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 4.6のテーブルパーティショニング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

2021.10.22

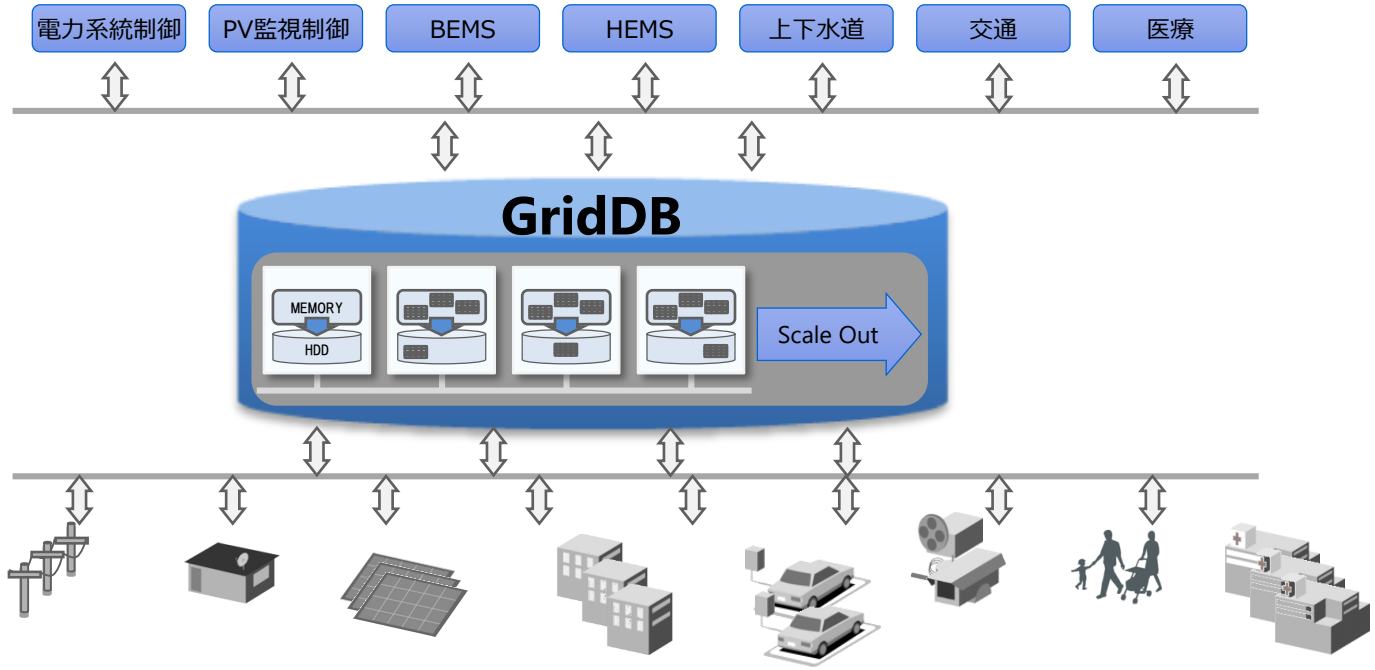
Contents

01

IoT向けデータベースGridDBの概要

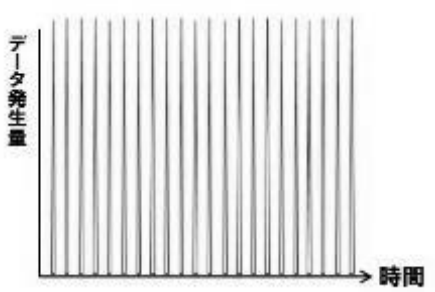
IoT向けデータベースGridDB

- 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)、V4.6CE(2021年2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



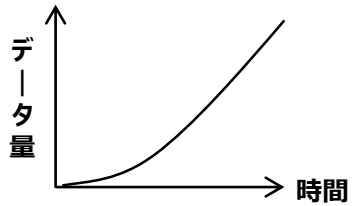
IoTデータの特長

ミリ秒オーダーで
高頻度に発生



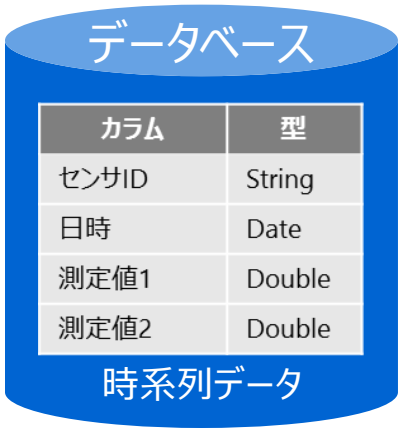
24H365D
絶え間なく発生

大量データが
単調増加

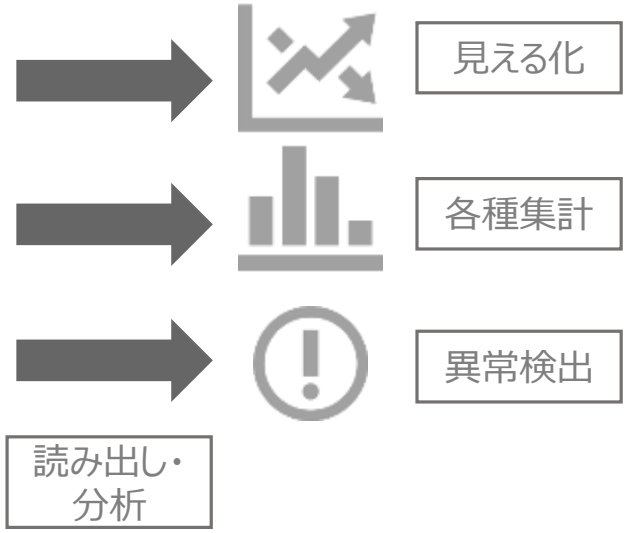


発生直後から
リアルタイム参照

収集



蓄積



読み出し・
分析

見える化

各種集計

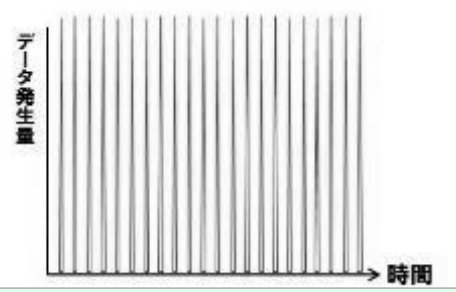
異常検出

データー貫性
の保証

データベースへの要求

ミリ秒オーダーで
高頻度に発生

高い処理能力
High Performance



高い信頼性
High Reliability

24H365D
絶え間なく発生

大量データが
単調増加

高い拡張性
High Scalability

センサー
APL ログ
Web・SNS ログ
Web サイト
CRM/ERP

時間

データベース

カラム	型
センサID	String
日時	Date
測定値1	Double
測定値2	Double

収

IoT指向
IoT Oriented

データー貫性
の保証

発生直後から
リアルタイム参照

高い処理能力
High Performance

読み出し・
分析

見える化

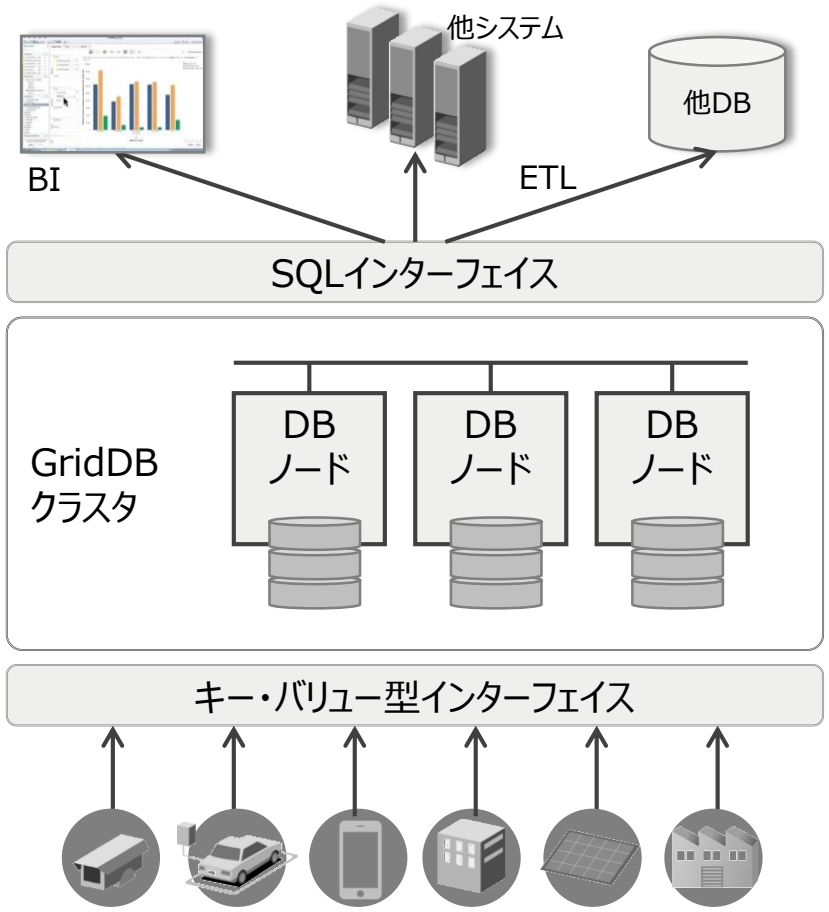
各種集計

異常検出

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
- 巨大テーブルに対するテーブルパーティショニング
- ジョインなど複数テーブルに対するSQL
- JDBC/ODBCドライバー

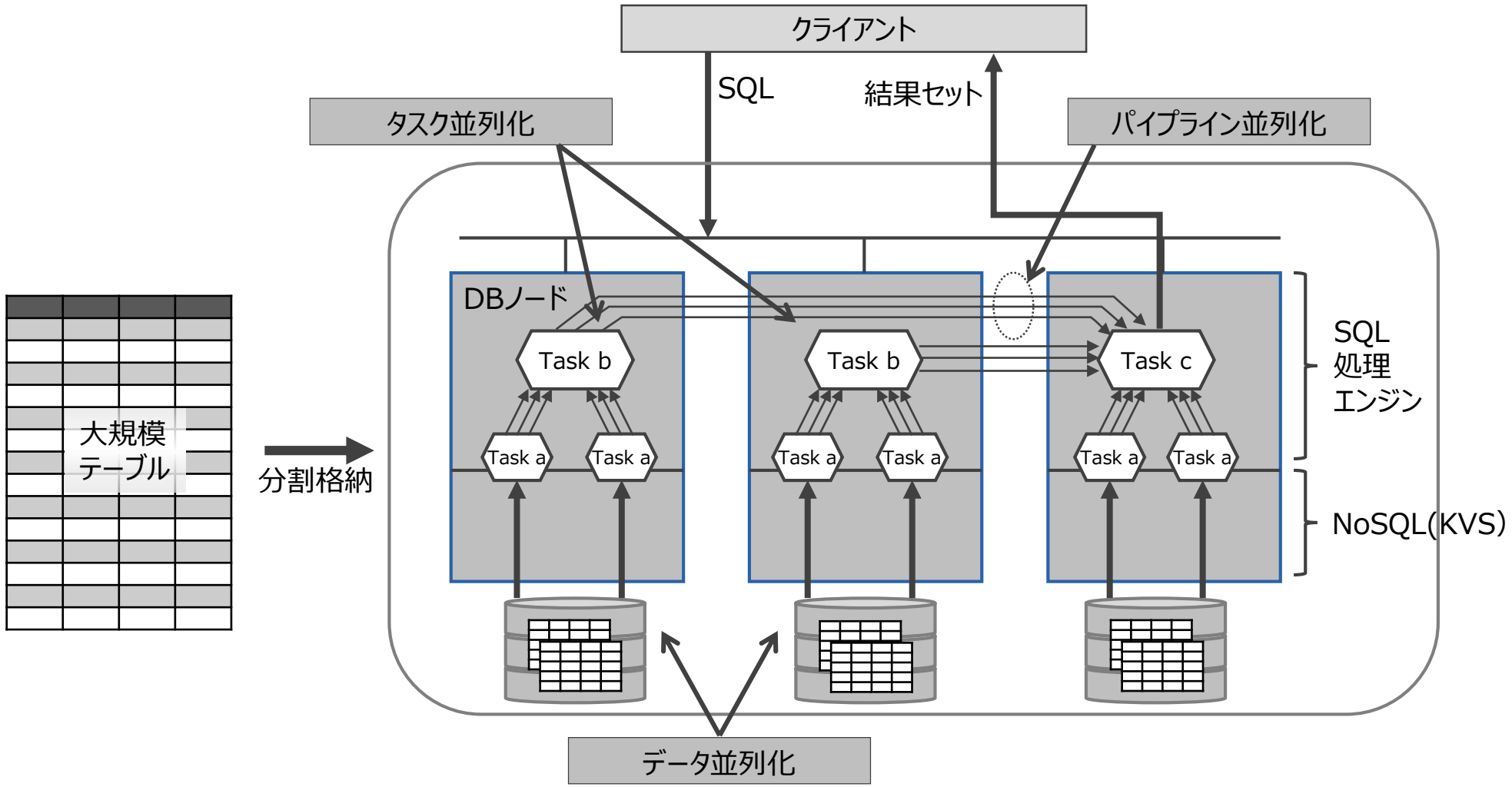
NoSQL(キー・バリュー型)インターフェイス

- 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

GridDB SQL

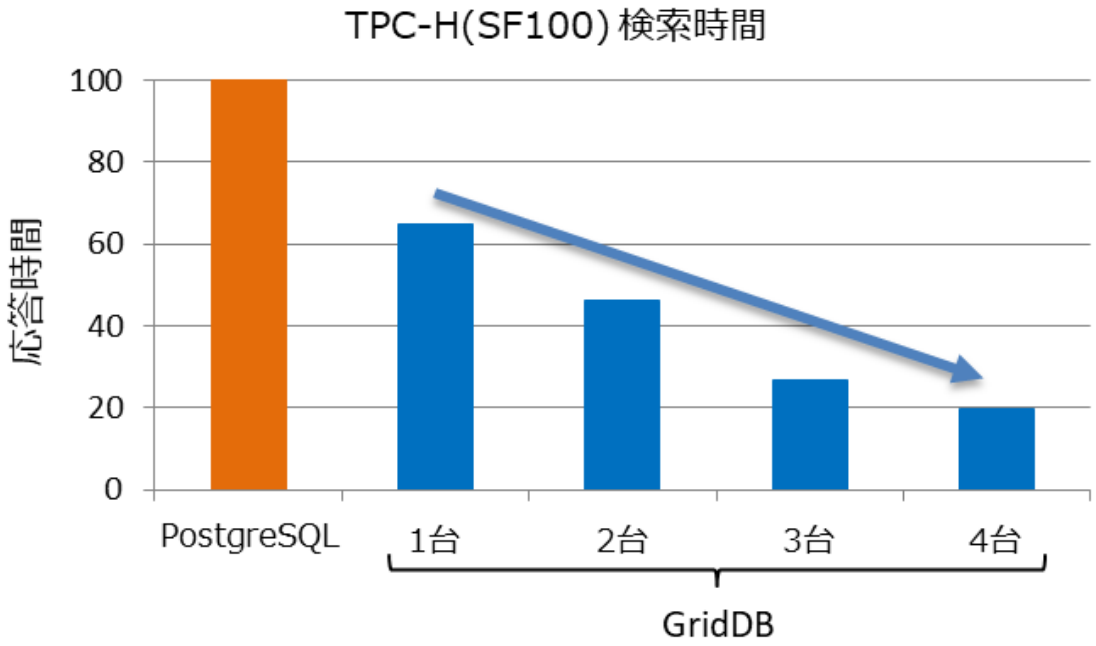
- **データ定義言語（DDL）：**CREATE(DROP) DATABASE/USER/TABLE/INDEX/VIEW
- **データ制御言語（DCL）：**GRANT/REVOKE/SET PASSWORD
- **データ操作言語（DML）：**SELECT/INSERT/UPDATE/DELETE
- **句**
  - FROM/GROUP BY/HAVING/ORDER BY/WHERE/LIMIT/OFFSET
  - JOIN
    - 内部結合　[NATURAL] [INNER] JOIN
    - 左外部結合 [NATURAL] LEFT [OUTER] JOIN
 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

テーブルパーティショニングと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
….

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**

- **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

- ハッシュパーティショニング
  - ✓ 選択基準：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
  - ✓ 選択基準：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
  - ✓ 選択基準：インターバルパーティショニングでは力不足の場合

```
-- 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
CREATE TABLE a1 (code INT, ts TIMESTAMP NOT NULL, dest STRING)
    PARTITION BY RANGE(ts) EVERY(1,DAY)
-- インターバルハッシュ
CREATE TABLE a4 (code INT NOT NULL, ts TIMESTAMP, dest STRING)
    PARTITION BY RANGE(ts) EVERY(1,DAY)
    SUBPARTITION BY HASH(dest) SUBPARTITIONS 2
```

02

GridDB V4.6CEのSQLインタフェース(ハンズオン)

https://github.com/knonomura/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

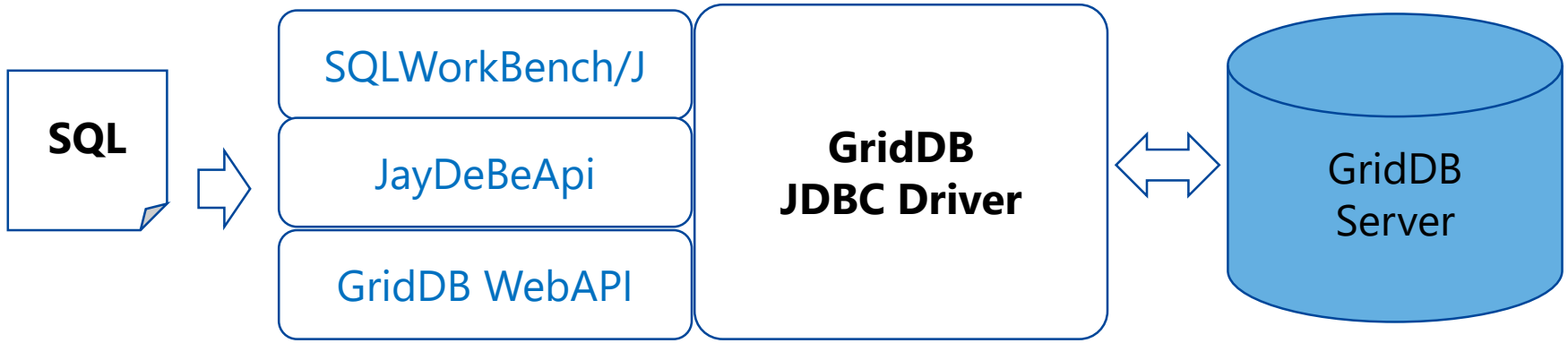
※特定の国の政府機関は利用できない制限があるが、日本は利用制限に含まれていない

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
- **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)]

※NoSQLインターフェースとの違い：
・デフォルトのportNo：41999(SQL), 31999(NoSQL)

※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
);
```

センサデータ

date	id	alertLevel	detail
コレクションテーブル **インターバルハッシュパーティション：** 分割幅30日、サブパーティション数6 パーティション解放：60日			

```
CREATE TABLE sensorTable (
  date TIMESTAMP, -- 時刻
  id INTEGER, -- 装置ID
  value DOUBLE, -- センサ値
  PRIMARY KEY(date, id)
) WITH (
  expiration_type='PARTITION',
  expiration_time=60,
  expiration_time_unit='DAY'
) PARTITION BY RANGE (date) EVERY (30, DAY);
SUBPARTITION BY HASH(id) SUBPARTITIONS 6;
```

SQL（データの登録）

装置

id	type	floor	room_no
1	CAMERA	1	1
2	THERMO	1	1
｡｡｡			

センサデータ

date	id	value
2021-11-01T10:30:00Z	2	18.5
2021-11-01T10:30:00Z	3	20.0
｡｡｡		

```
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);
```

```
INSERT INTO sensorTable
  VALUES(TIMESTAMP('2021-11-01T10:30:00Z'), 2, 18.5);
INSERT INTO sensorTable
  VALUES(TIMESTAMP('2021-11-01T10:30:00Z'), 3, 20.0);
｡｡｡
```

SQL（検索）

- **全件**

```
SELECT * FROM equipTable;
SELECT * FROM sensorTable;
```

- **JOIN、GROUP BY**

```
SELECT equipTable.id, type, floor, room_no, min FROM equipTable JOIN
  (SELECT id, MIN(value) AS min FROM sensorTable WHERE date >= TIMESTAMP('2021-11-01T12:00:00Z')
    AND date < TIMESTAMP('2021-11-01T18:00:00Z') GROUP BY id) t
  ON equipTable.id = t.id AND min >= 20.0;
```

- **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

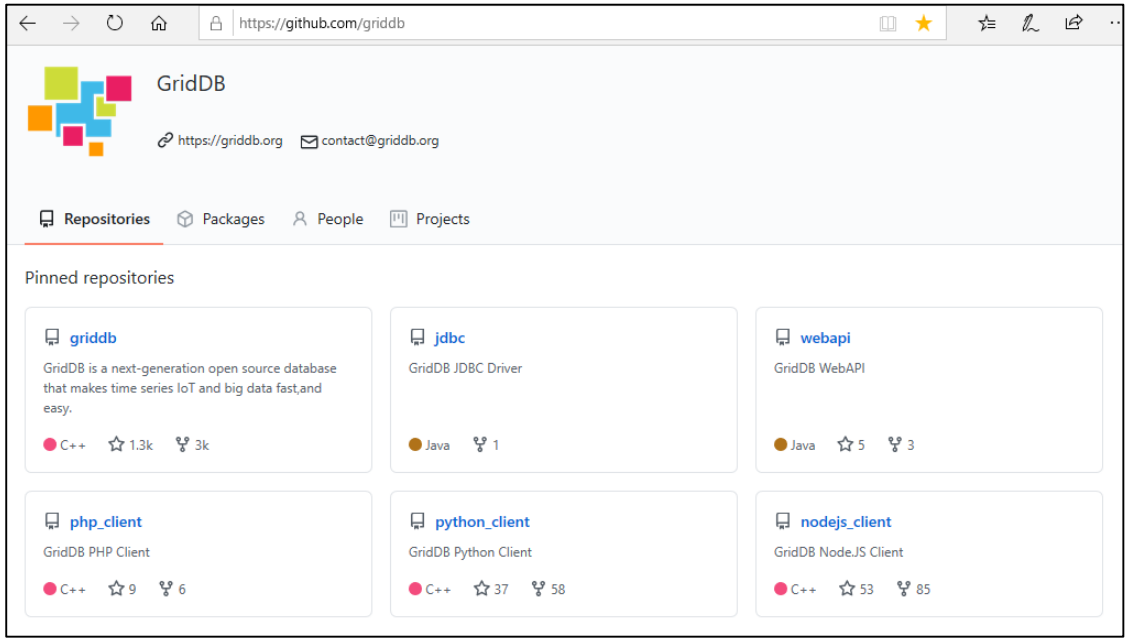
GridDBのOSS活動

GridDBのオープンソース化

https://github.com/griddb/

griddb github | 検索

- **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
- **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 …

④GitHub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

⑥プラットフォームの拡充
CentOS＋
Ubuntu、openSUSE
Windows、MacOS
Docker

WebAPI
Python/Node.JS/Go/PHP/Ruby/Perlクライアント

③APIの拡充

Javaクライアント
JDBCドライバ
Cクライアント

①GridDB本体の機能強化
GridDB V4.6 CE(Community Edition)

https://github.com/griddb/
GitHub

⑤主要OSSリポジトリへのコントリビュート

最近の主な活動（2021年）

2021年

- **2月** V4.6CEのソース公開
  CLI (Command Line Interface)のソース公開
- **3月** Node API (node-addon-api版)のソース公開
- **5月** Node API (Node.JS V16対応)
- **7月** Redash Plugin (SQL対応)
- **8月** V4.6.1CEのソース公開
  Node API (バッチ処理対応)
- **9月** Pythonクライアント (Python3.9対応)

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

  など

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)

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V4.6CEのテーブルパーティショニング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 デベロッパーズサイト**
  - https://griddb.net/
- **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**
  - 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

griddb github	検索

griddb net	検索

griddb jp	検索

TOSHIBA