「ものづくり」の現場に必要な機能を備えたスケールアウト型データベース GridDBとそのオープンソース活動
～膨大なIoTデータの管理を実現 ～

TOSHIBA

東芝デジタルソリューションズ株式会社
野々村 克彦
2019.2.22

プロフィール

Katsuhiko Nonomura

knonomura

Block or report user

Organizations

名前：野々村 克彦

所属：東芝デジタルソリューションズ（株）

ソフトウェア＆AIテクノロジーセンター　知識・メディア処理技術開発部

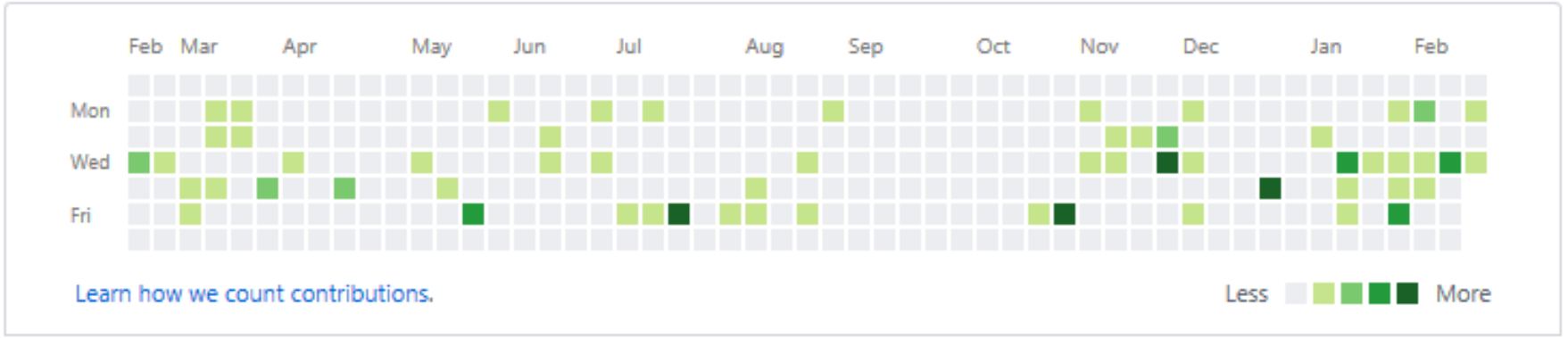
２０１１年　スケールアウト型DB GridDBの開発メンバ

２０１５年　GridDBのオープンソースPJ開始

現在 GridDBコミュニティ版の開発・コミッター、海外展開の技術支援など。

GitHub歴　４年

320 contributions in the last year



出身高校：県立米子東（鳥取県）　祝！23年ぶり甲子園出場

発表内容

1. スケールアウト型データベースGridDBの概要

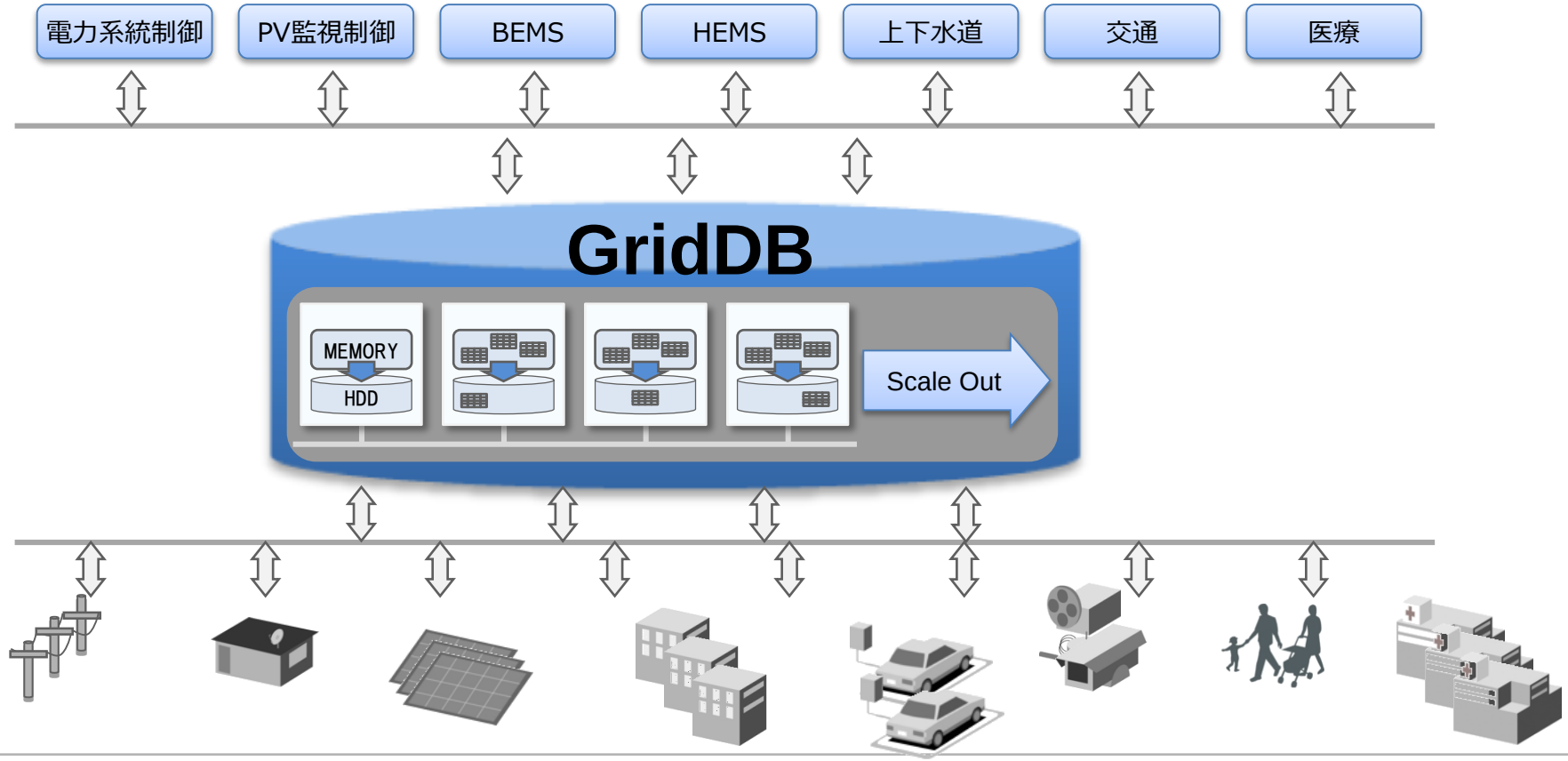
2. GridDBのオープンソース活動

- GridDB V4.1 CE(Community Edition)
- API拡充（Node.JSクライアント、WebAPI）
- その他の活動

3. まとめ

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
- V1.0製品化(2013年)、OSS化(2016年)、V4.1(2019年1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



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
・デンソー Factory-IoT (IoTを活用したダントツ工場)
　https://www.youtube.com/watch?v=9-yf-XN1Bgg&feature=youtu.be (ビデオ)
・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
….

- 東芝のIoT「SPINEX」の構成ソリューション

「ものづくり」の現場からの要求

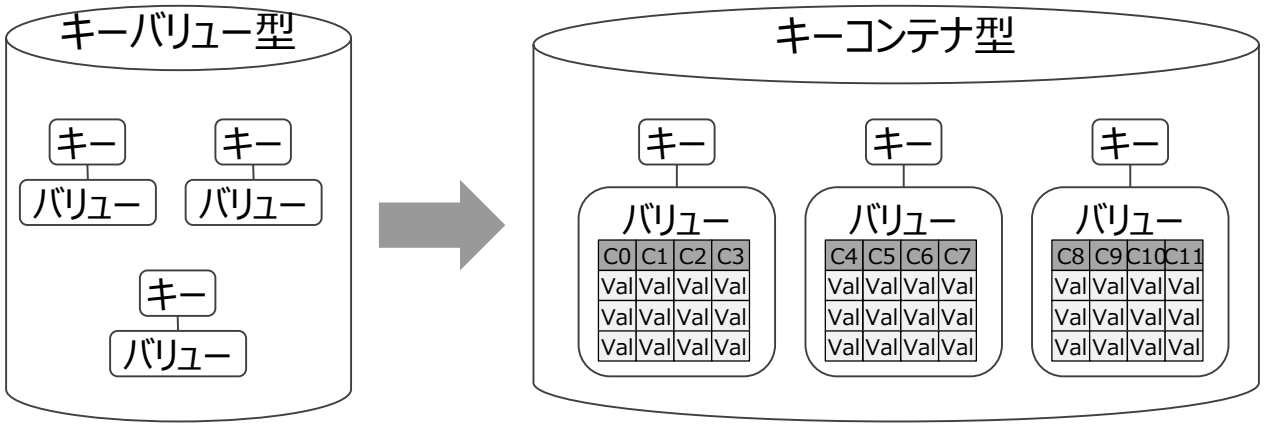
- 「製品がいつ・どこで・どのように製造されたか」を追跡して可視化するトレーサビリティの確保が求めれている。
- そのためには、製造装置や製造ラインから発生するIoTデータを、10年や20年といった長期間にわたって管理する必要がある。

GridDBの特長

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"

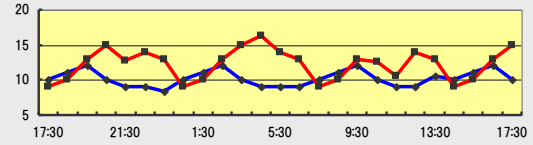
コレクション(Collection)

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"

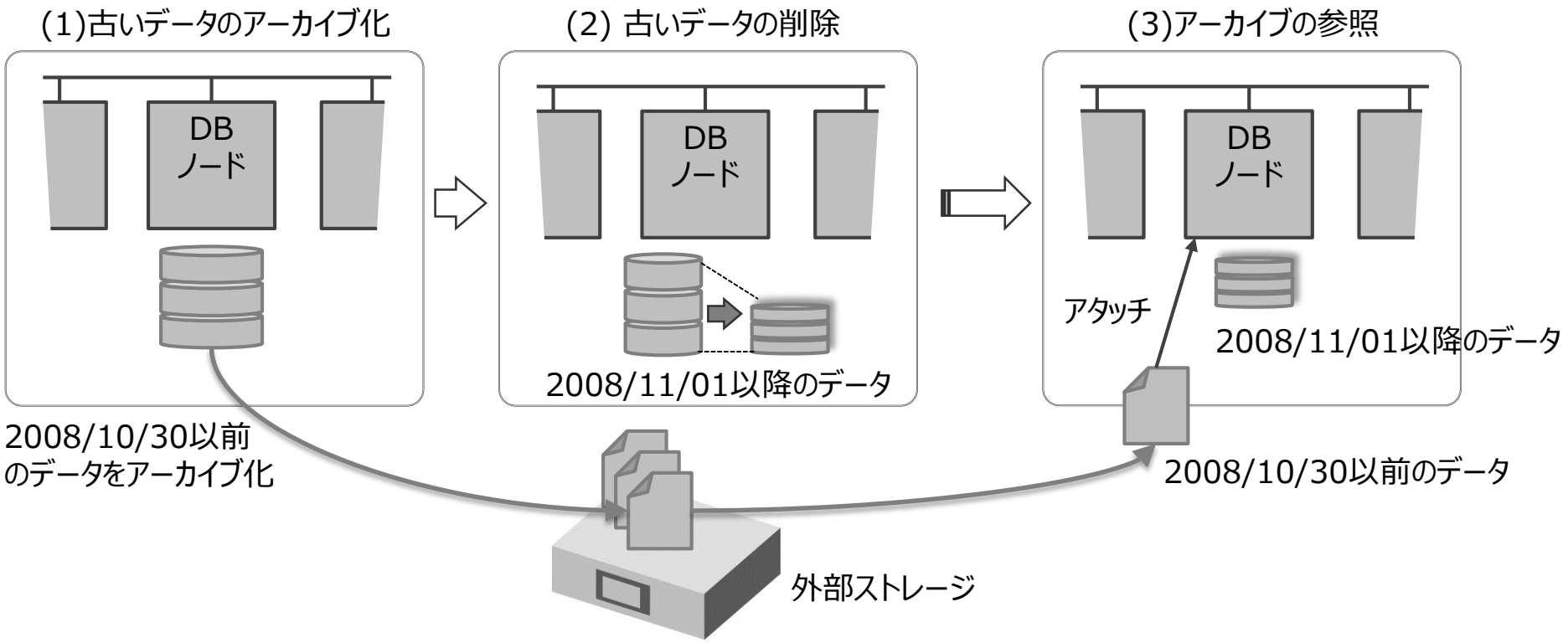
時系列(TimeSeries)

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



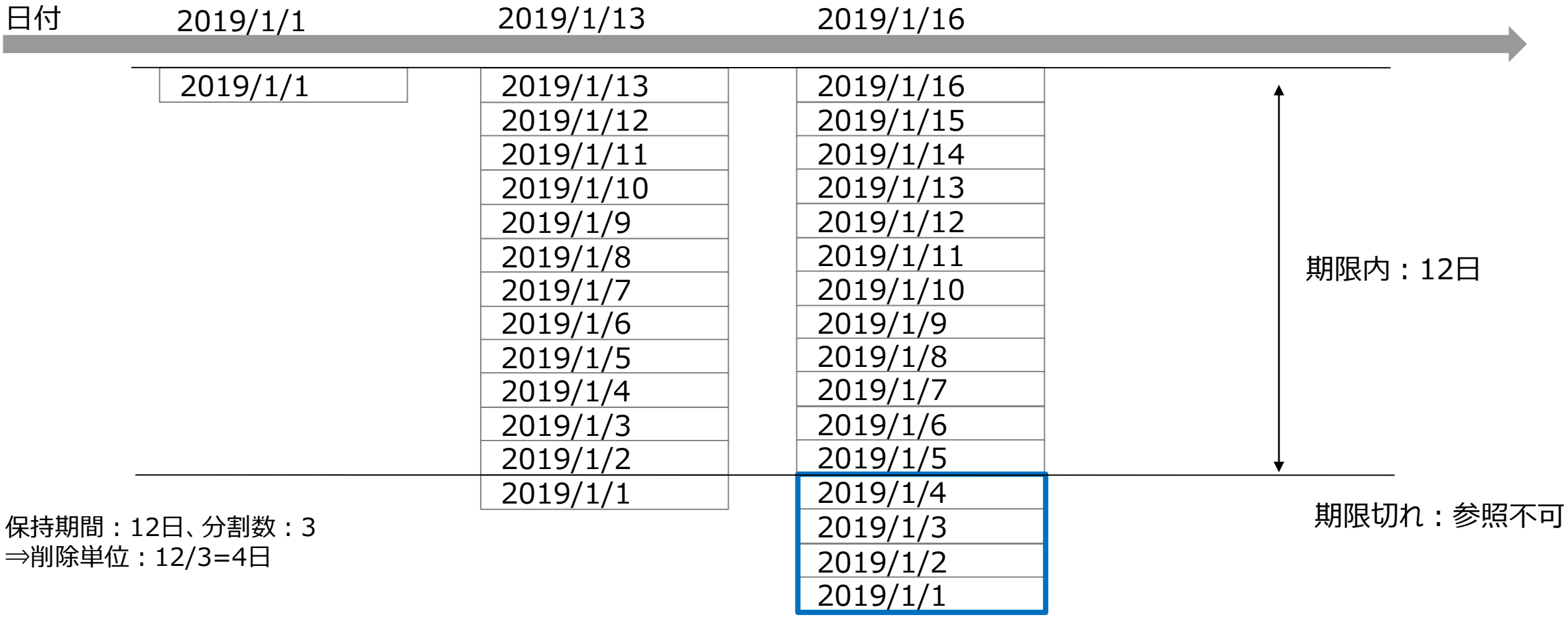
長期アーカイブ機能

- 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



具体的な挙動

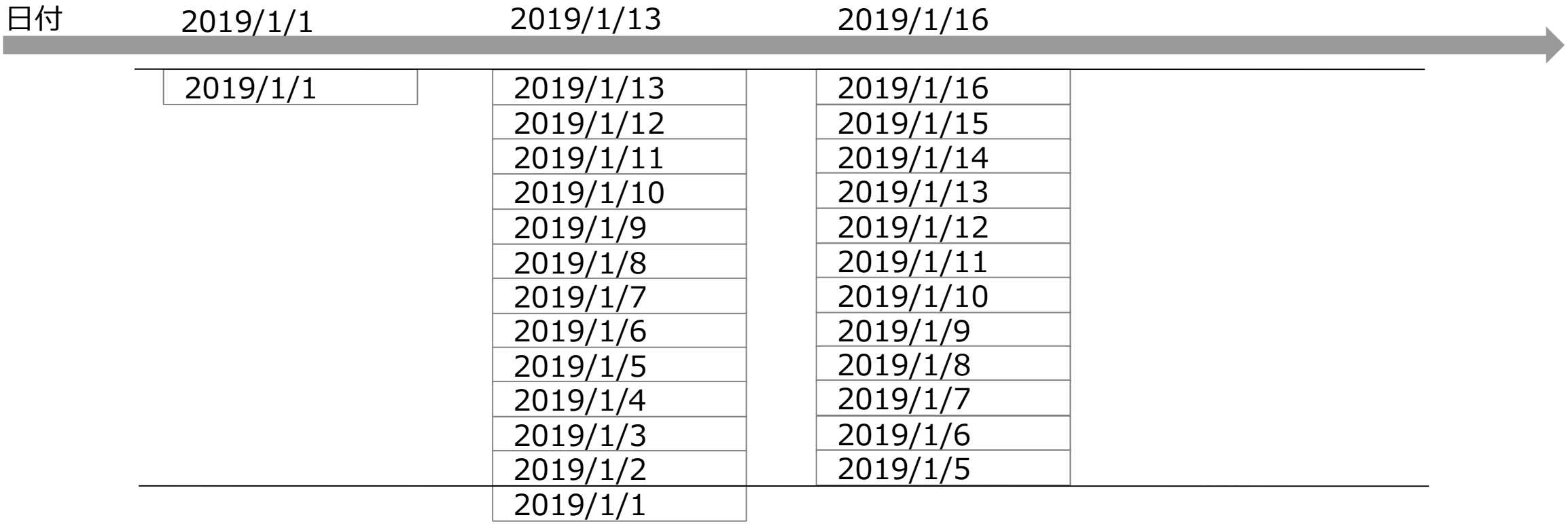
（従来の）期限解放機能



保持期間：12日、分割数：3
⇒削除単位：12/3=4日

具体的な挙動

（従来の）期限解放機能

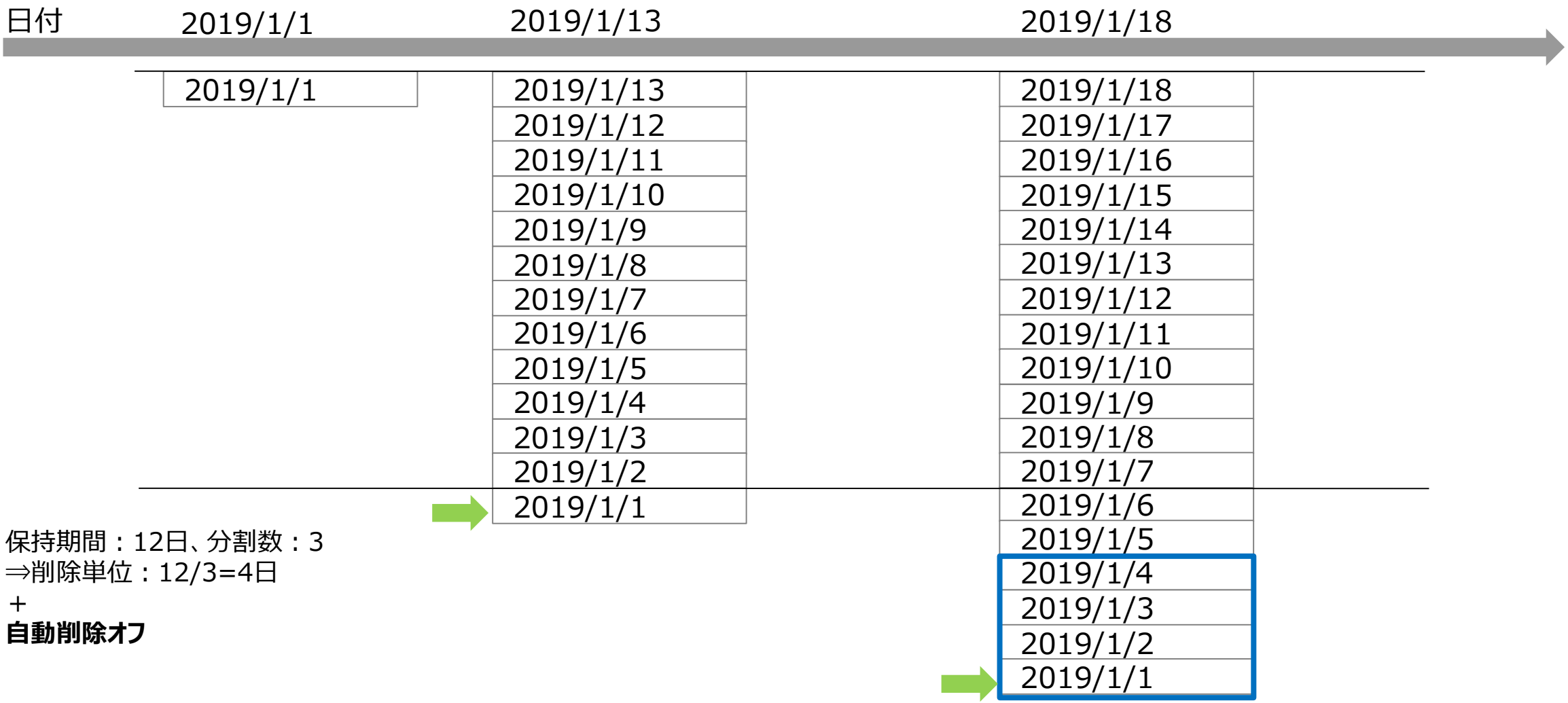


保持期間：12日、分割数：3
⇒削除単位：12/3=4日

効率的に
自動削除

具体的な挙動

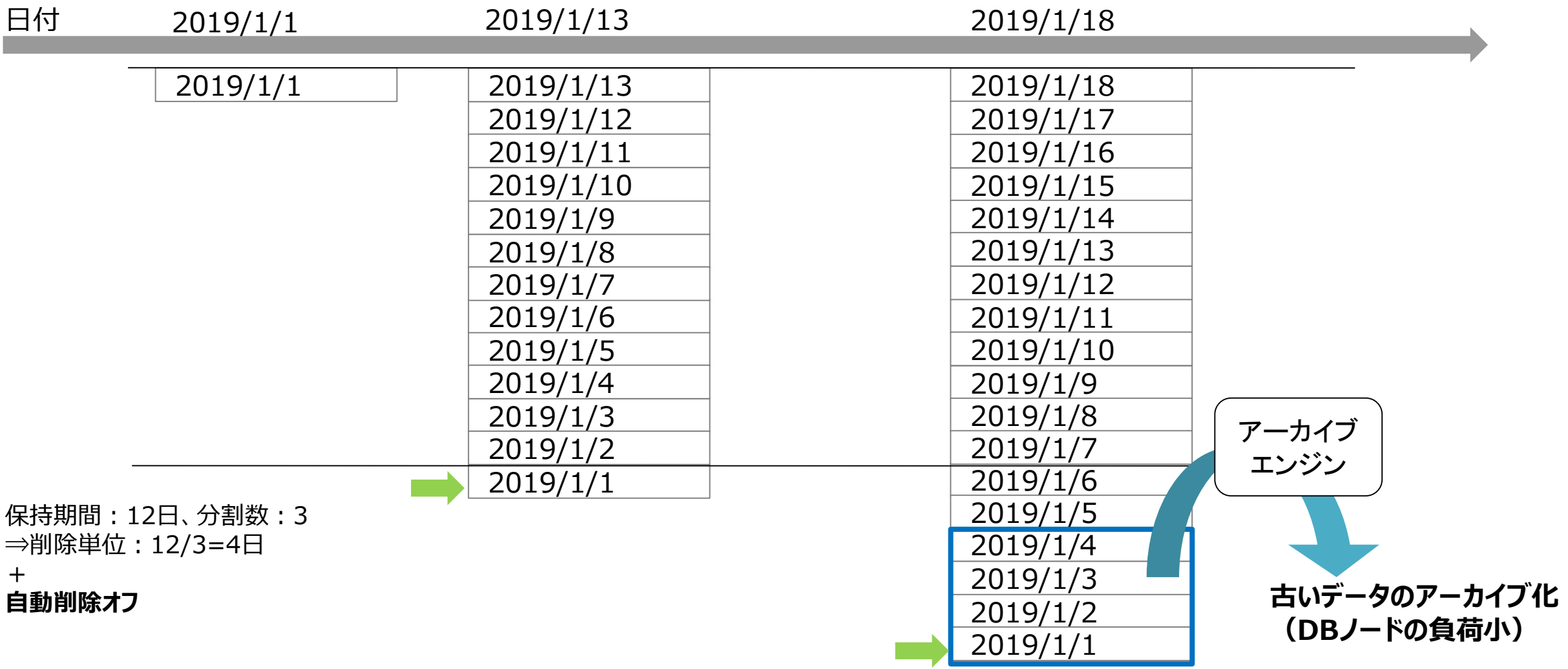
長期アーカイブ：期限解放機能を拡張



保持期間：12日、分割数：3
⇒削除単位：12/3=4日
＋
自動削除オフ

具体的な挙動

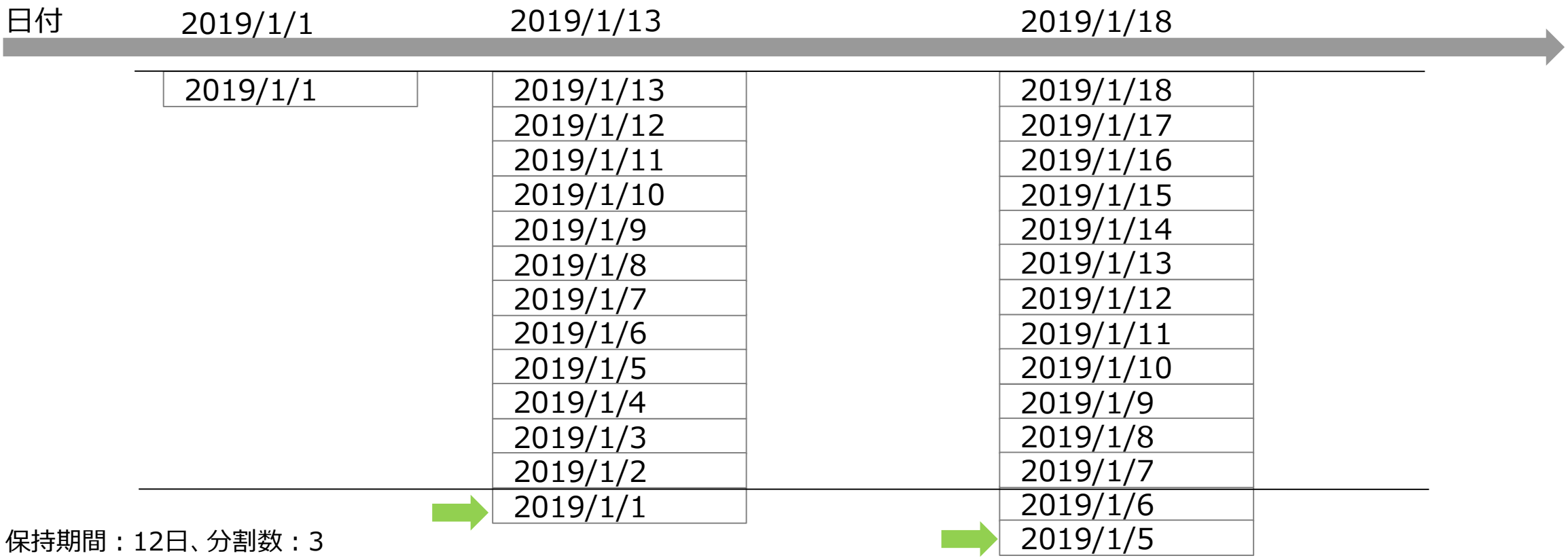
長期アーカイブ：期限解放機能を拡張



保持期間：12日、分割数：3
⇒削除単位：12/3=4日
＋
自動削除オフ

具体的な挙動

長期アーカイブ：期限解放機能を拡張



保持期間：12日、分割数：3
⇒削除単位：12/3=4日
＋
自動削除オフ

古いデータの削除
（DBノードの負荷小）

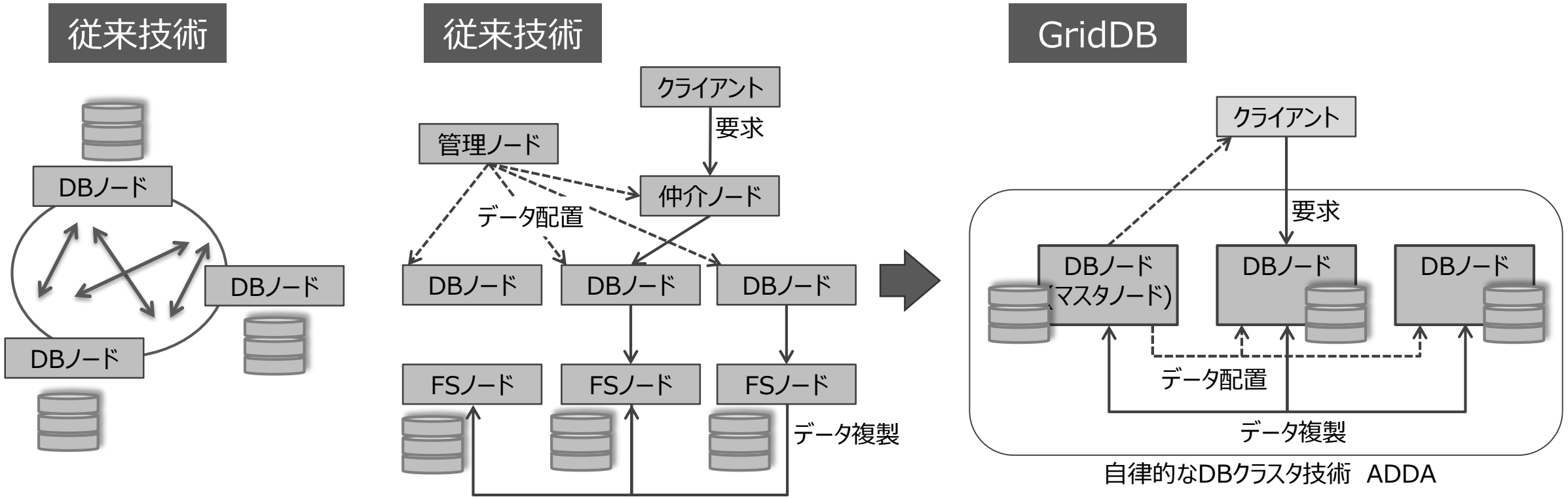
GridDBの特長

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

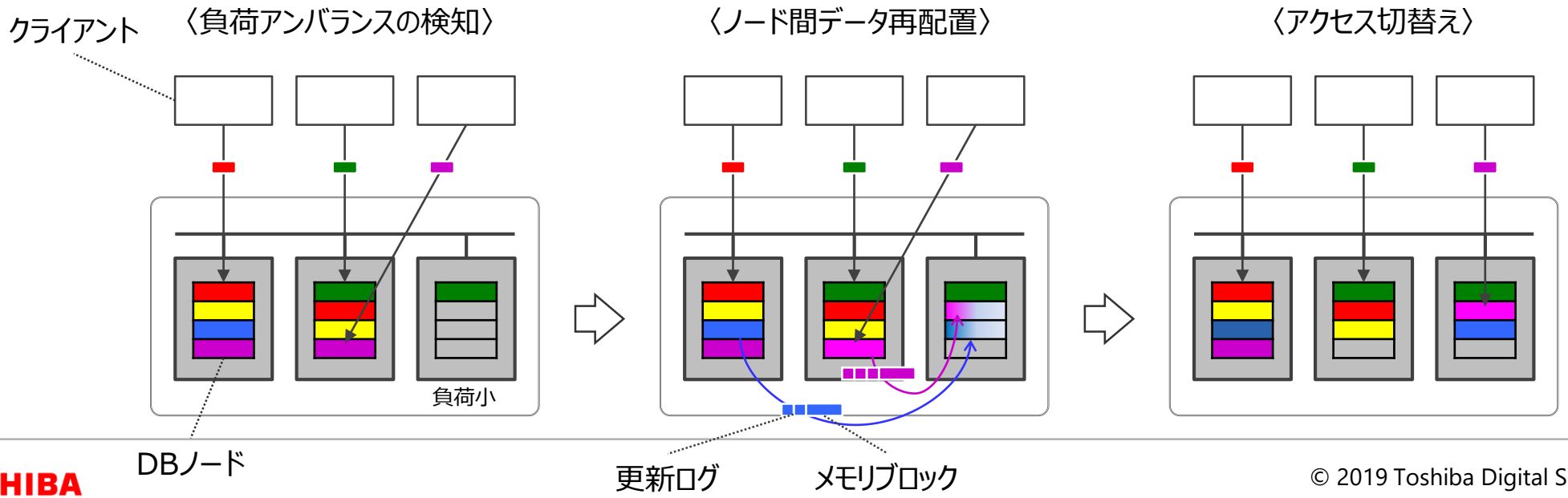
平成30年度 関東地方発明表彰

- 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

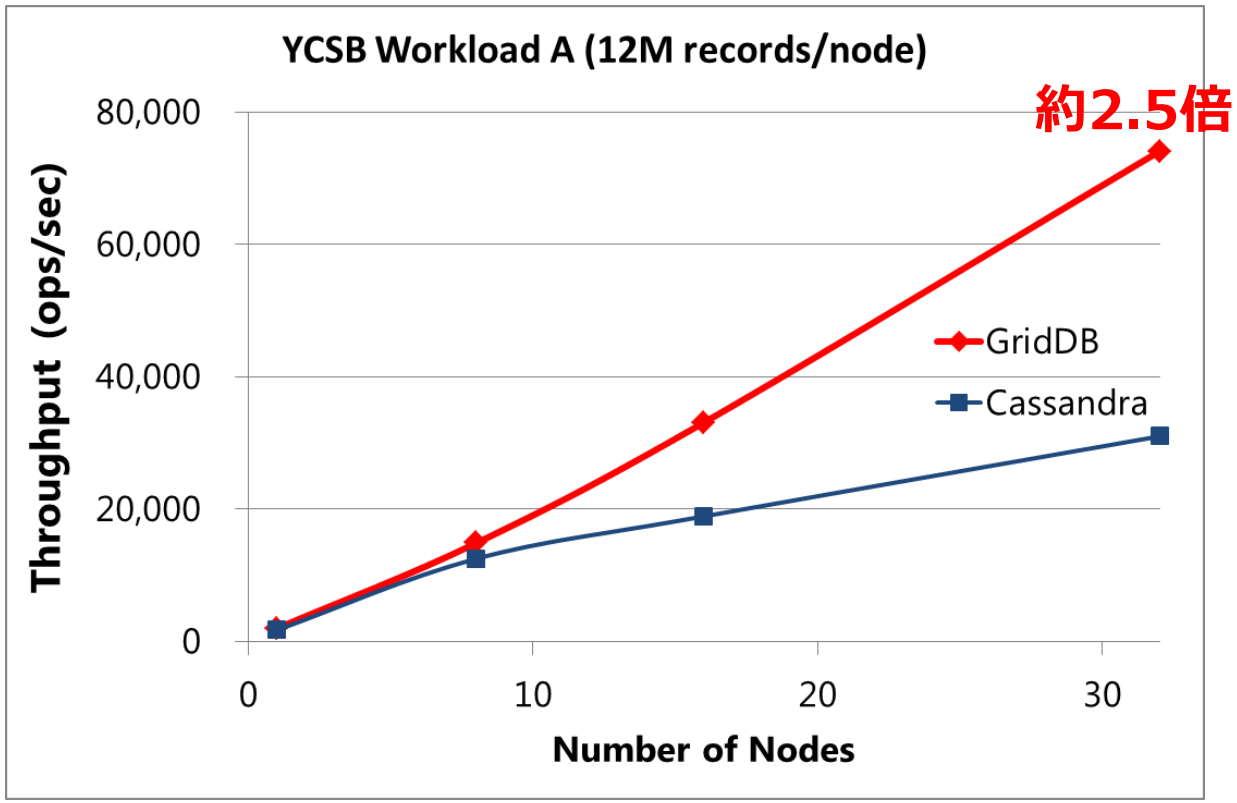


GridDBの特長

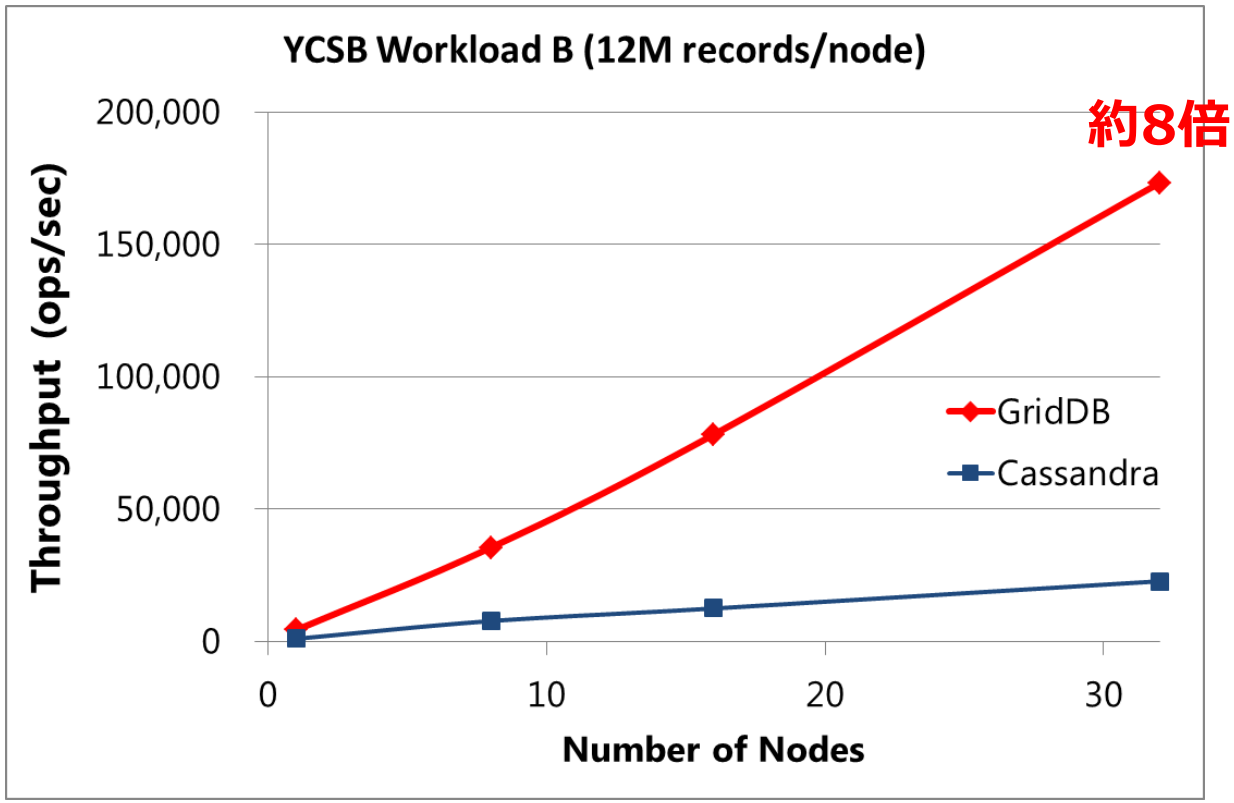
IoT指向のデータモデル	• データモデルはキー・コンテナ。コンテナ内でのデーター貫性を保証 • 時系列データ管理する特別な機能 • **過去データをコールド保存する長期アーカイブ機能**
高い信頼性と可用性	• データの複製をノード間で自動的に実行 • ノード障害があってもフェールオーバによりサービス継続 • 数秒から数十秒の切替え時間
高いスケーラビリティ	• 少ないノード台数で初期投資を抑制 • **負荷や容量の増大に合わせたノード増設が可能** • **自律データ再配置により、高いスケーラビリティを実現**
高性能なNoSQL+SQL	• メモリを主、ストレージを従としたハイブリッド型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

NoSQL性能

- **YCSB (Yahoo! Cloud Serving Benchmark)**
  NoSQLの代表的なベンチマーク　https://github.com/brianfrankcooper/YCSB



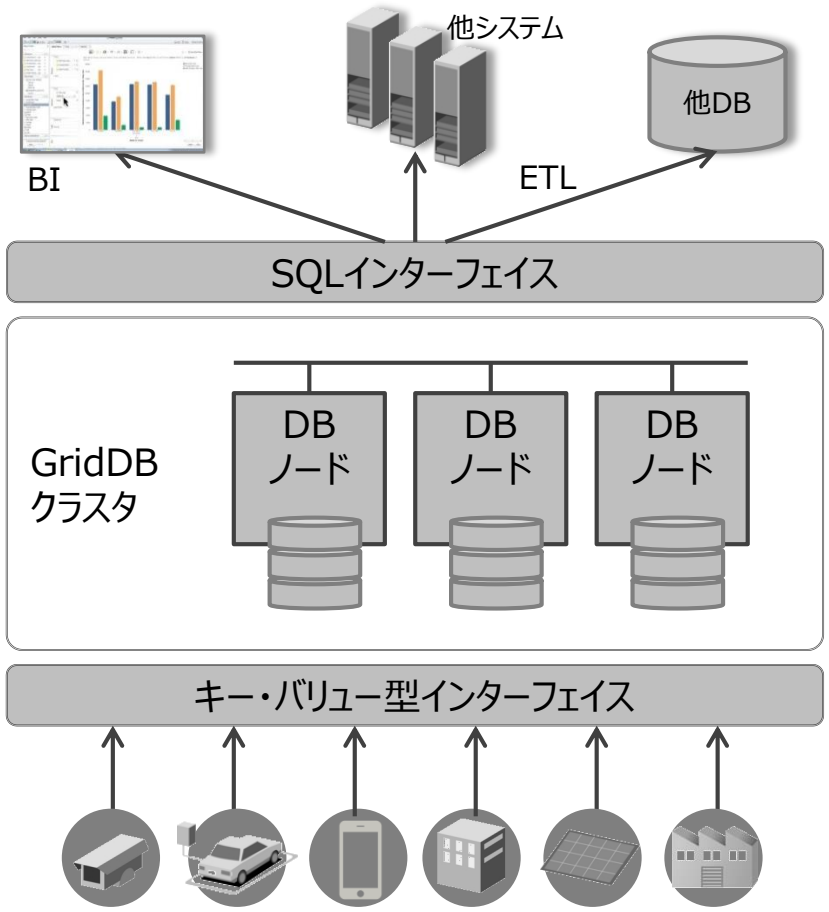
Read 50% + Write 50%



Read 95% + Write 5%

※フィックスターズ社によるYCSBベンチマーク結果

NoSQLとSQLのデュアルインターフェイス



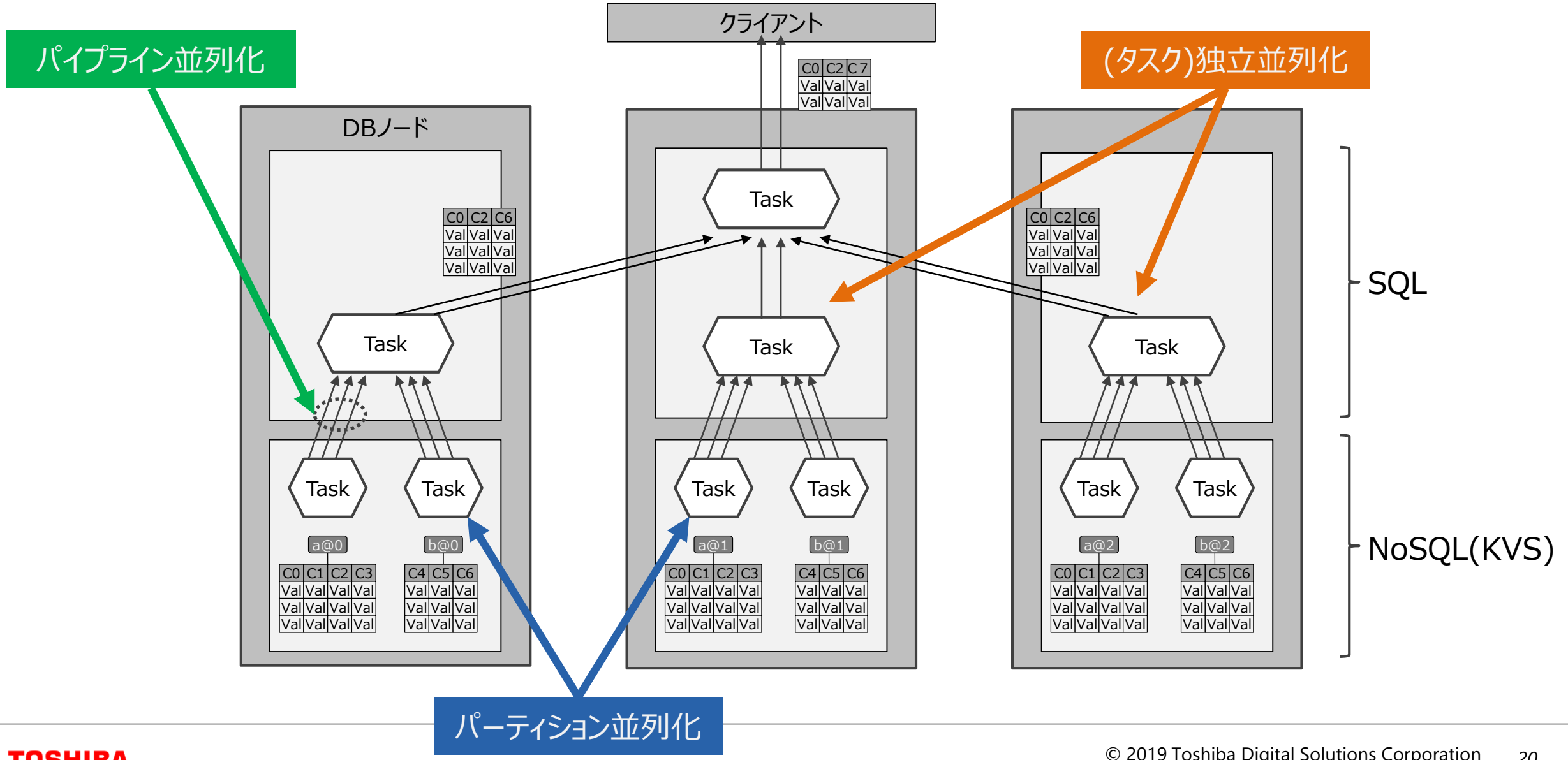
SQLインターフェイス

- 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

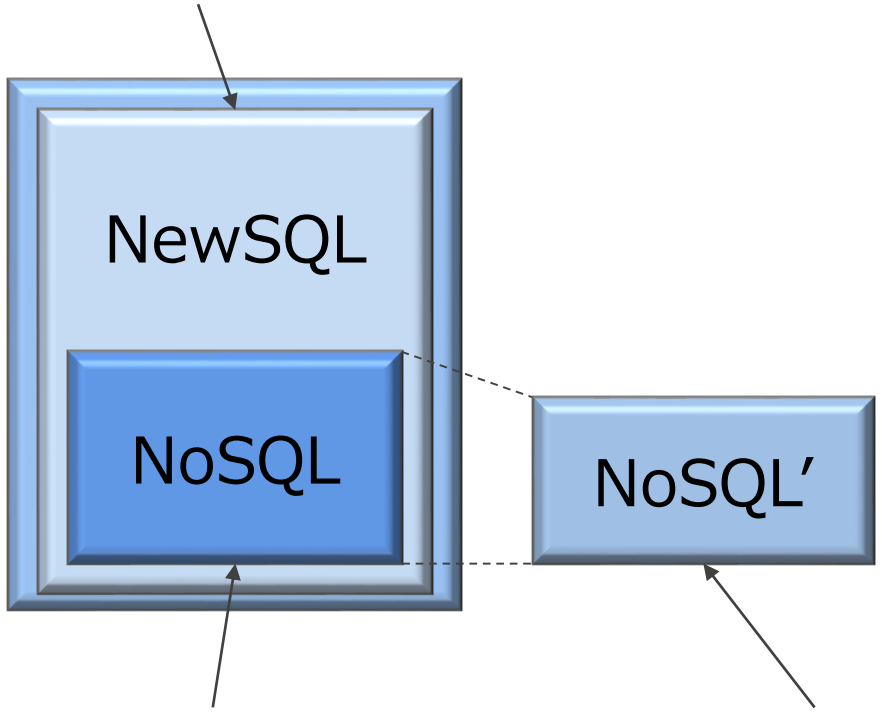
- 高可用、高スループット指向のKVS
- キーコンテナに対するCRUD
- Java/C/Python/Node.JS/Goクライアント

SQLにおける分散並列処理



製品ラインアップ

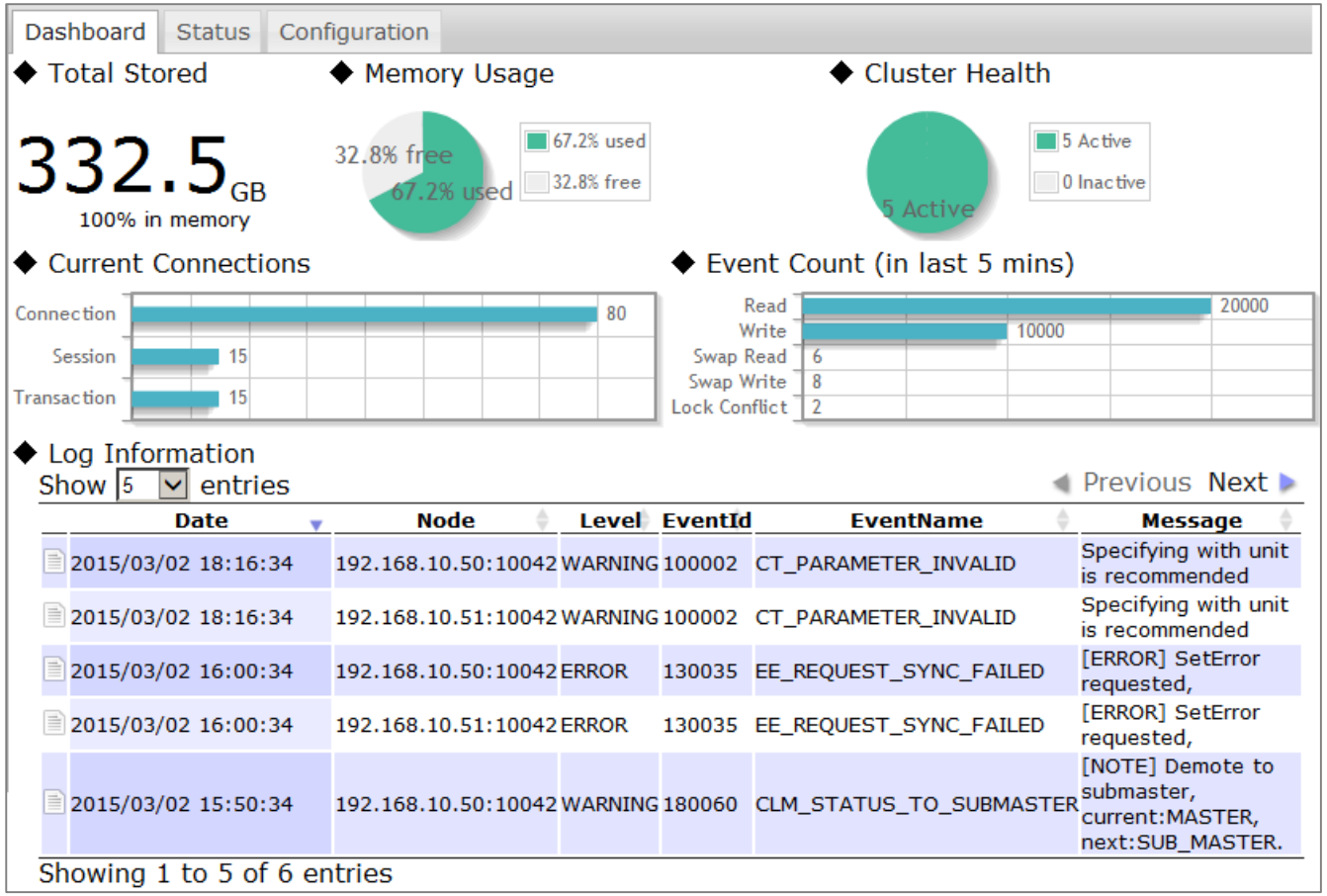
②GridDB AE
(Advanced Edition)



①GridDB SE
(Standard Edition)

③GridDB CE
(Community Edition)

Monitoring Dashboard



GridDBのオープンソース活動

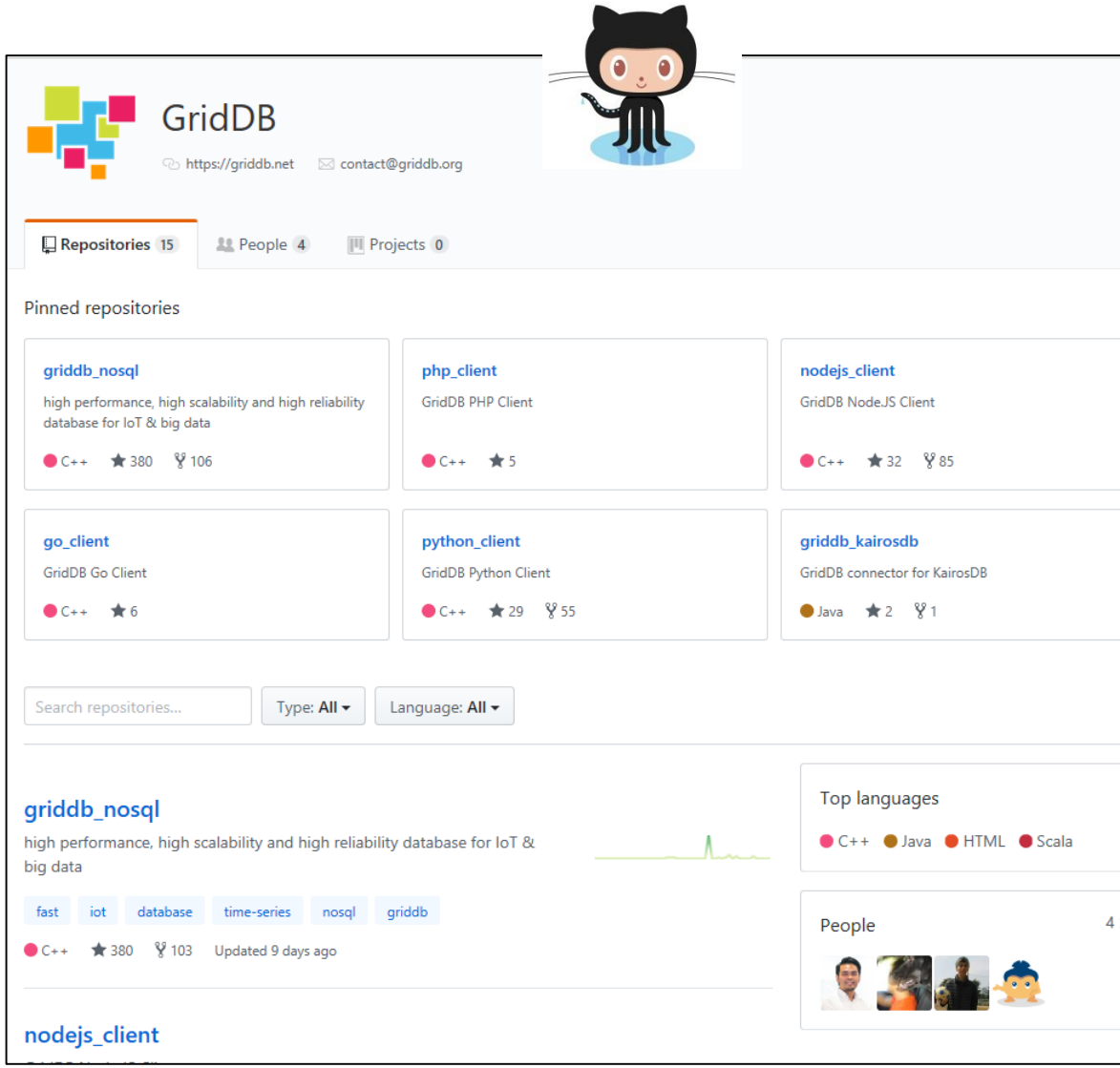
- **GridDB V4.1 CE**
- **API拡充**
  （Node.JSクライアント、WebAPI）
- **その他**

GitHubサイト https://github.com/griddb/

griddb github | 検索

- **NoSQL機能、様々な開発言語のクライアント、主要OSSとのコネクタをソース公開**

  https://github.com/griddb/griddb_nosqlなど
- **目的**
  - ビッグデータ技術の普及促進
    - 多くの人に知ってもらいたい、使ってみてもらいたい。
    - いろんなニーズをつかみたい。
  - 他のオープンソースソフトウェア、システムとの連携強化



最近のOSS活動（前回報告）

2018年

- 2月　GridDB Pythonクライアントのソース公開(GitHub)
- 3月　GridDB Goクライアントのソース公開(GitHub)
- 5月　GridDB V4.0 Community Editionのソース公開(サーバ、Javaクライアント) (GitHub)
- 6月　GridDB Node.JSクライアントのソース公開(GitHub)
  GridDB PHPクライアントのソース公開(GitHub)
- 7月　GridDB V4.0 CE Cクライアントのソース公開(GitHub)
- 8月　GridDB用Kafkaコネクタ･サンプルのソース公開(GitHub)
- 9月　GridDB Pythonクライアントのパッケージ公開(PyPI)
  GridDB Node.JSクライアントのパッケージ公開(npm)

最近のOSS活動

2018年

- **10月　GridDB Perlクライアントのソース公開(GitHub)**
- **11月　YCSBリポジトリ本家へのPullRequest(活動中)**

※YCSBのメインコミッターからお誘いがあった

2019年

- **1月　GridDB V4.1 Community Editionのソース公開(GitHub)★**
- **2月　Javaクライアントのパッケージ公開(Maven Central Repository)**

GridDB V4.1 CEのUbuntu向けパッケージ公開(GitHub)

Node.JSクライアントのインタフェース改良 v0.8★

WebAPIのソース公開(GitHub)★

GridDB V4.1 CE
2019/1ソース公開

GridDB V4.1 CE 追加機能

1. オンライン増設

– 稼働中のクラスタに対して、無停止でノード増設やノード切り離しを行うことができます。

2. 空間型

– データ型として空間型が追加されました。また空間索引も利用できます。

3. 時系列圧縮

– 時系列データに適した圧縮を時系列コンテナで利用できます。

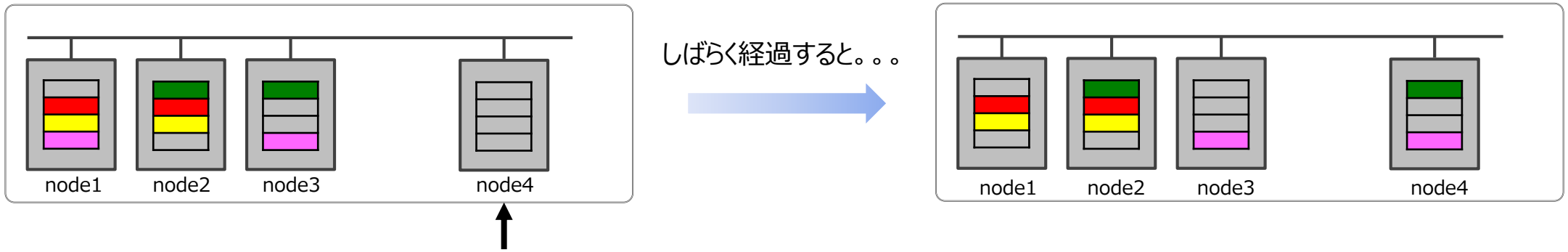
4. カラム数の上限値の拡大

– コンテナで扱えるカラム数の上限値を拡大しました。1024～32,000個

5. 動的なスキーマ変更(カラム追加)の改善

– カラム追加処理中のコンテナへの同時アクセスが可能になります。

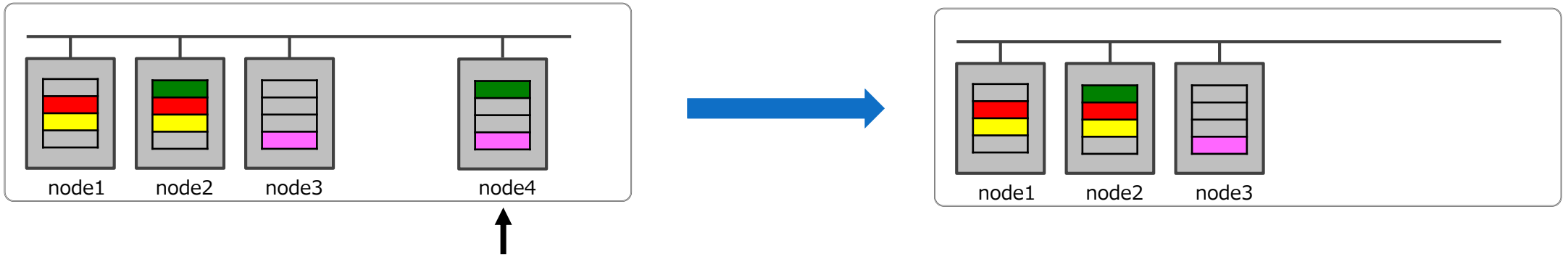
オンライン増設（ノード増設）



gs_appendcluster –u (username)/(passwd) –cluster (node1/2/3のいずれかのIPアドレス) –s (node4のIPアドレス)

- **手順**
  1. クラスタの稼働状況を確認する。
  2. 増設したいノード（node4）を起動する。
  3. 増設したいノードに対し増設コマンド（gs_appendcluster）を実行する。
  4. 増設したいノードの状況を確認する（gs_stat）。クラスタ状態がFOLLOWERになっていればOK
- **注意点**
  - 構成ノード=有効ノード数（現在クラスタに参加している台数）の場合に使用できます。

オンライン増設（ノード切り離し）



gs_leavecluster –u (username)/(passwd) –s (node4のIPアドレス)

- **手順**
  1. クラスタの稼働状況を確認する。
  2. 切り離したいノード（node4）に対し離脱コマンド（gs_leavecluster）を実行する。
- **注意点**
  - 指定したノードを離脱させるとデータロストが起こる可能性がある場合、クラスタの縮小は行えません。

API拡充

- **Node.JSクライアント**
  2019/2 インタフェース改良　v0.8
- **WebAPI**
  まもなくソース公開予定

Node.JSクライアント

- **実現方法**
  - GridDB CクライアントとSWIG(Simplified Wrapper and Interface Generator)を利用
  - Node.js V4で導入されたPromise(非同期処理を抽象化したオブジェクト)を利用して、Node.js層でラッピングすることで、Webアプリに必要となる「非同期機能」に対応

sample.js (ユーザプログラム)

```
var griddb = require('griddb_node');
con.put(["key1", 1, false])
.then(con => {
    console.log("completed");
})
```

griddb_node.js (Node.js層)

```
var griddb = require('griddb_client');
class Container {
  // 非同期機能
  put(arr) {
    return new Promise(function(resolve, reject) {
      setTimeout(function(){
        try {
          resolve(this_.container.put(arr));
        } catch(err) {
          reject(err);
        } }, 0);});}
```

griddb_client.node (C++層)

```
Container::put(array<object> row){
    // Cクライアント呼び出し
}
```

- **最新版 v0.8の改良点**
  - Multi-Put/Get/Query (バッチ処理) への対応
  - Blob型のデータのやり取りを（文字列形式でなく）Bufferクラスで対応
  - Object型によるキーワード引数への対応：func({k1:v1, k2:v2})

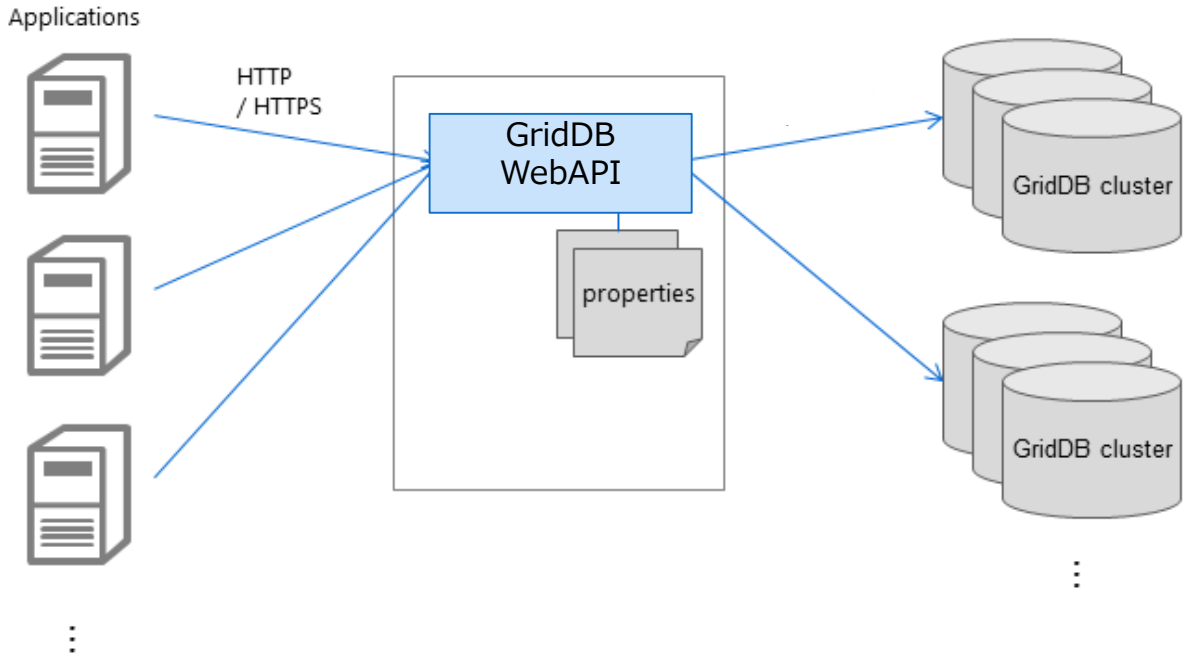
Node.JSクライアント　サンプル

```
var griddb = require('griddb_node');

var factory = griddb.StoreFactory.getInstance();

# プロパティ（主に接続用）の設定
var store = factory.getStore({
    "host": process.argv[2],
    "port": parseInt(process.argv[3]),
    "clusterName": process.argv[4],
    "username": process.argv[5],
    "password": process.argv[6]});

# コンテナ情報（コンテナ名、カラム情報、コンテナタイプなど）の設定
var conInfo = new griddb.ContainerInfo({'name': "col01",
    'columnInfoList': [
        ["name", griddb.Type.STRING],
        ["status", griddb.Type.BOOL],
        ["count", griddb.Type.LONG],
        ["lob", griddb.Type.BLOB] ],
    'type': griddb.ContainerType.COLLECTION,
    'rowKey': true});
```

```
var container;
# コンテナの生成
store.putContainer(conInfo, false)
    .then(cont => {
        container = cont;
       # 索引（カラム名、索引タイプ）の設定
        return container.createIndex({
              'columnName': 'count',
              'indexType': griddb.IndexType.DEFAULT });
    })
    .then(() => {
       # データの登録
        return container.put(["name01", false, 1,
              Buffer.from([65, 66, 67, 68, 69, 70, 71, 72, 73, 74])]);
    })
    .then(() => {
      # 検索
       query = container.query("select *")
       return query.fetch();
    })
    .then(rs => {
       while (rs.hasNext()) {
          console.log(rs.next());
       }
    })
```

WebAPI

- **実現方法**
  - GridDB JavaクライアントとSpring Bootフレームワークの組み込み型Tomcatを利用
- **機能一覧**
  - コンテナ生成、コンテナ削除
  - コンテナ名一覧取得、コンテナ情報取得
  - ロウデータ登録、ロウデータ削除
  - クエリ実行



WebAPI　サンプル１

- **コンテナ生成**
  - Method　POST
  - Request　http://[host]:[port]/griddb/v2/[clusterName]/dbs/public/containers
  - Bodyの例

```
{“container_name":"test",
"container_type":"COLLECTION", "rowkey":true,
"columns":[{"name":"col1", "type":"STRING", "index":["TREE"]},
            {"name":"col2", "type":"INTEGER"},
            {"name":"col3", "type":"BOOL}]}
// コンテナ名、コンテナタイプ、ロウキーの有無、カラム情報から成るコンテナ情報
```

WebAPI サンプル２

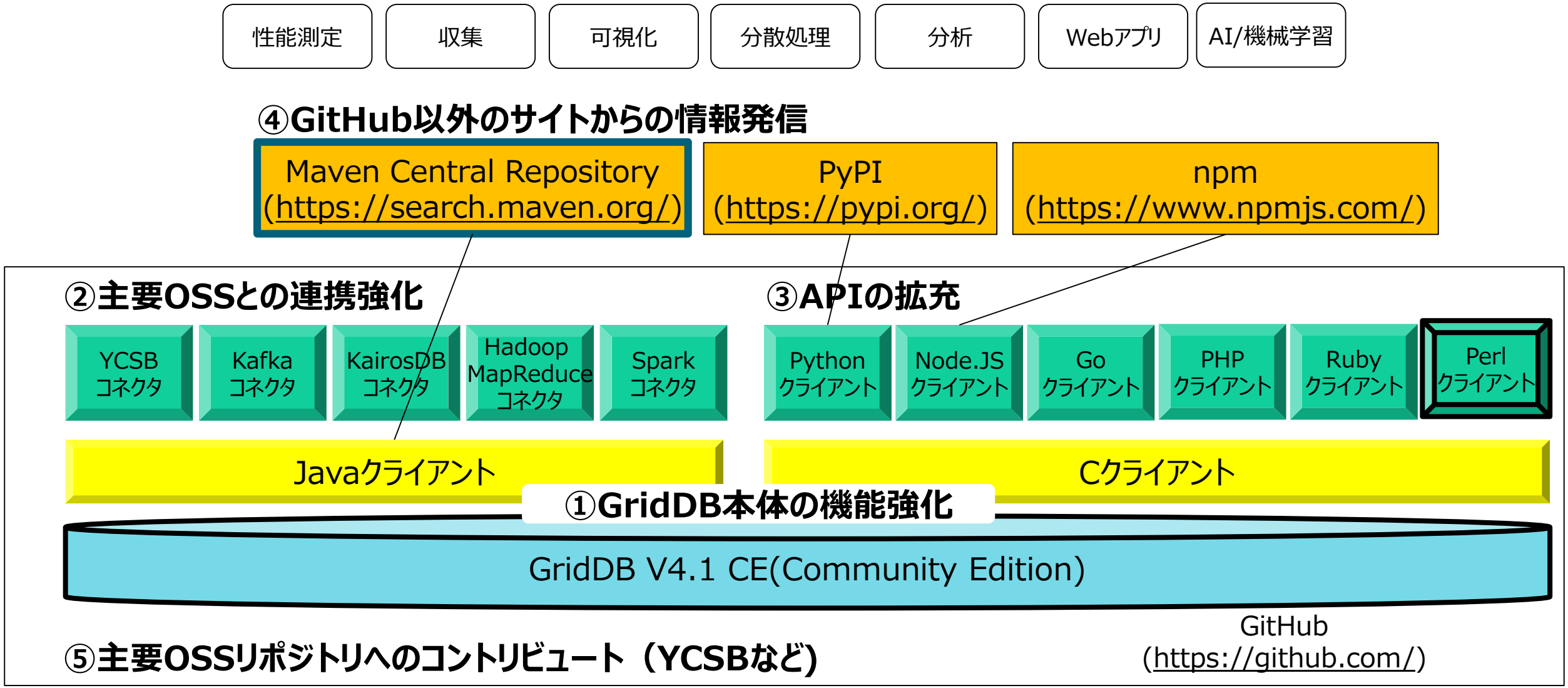
- **ロウデータ登録**
  - Method PUT
  - Request
    http://[host]:[port]/griddb/v2/[clusterName]/dbs/public/containers/[containerName]/rows
  - Bodyの例

    [[“value”, 1, true]] // ロウデータのリスト
- **クエリ実行**
  - Method POST
  - Request http://[host]:[port]/griddb/v2/[clusterName]/dbs/public/tql
  - Bodyの例

    {[“name”:”test”, “stmt”:“select *”, “columns”:[]]} // コンテナ名、クエリ文、出力カラム名

その他

GitHubサイトを中心とした活動状況



デベロッパーズサイト https://griddb.net/

griddb net | 検索

- **アプリケーション開発者向けのサイト**
- **様々なコンテンツを公開**
  - ホワイトペーパ、ブログ
  - マニュアル
  - サンプルコード

  など
- **コミュニケーションの場(フォーラム)を提供**

ツイッター https://twitter.com/griddb_jp

griddb_jp | 検索

- **GridDBに関するリリース、イベント、などをお知らせします。(日本国内向け)**

最近の記事

- **Technologist's magazine**［テクノロジストマガジン］vol.17発刊
  - 『GridDB』の開発を手掛ける東芝デジタルソリューションズ・服部雅一氏の紹介
  - https://www.criprof.com/magazine/2019/01/17/post-5569/
- **GridDB V4.1プレスリリース**
  - 膨大なIoTデータの長期保存を実現
    スケールアウト型データベース「GridDB®」機能強化版の提供を開始
  - https://www.toshiba-sol.co.jp/news/detail/20190115.htm

- **中国「2018年新オープンソースソフトウェアトップ50」にてGridDBが42位に選ばれた**
  - https://tech.sina.com.cn/roll/2019-01-26/doc-ihqfskcp0397662.shtml

※昨年春にGridDB V4.0 CEをソース公開した際に、中国人に注目された。GitHub中国版（？）のgitee.comから非常に多くのアクセスがあり、GridDBのスター数が2日間で30も上昇した。

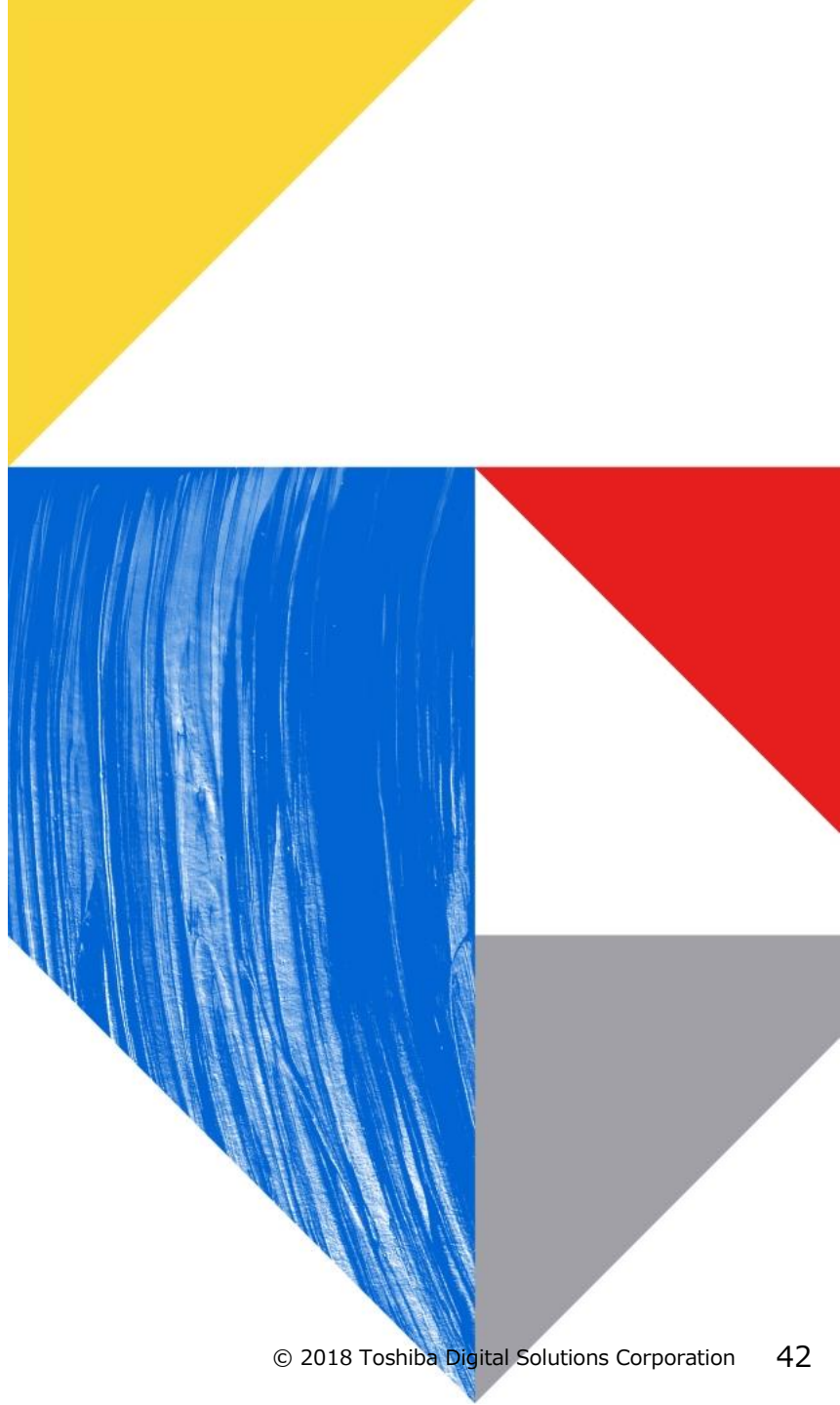
まとめ

- **GridDB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です。**
- **今回、「ものづくり」の現場に必要な機能を備えたGridDBの特長、GridDB V4.1CE、API拡充についてご説明いたしました。**
- **GitHubサイト、デベロッパーズサイトなどで、GridDB機能強化や主要OSSとの連携強化、APIの拡充などの様々なOSS活動を実施しています。**

オープンソースのGridDBを是非とも使ってみてください。

TOSHIBA

ご清聴ありがとうございました



ご参考：GridDBに関する情報

- **GridDB デベロッパーズサイト**
  - https://griddb.net/
- **GridDB GitHubサイト**
  - https://github.com/griddb/griddb_nosql/
- **Twitter: GridDB (日本)**
  - https://twitter.com/griddb_jp
- **Twitter: GridDB Community**
  - https://twitter.com/GridDBCommunity
- **Facebook: GridDB Community**
  - https://www.facebook.com/griddbcommunity/

- **Wiki**
  - https://ja.wikipedia.org/wiki/GridDB

- **GridDB お問い合わせ**
  - プログラミング関連：Stackoverflow(https://ja.stackoverflow.com/search?q=griddb)もしくはGitHubサイトの各リポジトリのIssueをご利用ください
  - その他：contact@griddb.netもしくはcontact@griddb.orgをご利用ください

ご参考：デベロッパーズサイトの最近のブログ

09-29-2018

GridDB PHPクライアント入門

わたしたちは、GridDBのPHPプログラミング言語用のデータベースコネクタをリリー
このコネクタはPHP
ジョンをサポートし
CentOSバージョン
します。

09-29-2018

Fixed ListでGridDBを使う方法

FIXED_LISTモードとMulticastモードでGridDBを実行する場合、どのような違いがあるので

09-29-2018

GridDB Node.jsクライアントを使ってみよう

GridDBは近年非常に人気のあるnode.jsとのデータベースコネクタ

09-29-2018

PostgreSQL用のGridDB外部データラッパーを使ってみよう

複数の異なるインタフェースから

06-09-2018

GridDBのGo言語 Client入門

GridDBは、Googleのプログラミング言語Go用のデータベースコ

06-06-2018

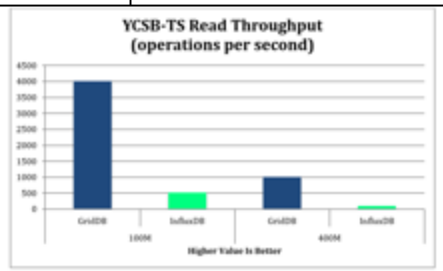
GridDB Community Edition バージョン4.0

GridDB Community Editionがアップデートされました。バージョン4.0では、コンテナ名やクラスター名の文字の幅を広げるなど、

04-05-2018

PostgreSQLからGridDBへの移行

このブログでは、PostgreSQL databaseをPythonを使ってGridDBへ移行する方法をご紹介します。



04-05-2018

GridDBとInfluxDBのTimeSeriesベンチマーク比較

時系列(Timeseries) データベースであるGridDBとInfluxDBを、CentOS 6.9 image を使った

ご参考：デベロッパーズサイトの主なコンテンツ（ハウツーもの）

- **AWS/Azure上で動かす方法**
  - 「GridDB Azureクラスタの構築」
  - 「Fixed ListでGridDBを使う方法」
- **Docker上で動かす方法**
  - 「Docker上でGridDBを実行する」
- **Pappetによる設定方法**
  - 「Puppetを使用したGridDBの設定方法」
- **各種クライアント**
  - 「GridDB's C/Python/Ruby APIsを使ってみよう」
  - 「GridDBのGo言語 Client入門」
  - 「GridDB Node.jsクライアントを使ってみよう」
  - 「GridDBのPHPクライアント入門」
- **MapReduceコネクタ**
  - 「Hadoop MapReduce用のGridDBコネクタの使い方」
- **Sparkコネクタ**
  - 「Apache SparkのためのGridDBコネクタ」
- **YCSBコネクタ**
  - 「YCSB向けGridDBコネクタを使ってみよう」

※https://griddb.net/ja/blog/

ご参考：デベロッパーズサイトの主なコンテンツ(ホワイトペーパなど)

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とInfluxDBのTimeSeriesベンチマーク比較
- GridDB Reliability and Robustness

など

ブログ：

- IoT産業におけるGridDB導入事例
- 自動車産業におけるGridDB導入事例
- 自律型データ配信アルゴルズム (ADDA)
- 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

など

ご参考：その他のコンテンツ

- **Wiki：GridDB**
  - https://ja.wikipedia.org/wiki/GridDB
- **動画：**db tech showcase Tokyo 2018のセッション『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** ～』
  - https://crash.academy/video/527/1855

ご参考：WebAPI コマンド一覧

Base URL: http(s)://<host>:<port>/griddb/v2/

- **コンテナ生成**
  - URL： POST {cluster}/dbs/public/containers
  - BODY：コンテナ情報
- **コンテナ削除**
  - URL： DELETE {cluster}/dbs/public/containers
  - BODY：コンテナ名の一覧
- **コンテナ名一覧取得**
  - URL： GET {cluster}/dbs/public/containers
  - BODY：取得条件 (type, limit, offset, sort)
- **コンテナ情報取得**
  - URL： GET {cluster}/dbs/public/containers/{container}/info
- **ロウデータ登録**
  - URL：PUT {cluster}/dbs/public/containers/{container}/rows
  - BODY：ロウデータの一覧
- **ロウデータ取得**
  - URL：POST {cluster}/dbs/public/containers/{container}/rows
  - BODY：取得条件 (offset, limit, condition, sort)
- **ロウデータ削除**
  - URL：DELETE {cluster}/dbs/public/containers/{container}/rows
  - BODY：ロウキーの一覧
- **クエリ実行**
  - URL：POST {cluster}/dbs/public/tql
  - BODY：{name, stmt, columns}の一覧