

NoSQL/SQLデュアルインターフェースを備えた IoT向けデータベースGridDB ~ GridDB CE 4.5のSQLインタフェースを使ってみましょう ~



TOSHIBA

東芝デジタルソリューションズ株式会社 GridDBコミュニティ版担当

野々村 克彦

2020.10.23

Contents

01 IoT向けデータベースGridDBの概要

02 SQLインターフェース（ハンズオン）

https://github.com/konomura/griddb-docker/blob/master/README_ja.md

03 OSS活動

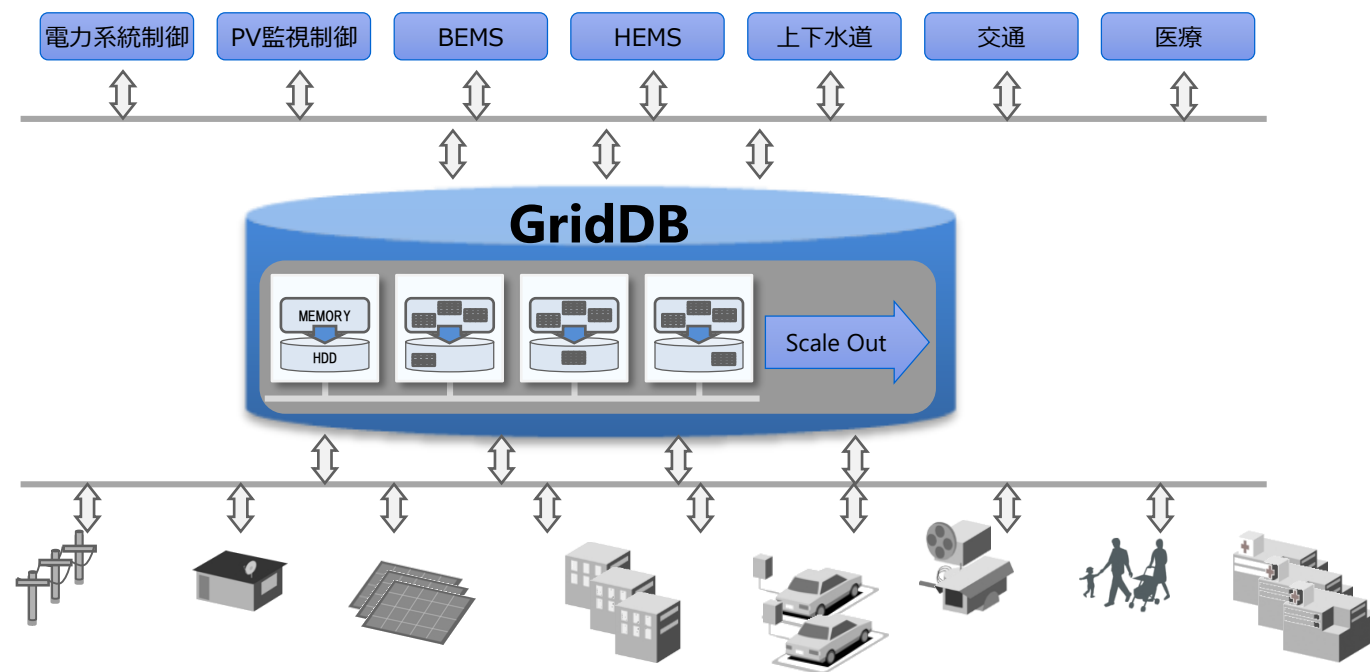
04 まとめ

01

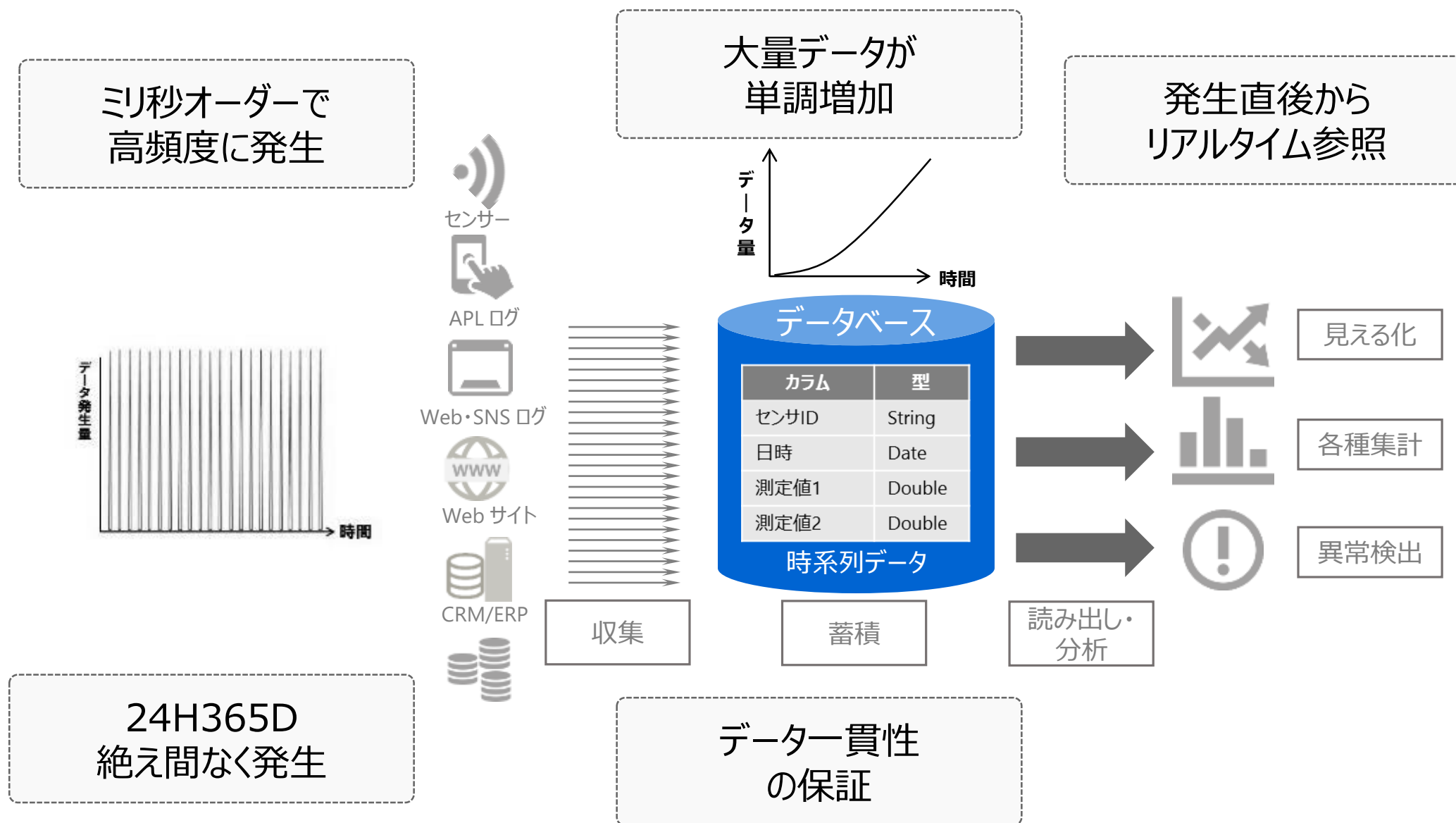
IoT向けデータベースGridDBの概要

IoT向けデータベースGridDB

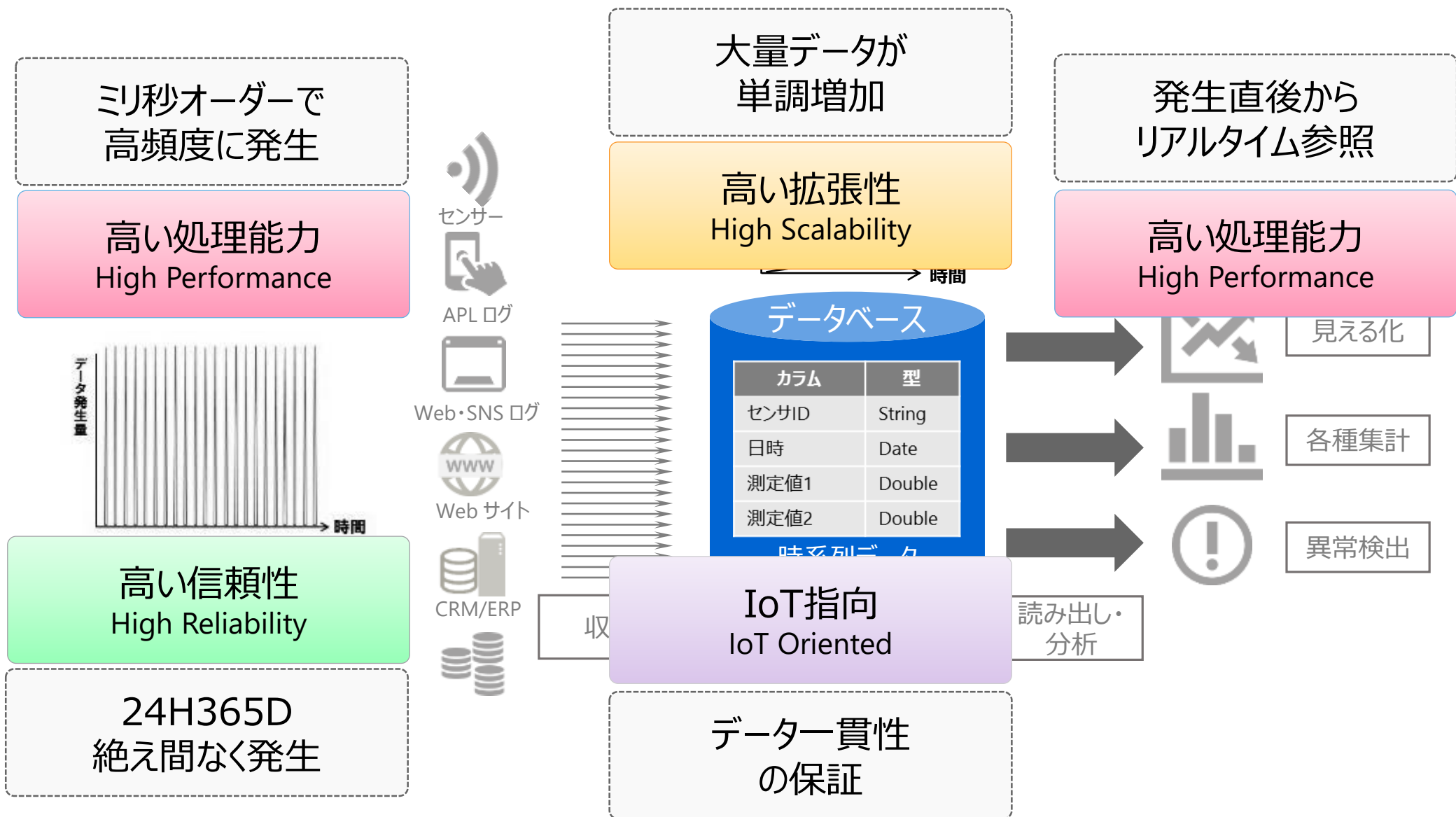
- 日本発のビッグデータ／IoT向けデータベース
- V1.0製品化(2013年)、OSS化(2016年)、V4.5CE(2020年6月)
- 社会インフラ、製造業を中心に、高い信頼性・可用性が求められるシステムに適用されている



IoTデータの特長



データベースへの要求



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一貫性を保証
- 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ーによりサービス継続
- 数秒から数十秒の切替え時間

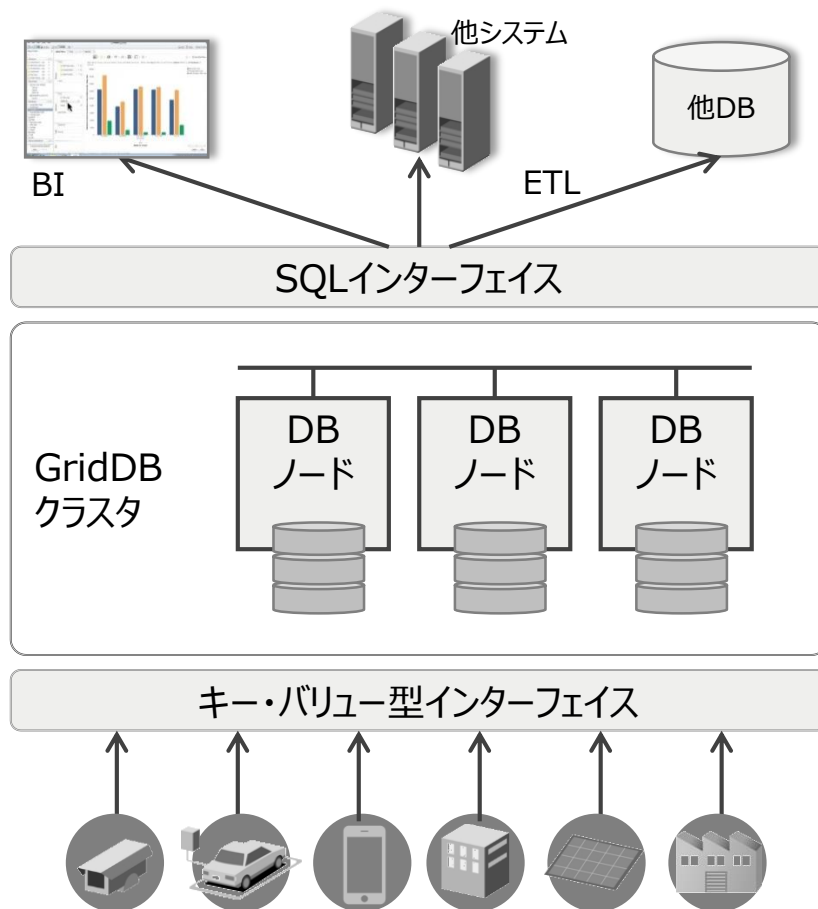
高い拡張性

- 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

高い処理能力 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

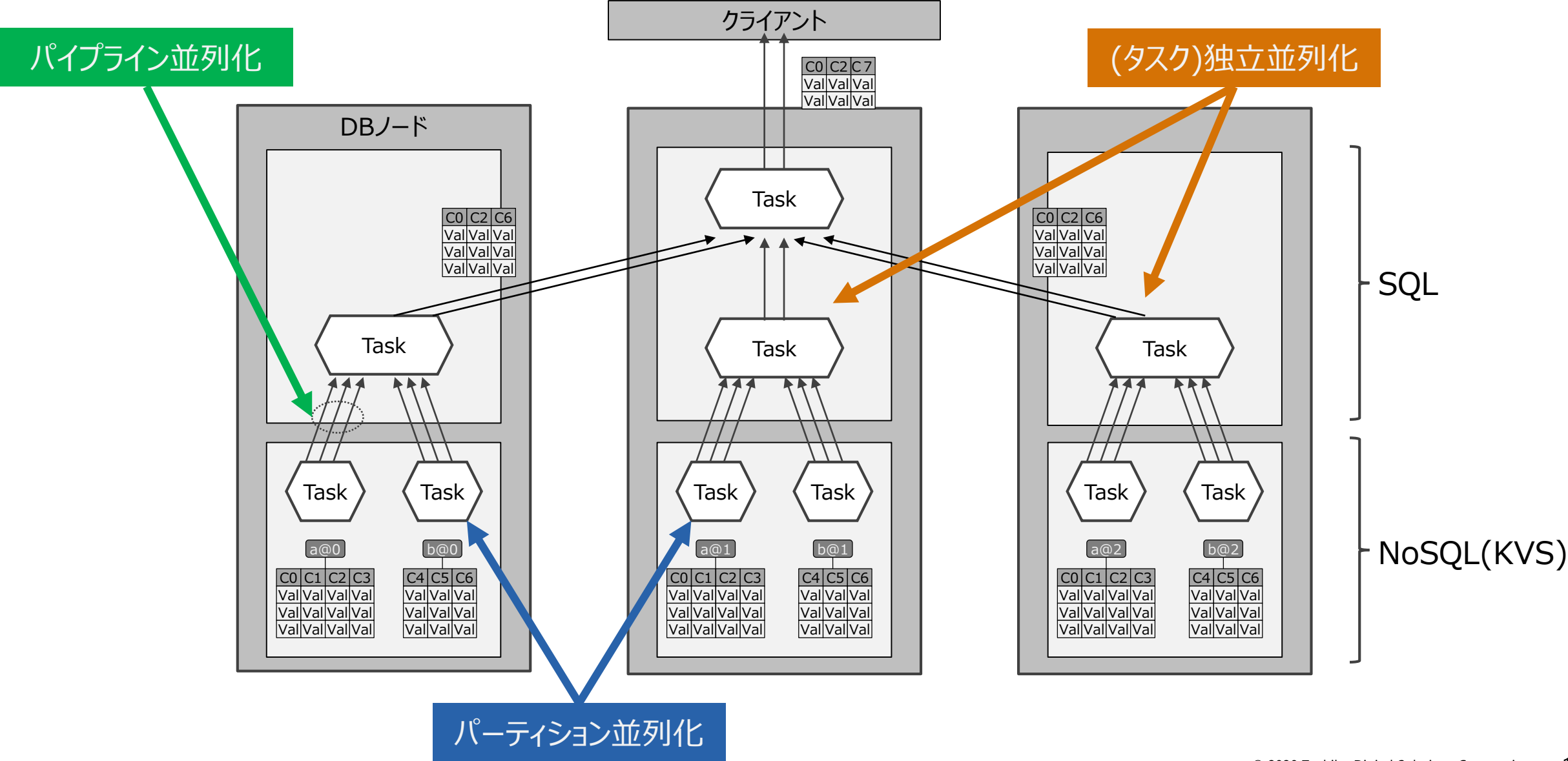
- 分散並列SQLデータベース
- 巨大コンテナに対するコンテナパーティショニング
- ジョインなど複数コンテナ(テーブル)に対するSQL
- JDBC/ODBCドライバー

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 API

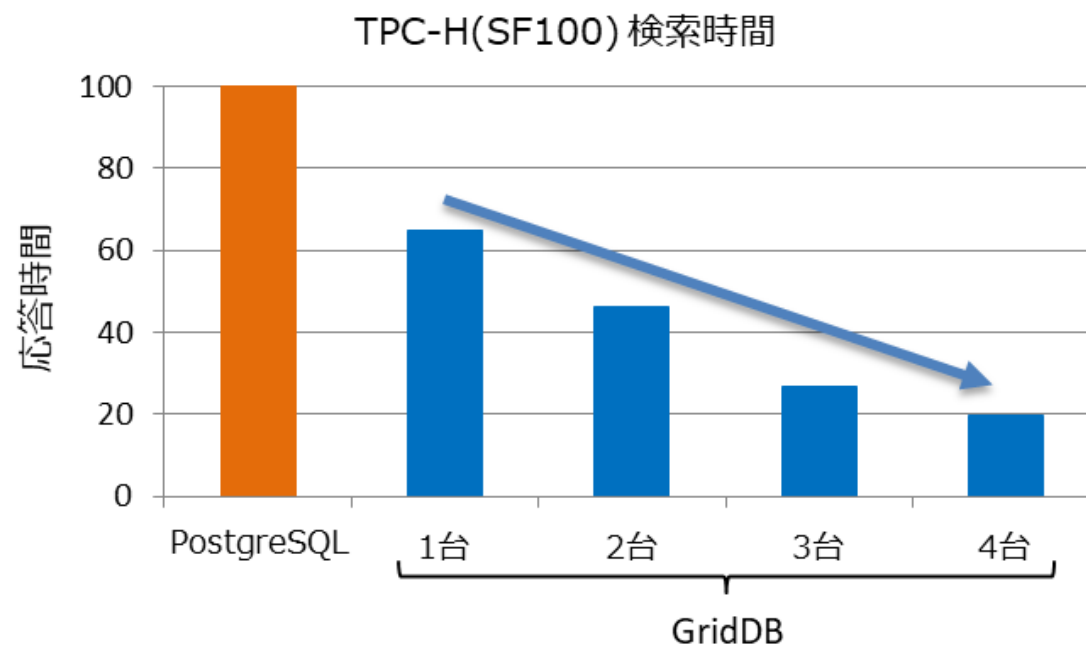
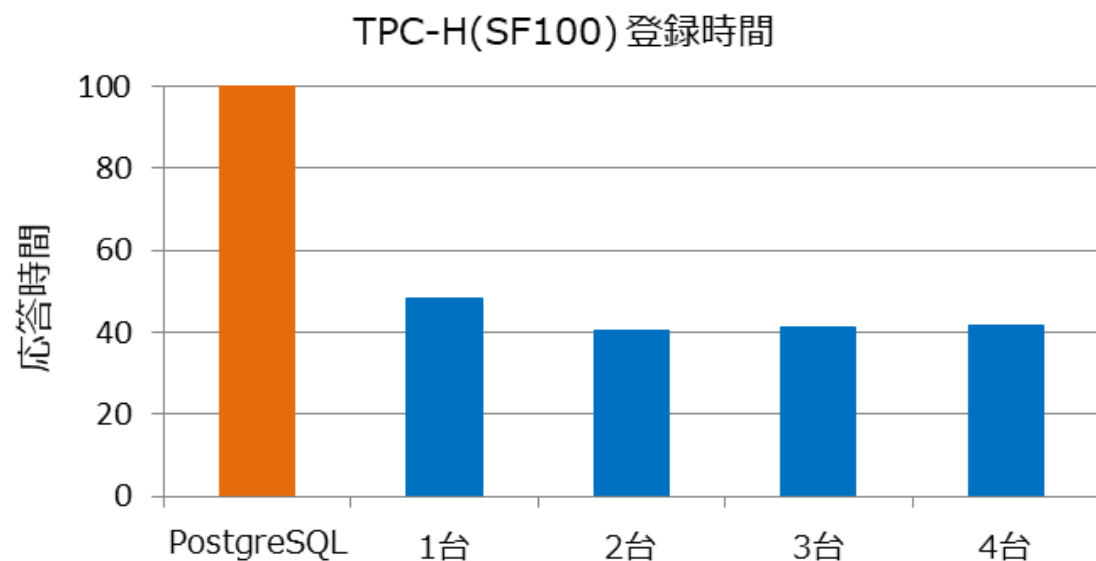
- **データ定義言語 (DDL)** : CREATE(DROP) DATABASE/USER/TABLE/INDEX/VIEW
- **データ制御言語 (DCL)** : GRANT/REVOKE/SET PASSWORD
- **データ操作言語 (DML)** : SELECT/INSERT/UPDATE/DELETE
- **句**
 - FROM/GROUP BY/HAVING/ORDER BY/WHERE/LIMIT/OFFSET
 - JOIN
 - 内部結合 [NATURAL] [INNER] JOIN
 - 左外部結合 [NATURAL] LEFT [OUTER] JOIN
 - クロス結合 [NATURAL] CROSS JOIN
 - UNION/INTERSECT/EXCEPT
- **演算子、関数、CASEなど**

SQLにおける分散並列処理



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

主な適用事例

- 社会インフラ、製造業を中心に、高い信頼性・可用性が求められるシステムに適用

- 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
- 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
- HDD製造会社 品質管理システム
製造装置のセンサーデータを長期に渡って蓄積・分析し、品質分析・改善に適用
- 半導体製造ライン 履歴管理システム
製造履歴や品質履歴、材料データなどを横串で分析し、製品の品質管理やトレーサビリティに利用
- 半導体製造ライン 異常検知システム
製造ラインのセンサーデータをリアルタイムにAIで分析し、製造ラインの異常を検知
- デンソー ファクトリー IoT
工場のDigitalTwinを実現し、生産性向上
- DENSO International Americaの次世代の車両管理システム
車両の各種センサーデータを用いる車両管理システムのPoC
-

- **NoSQLとSQLのデュアルインターフェースの提供**

- **SQLインタフェース**

- 従来のNoSQLインタフェースに加え、**JDBCドライバ**にてSQLを用いてデータベースにアクセス可能
 - Group Byや異なるテーブル間のJoinなど利用可能

- **テーブルパーティショニング**

- データ登録数が多い巨大なテーブルのデータを分散配置することで、プロセッサの並列実行を可能とし、巨大テーブルのアクセスを高速化するための機能

※JDBCドライバ：Javaアプリケーションからデータベース操作を行う標準API

02

GridDB V4.5CEのSQLインタフェース(ハンズオン)

https://github.com/konomura/griddb-docker/blob/master/README_ja.md

今回のハンズオン内容

①Java環境によるSQLインターフェースの利用

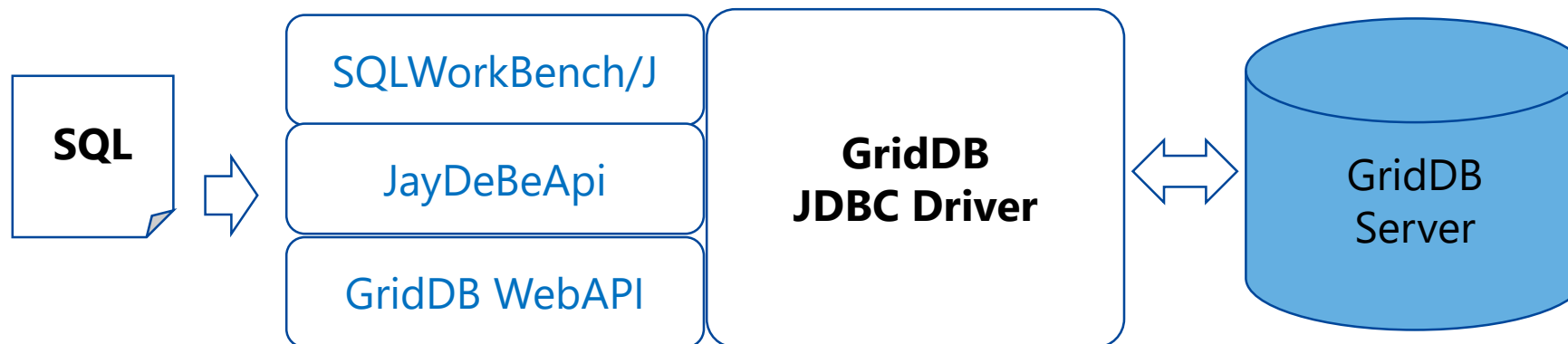
- SQLWorkbench/J (<https://www.sql-workbench.eu/>)
 - Java環境 & JDBCドライバ経由で動くSQL実行ツール
- ※特定の国の政府機関は利用できない制限があるが、日本は利用制限に含まれていない

②Python環境によるSQLインタフェースの利用

- JayDeBeApi (<https://github.com/baztian/jaydebeapi>)
 - Python標準のデータベースAPIの仕様であるDB-API(PEP 249)の1実装
- <https://www.python.org/dev/peps/pep-0249/>

③WebAPIによるSQLインタフェースの利用

- GridDB WebAPI ※SQLはSELECT文のみ



主な流れ

1. GridDBサーバのインストール&起動
2. 実行環境のインストール&起動
3. GridDBサーバへの接続
4. SQL実行
 1. テーブルの作成
 2. データの登録
 3. 検索

GridDB JDBCの接続方法

- **Jarファイル :**

- gridstore-jdbc.jar

- **ドライバクラス :**

- com.toshiba.mwcloud.gs.sql.Driver

※NoSQLインターフェースとの違い :

•デフォルトのportNo : 41999(SQL), 31999(NoSQL)

- **接続時のURL :**

(マルチキャスト方式でクラスタ内のノードに自動接続の場合)

- jdbc:gs://(multicastAddress):(portNo)/(clusterName) [/(databaseName)]

(マルチキャスト方式でノード指定の場合)

- jdbc:gs://(nodeAddress):(portNo)/(clusterName) [/(databaseName)]

※GridDB JDBCドライバ説明書 https://github.com/griddb/docs-ja/blob/master/manuals/GridDB_JDBC_Driver_UserGuide/toc.md

※GridDB SQLリファレンス https://github.com/griddb/docs-ja/blob/master/manuals/GridDB_SQL_Reference/toc.md

SQL (テーブルの作成)

装置

id	type	floor	room_no
コレクションテーブル			

```
CREATE TABLE equipTable (  
  id INTEGER PRIMARY KEY, -- 装置ID  
  type STRING, -- 装置タイプ  
  floor INTEGER, -- 設置階  
  room_no INTEGER -- 設置ルームNo  
);
```

アラート履歴

date	id	alertLevel	detail
時系列テーブル			
インターバルパーティション：分割幅30日			
パーティション解放：60日			

```
CREATE TABLE alertTable (  
  date TIMESTAMP PRIMARY KEY, -- 検知時刻  
  id INTEGER, -- 装置ID  
  alertLevel INTEGER, -- アラートレベル  
  detail STRING -- 詳細メッセージ  
) USING TIMESERIES WITH (  
  expiration_type='PARTITION',  
  expiration_time=60,  
  expiration_time_unit='DAY'  
) PARTITION BY RANGE (date) EVERY (30, DAY);
```

SQL (データの登録)

装置

<u>id</u>	type	floor	room_no
1	CAMERA	1	1
2	THERMO	1	1
...			

```
INSERT INTO equipTable VALUES(1, 'CAMERA', 1, 1);
INSERT INTO equipTable VALUES(2, 'THERMO', 1, 1);
INSERT INTO equipTable VALUES(3, 'THERMO', 4, 3);
INSERT INTO equipTable VALUES(4, 'THERMO', 6, 2);
INSERT INTO equipTable VALUES(5, 'WATT', 1, 1);
INSERT INTO equipTable VALUES(6, 'WATT', 6, 1);
```

アラート履歴

<u>date</u>	id	alertLevel	detail
2020-10-01T10:30:00Z	1	1	xxx
2020-10-03T12:10:00Z	2	2	xxx
...			

```
INSERT INTO alertTable
VALUES(TIMESTAMP('2020-10-01T10:30:00Z'), 1, 1, 'xxx');
INSERT INTO alertTable
VALUES(TIMESTAMP('2020-10-03T12:10:00Z'), 2, 2, 'xxx');
...
```

SQL (検索)

- **全件**

```
SELECT * FROM equipTable;  
SELECT * FROM alertTable;
```

- **JOIN、GROUP BY**

```
SELECT equipTable.id, type, floor, room_no, max FROM equipTable JOIN  
(SELECT id, MAX(alertLevel) AS max FROM alertTable WHERE date >= TIMESTAMP('2020-10-05T00:00:00Z')  
AND date < TIMESTAMP('2020-10-06T00:00:00Z') GROUP BY id) t  
ON equipTable.id = t.id AND max > 1;
```

- **ORDE BY CASE**

```
SELECT * FROM equipTable WHERE floor >= 3 ORDER BY CASE type  
    WHEN 'CAMERA' THEN 1  
    WHEN 'THERMO' THEN 2  
    WHEN 'WATT' THEN 3  
    WHEN 'WIFI' THEN 4  
    ELSE 5  
END;
```

03

GridDBのOSS活動



- **GridDBをGitHub上にソース公開(2016/2)**

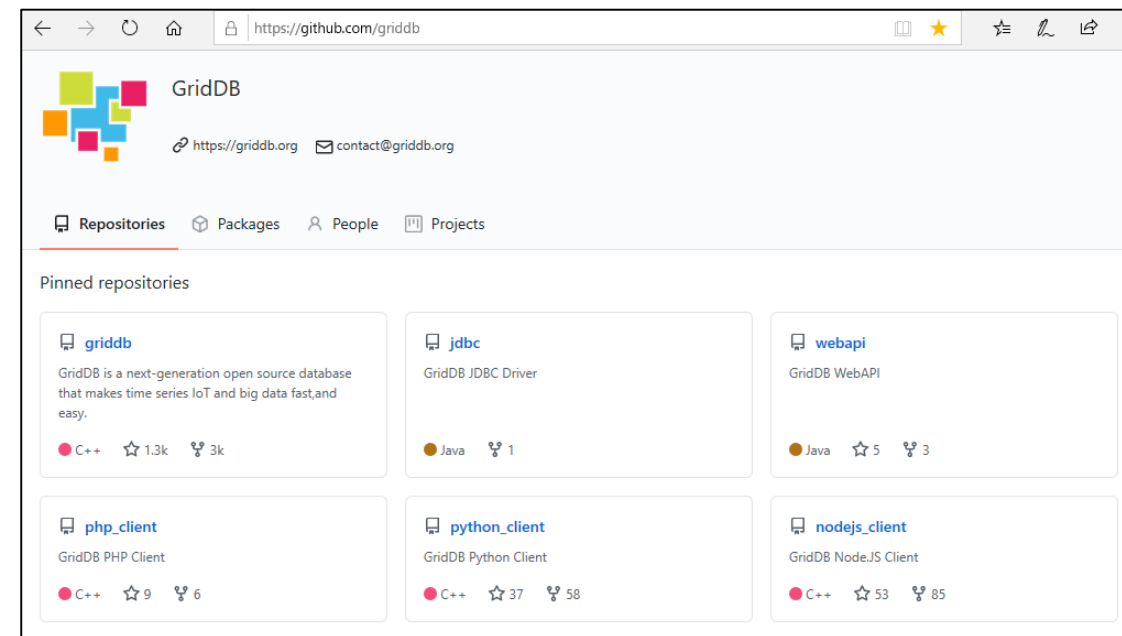
<https://github.com/griddb/griddb>

- **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

- **ライセンス**

- サーバはAGPL-3.0
- 各種開発言語のクライアント、OSSとのコネクタはApache-2.0



主なOSS活動

- ① **GridDB本体の機能強化**
- ② **主要OSSとの連携強化**
- ③ **APIの拡充**
- ④ **GitHub以外のサイトからの情報発信**
 - パッケージ
 - デベロッパーズサイト（WP、ブログなど）・・・フィックスターズ社
 - SNS・・・フィックスターズ社
- ⑤ **主要OSSリポジトリへのコントリビュート**
- ⑥ **プラットフォームの拡充**
- ⑦ **その他**
 - OSCなどカンファレンス参加
 - ハンズオン無料セミナー・・・（株）アイ・ティ・イノベーション

OSS活動の全体イメージ

性能測定

収集

分散処理

可視化

Webアプリ

分析

AI/機械学習 ...

④ GitHub以外のサイトからの情報発信

PyPI/npm/Maven/Packagist/...

② 主要OSSとの連携強化

Spark
コネクタ

Fluentd/Grafana/Redash
プラグイン

YCSB
コネクタ

Kafka
コネクタ

Hadoop
MapReduce
コネクタ

WebAPI

Python/Node.JS/Go/PHP/Ruby/Perlクライアント

③ APIの拡充

Javaクライアント

JDBCドライバ

Cクライアント

① GridDB本体の機能強化

GridDB V4.5 CE(Community Edition)

⑤ 主要OSSリポジトリへのコントリビュート

⑥ プラットフォームの拡充

CentOS+
Ubuntu、**openSUSE**
Windows、**MacOS**
Docker

GitHub



最近の主な活動（2020年）

2020年

- 1月 Goクライアントのビルド環境見直し
- 2月 Python/Node.JS/Go/PHPクライアントのサンプル追加
サーバのopenSUSE版のRPM公開(openSUSE Build Service(OBS))
PHPクライアントのパッケージ公開(Packagist)
- ~~• 4月 Goクライアントのビルド環境見直し~~
- 5月 Dockerイメージ公開(GitHub Packages)
- 6月 Fluentdプラグインのセキュリティ対応(GitHubからの自動PRの反映)
V4.5CE(JDBCドライバなど)のソース公開
- 7月 V4.5CEのパッケージ公開(Maven Central Repository, OBS)
- 8月 PHPクライアントI/F改訂版のソース公開
WebAPI(SQLインタフェース対応版)のソース公開
- 9月 Pythonクライアント(多カラム&Pandas向け高速版)のソース公開
Cクライアント(MacOS版)のソース公開

- 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
 - ホワイトペーパー
 - ブログ
- など



- 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)



05

まとめ

まとめ

- GridDBはビッグデータ・IoT向けのデータベースです。
- 最新版V4.5CEのSQLインタフェースとオープンソース活動をご紹介しました。
 - 今後も様々な拡張、拡充を進めて参ります。

GridDBのオープンソース版(GridDB CE)を是非とも使ってみてください。

<https://github.com/griddb/>

ご参考 : GridDBに関する情報

- **GridDB GitHubサイト**

- <https://github.com/griddb/griddb/>

griddb github	検索
---------------	----

- **GridDB デベロッパーズサイト**

- <https://griddb.net/>

griddb net	検索
------------	----

- **Twitter: GridDB (日本)**

- https://twitter.com/griddb_jp

griddb jp	検索
-----------	----

- **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**

- OSS版の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

- プログラミング関連以外 : contact@griddb.netもしくはcontact@griddb.orgをご利用ください



TOSHIBA