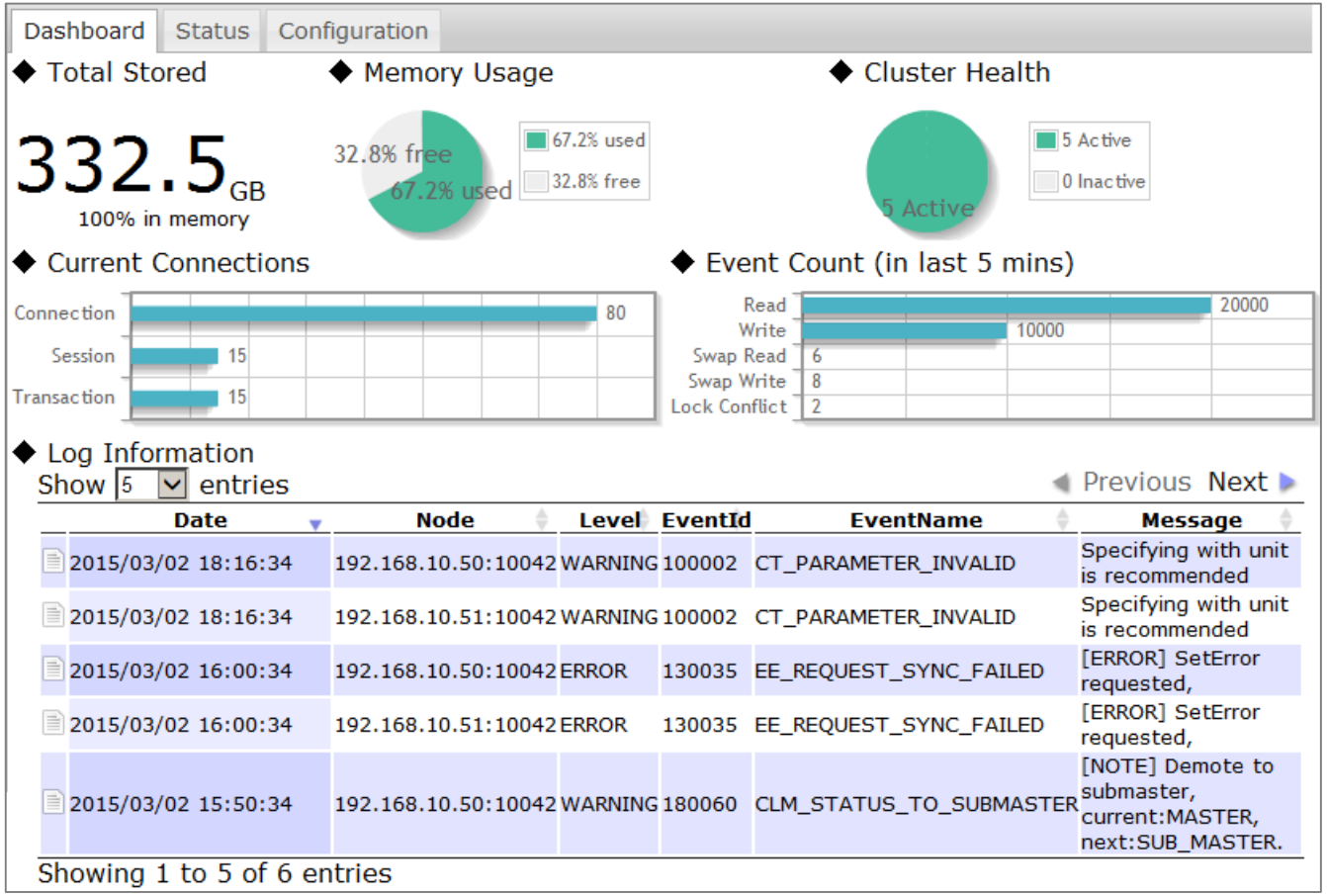
②GridDB AE (Advanced Edition)



①GridDB SE (Standard Edition)

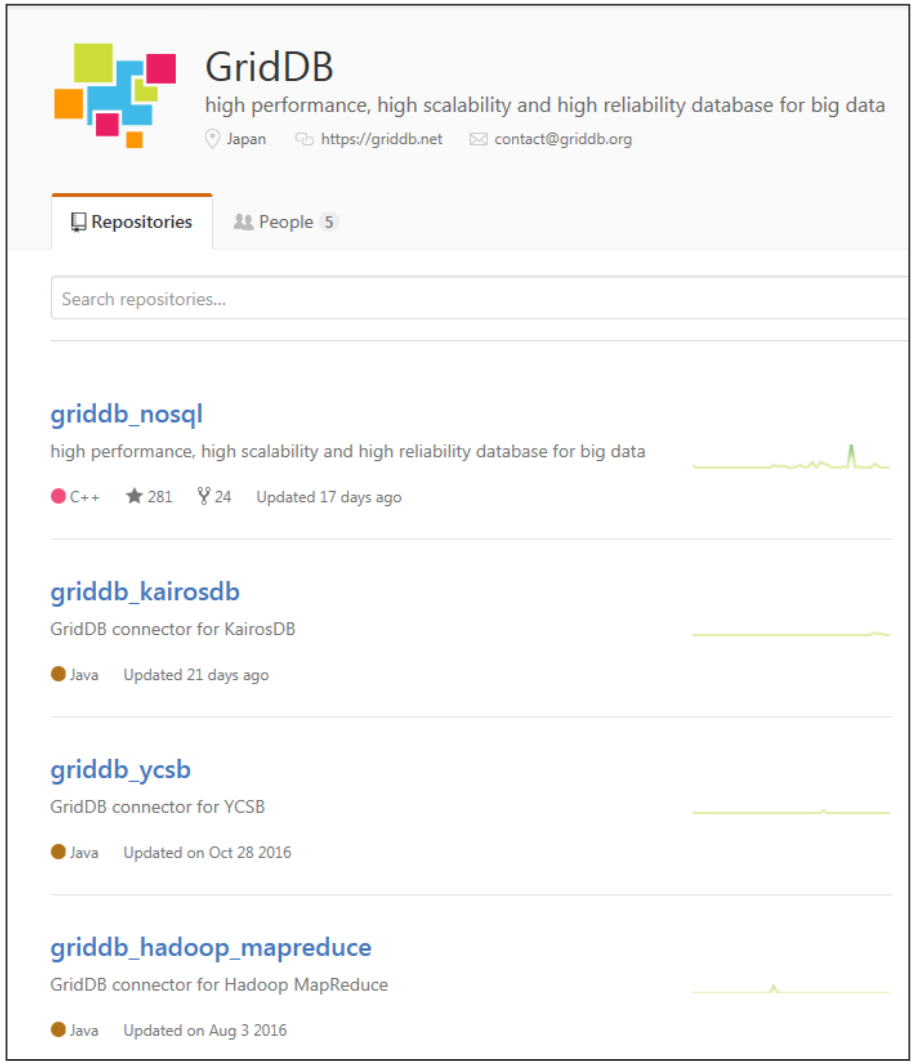
③GridDB CE (Community Edition)

Monitoring Dashboard



GridDB CE

- ビッグデータ技術の普及促進と多様なニーズの把握
- GitHub上にNoSQL主要機能をソース公開
  https://github.com/griddb/griddb_nosql/
- 主要OSSとのコネクタ、
  様々な開発言語のクライアントもソース公開
- 開発者向け情報
  https://griddb.net



クイックスタート

1. ノード毎のインストールおよびセットアップ
   - rpmコマンド
   - クラスタ共通のクラスタ定義ファイル(gs_cluster.json)の配置
   - ノードごとのノード定義ファイル(gs_node.json)の配置
2. クライアントのインストール
   - rpmコマンド
3. クラスタの起動
   - 全ノードの起動
   - クラスタの構成
4. クラスタの停止
   - 全ノードの停止
   - クラスタ構成の解除

```
$ gs_startnode -u admin/admin –w

$ gs_joincluster -c dms_150 -n 6 -u admin/admin

$ gs_stat -u admin/admin
{
    "checkpoint": {
        "archiveLog": 0,
        "backupOperation": 0,
        "duplicateLog": 0,
        "endTime": 1535672733362,
        "mode": "NORMAL_CHECKPOINT",
        "normalCheckpointOperation": 2152,
        "pendingPartition": 0,
        "requestedCheckpointOperation": 0,
        "startTime": 1535672733361
    },
    "cluster": {
        "activeCount": 6,
        "clusterName": "dms_150",
        "clusterStatus": "MASTER",
        "designatedCount": 6,
....

$ gs_stopcluster -u admin/admin

$ gs_stopnode -u admin/admin
```

GridDB設定例

gs_node.json

```
{
    "dataStore":{
        "dbPath":"data",
        "backupPath":"backup",
        "syncTempPath":"synctemp",
        "storeMemoryLimit":"10240MB",
        "storeWarmStart":false,
        "concurrency":3,
        "logWriteMode":1,
        "persistencyMode":"NORMAL",
        "affinityGroupSize":4,
        "storeCompressionMode":0
    },
    "checkpoint":{
        "checkpointInterval":"60s",
        "checkpointMemoryLimit":"1024MB",
        "useParallelMode":false
    },
    "cluster":{
        "servicePort":10190
    },
    "sync":{
        "servicePort":10170
    },
    "system":{
```

gs_cluster.json

```
{
    "dataStore":{
        "partitionNum":128,
        "storeBlockSize":"1MB"
    },
    "cluster":{
        "replicationNum":2,
        "notificationAddress":"239.0.0.1",
        "notificationPort":20150,
        "notificationInterval":"5s",
        "heartbeatInterval":"5s",
        "loadbalanceCheckInterval":"180s",
        "clusterName" : "dms_150"
    },
    "sync":{
        "timeoutInterval":"30s"
    },
    "transaction":{
        "notificationAddress":"239.0.0.1",
        "notificationPort":30150,
        "notificationInterval":"5s",
        "replicationMode":1,
        "replicationTimeoutInterval":"10s"
    },
    "sql":{
```

TOSHIBA

Leading Innovation