TOSHIBA

オープンソースのビッグデータ・IoT向け スケールアウト型データベースGridDBとPython連携 ～ GridDBとPythonと私 ～

２０１８年２月２３日

東芝デジタルソリューションズ株式会社
野々村 克彦

プロフィール

Katsuhiko Nonomura

knonomura

Block or report user

Organizations

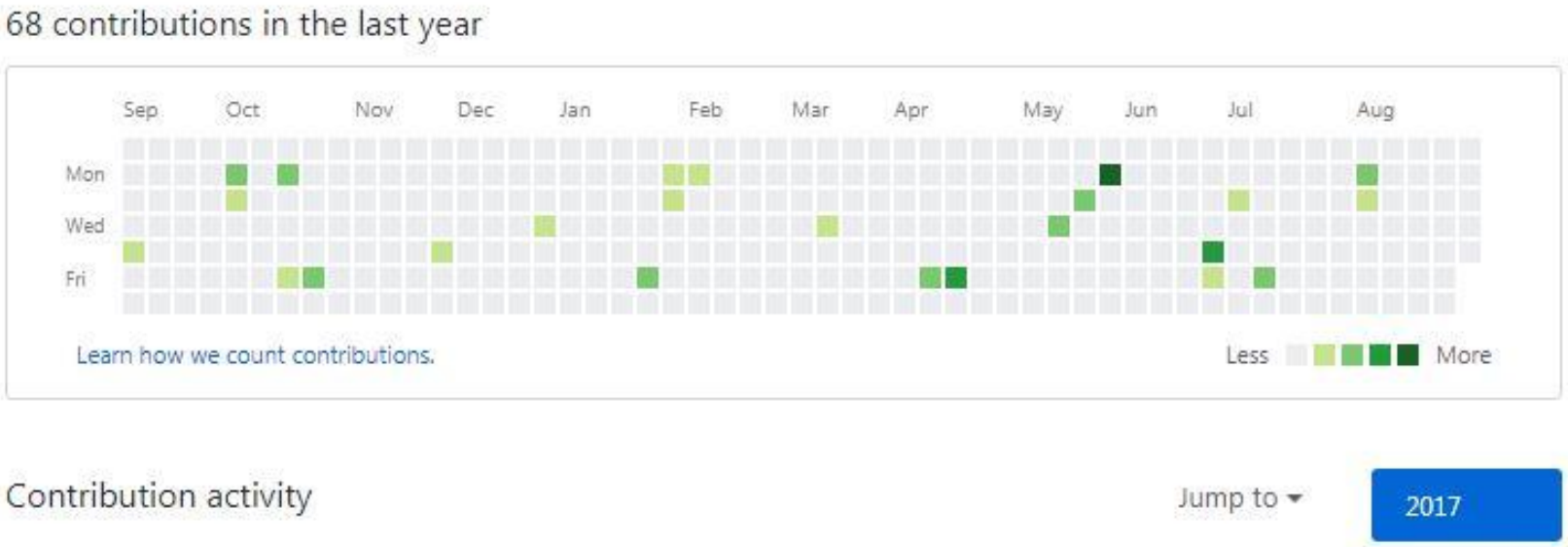
２０００年ごろ　XMLデータベースTX1開発メンバ

２０１１年　スケールアウト型DB GridDB開発メンバ

２０１５年　GridDBのオープンソースPJ開始

コミュニティ版の開発、海外展開の技術支援など

GitHub歴 3年、週末は小学3年の息子とサッカー



発表内容

1.スケールアウト型データベースGridDB

- 特長
- 性能、導入事例、Webサイト

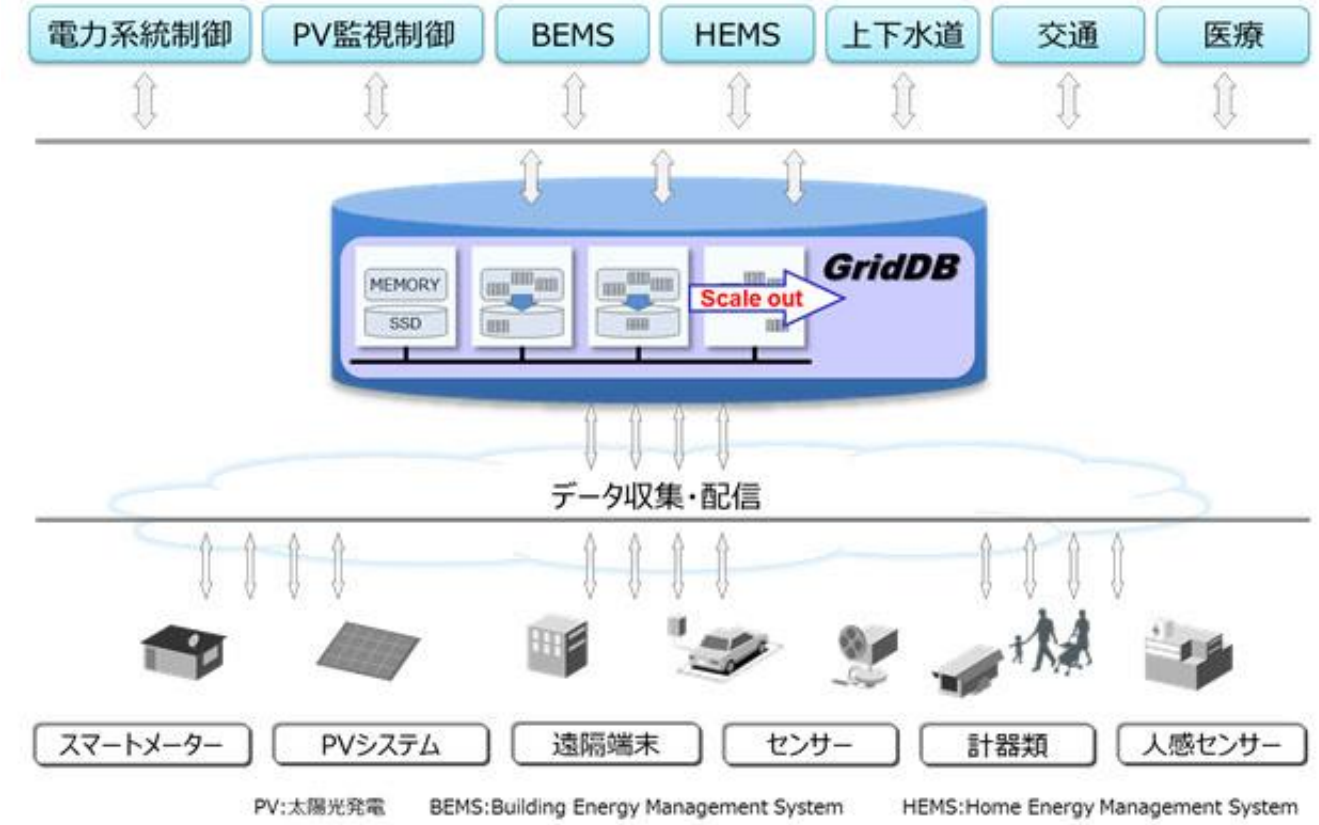
2.（私が開発した）Pythonクライアント

- これまでの開発の経緯
- 新Pythonクライアントについて
- 展開状況

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（２０１１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

①キーコンテナ型

高性能

- 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

- 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

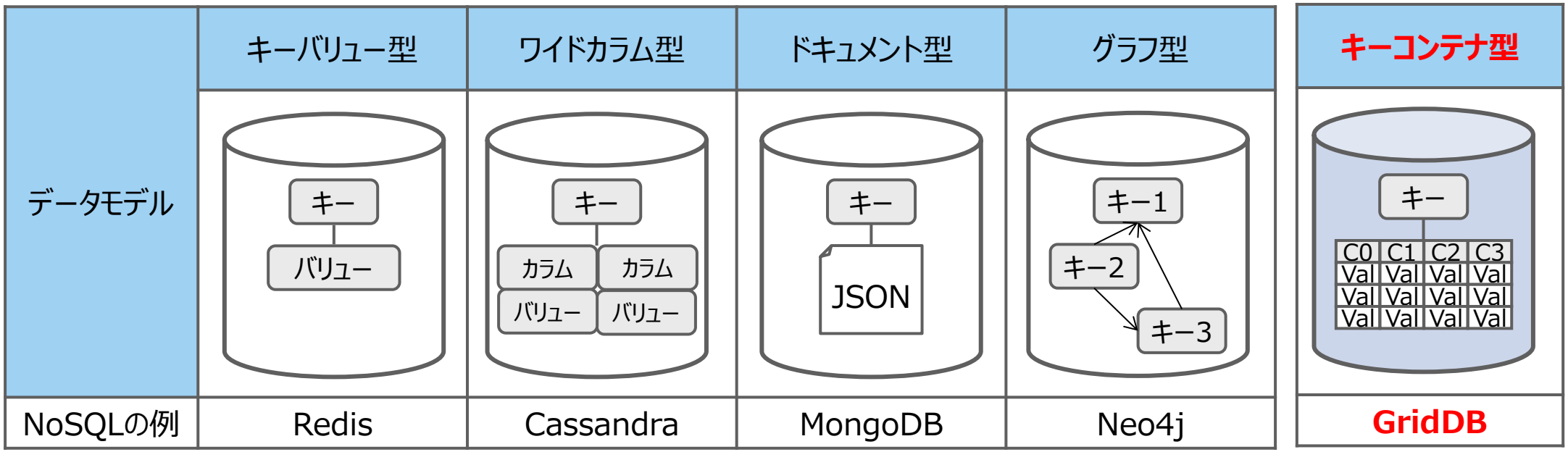
②ハイブリッド型のクラスタ管理
③ADDA

高い信頼性と可用性

- 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

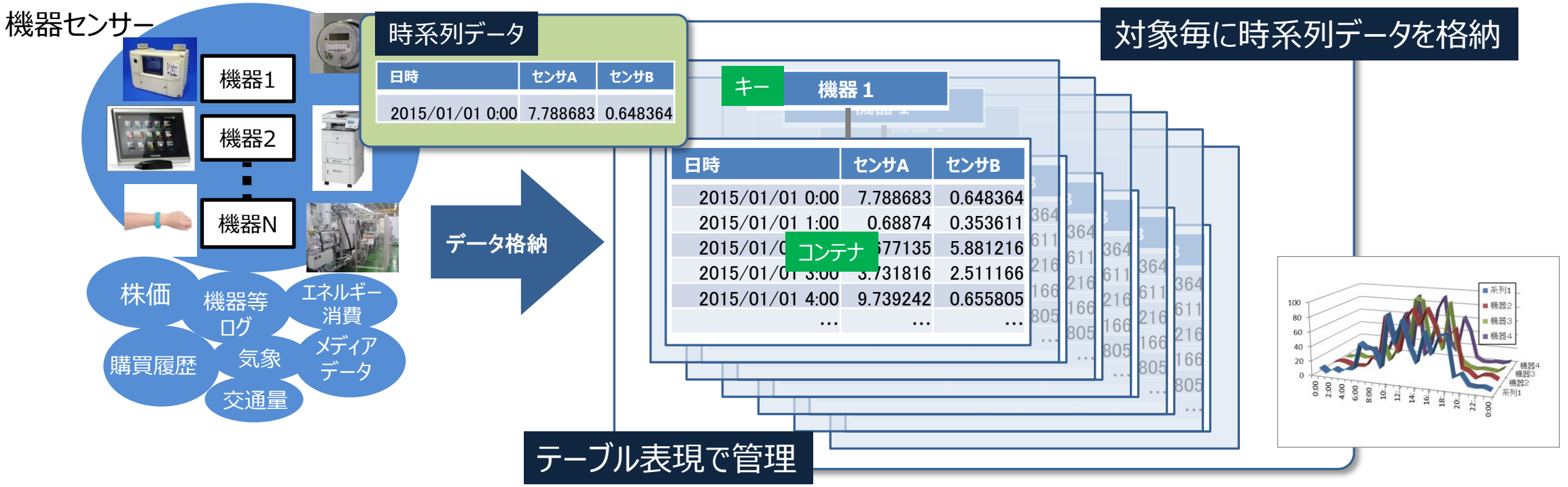
①データモデル

GridDBはキーコンテナ型

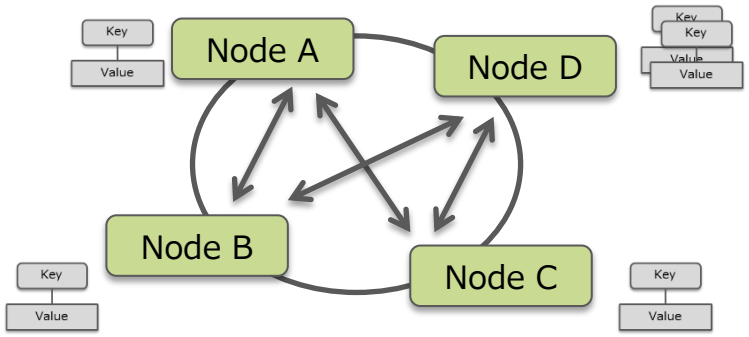
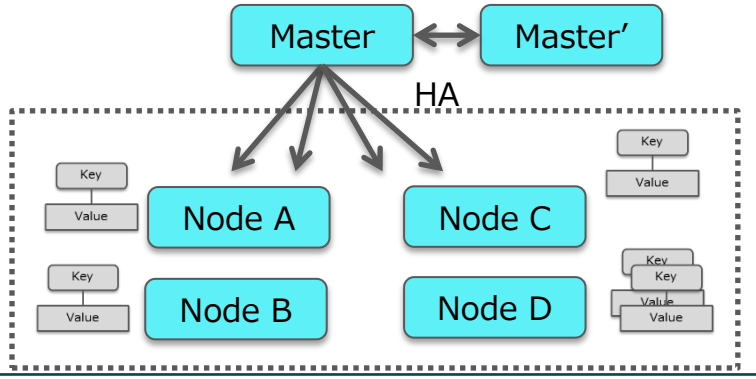


キーコンテナ型のデータモデル

- 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



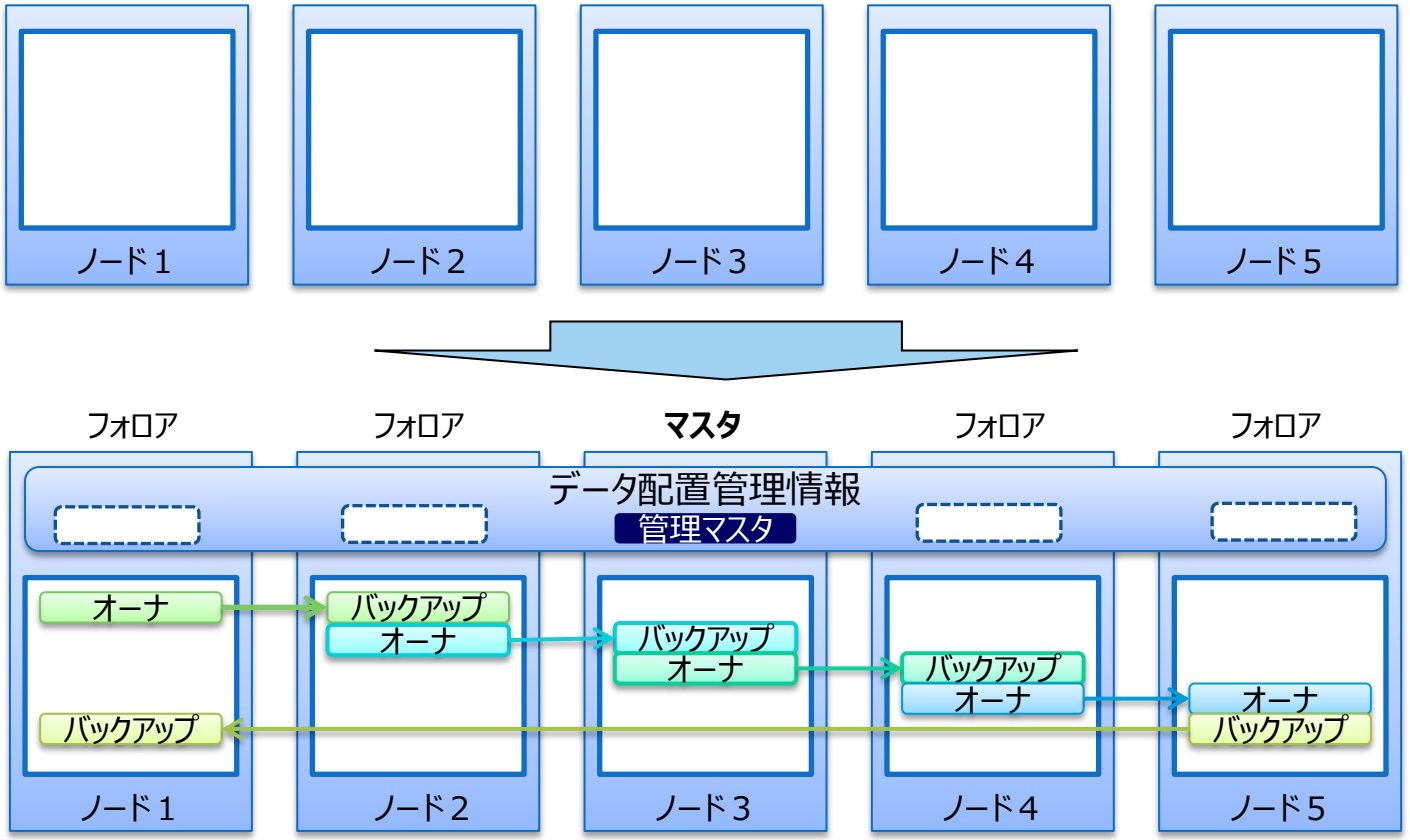
②クラスタ管理

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

ハイブリッド型のクラスタ管理

GridDBはハイブリッド型

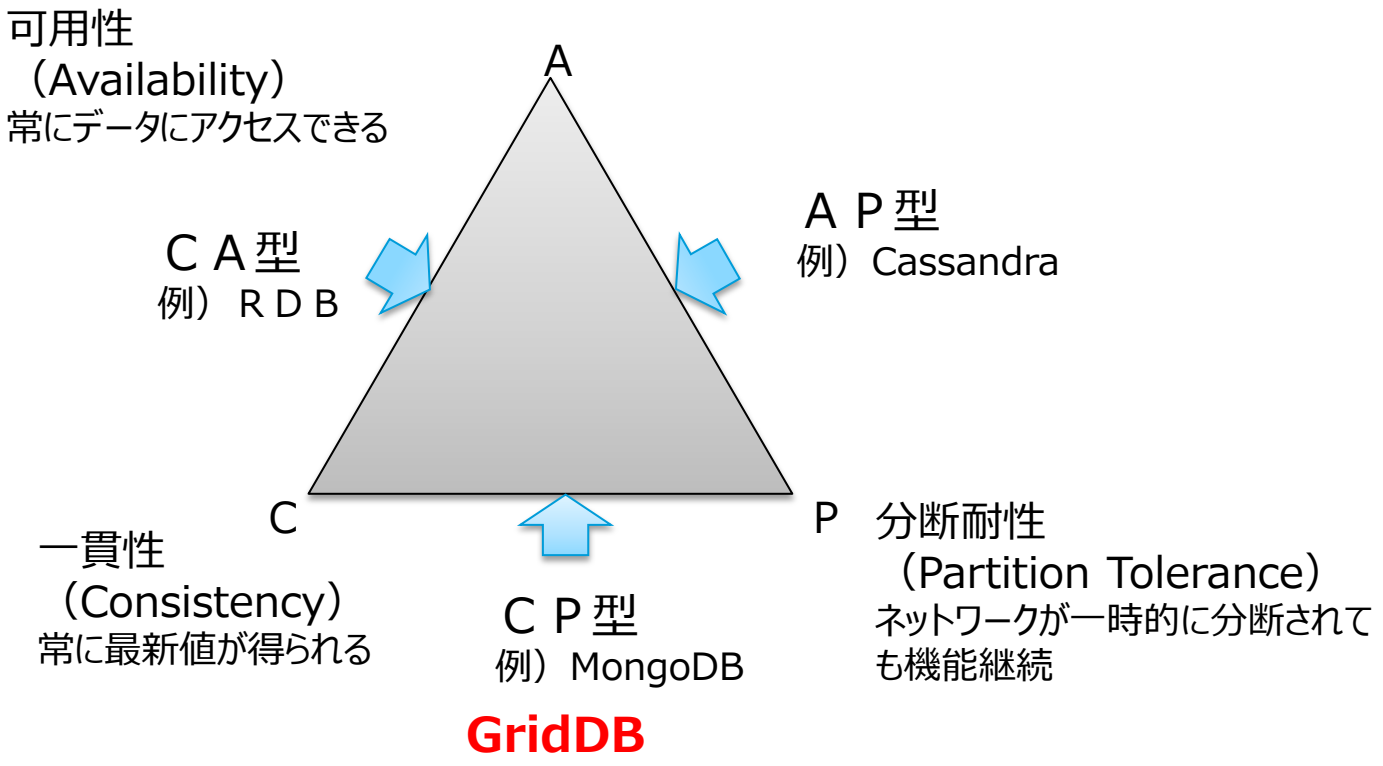
- ノード間で自律的、動的にマスタノードを決定。単一故障点（ＳＰＯＦ）を排除
- マスタがデータ配置（オーナ/バックアップ）を決定



③一貫性と可用性

GridDBはCP型

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

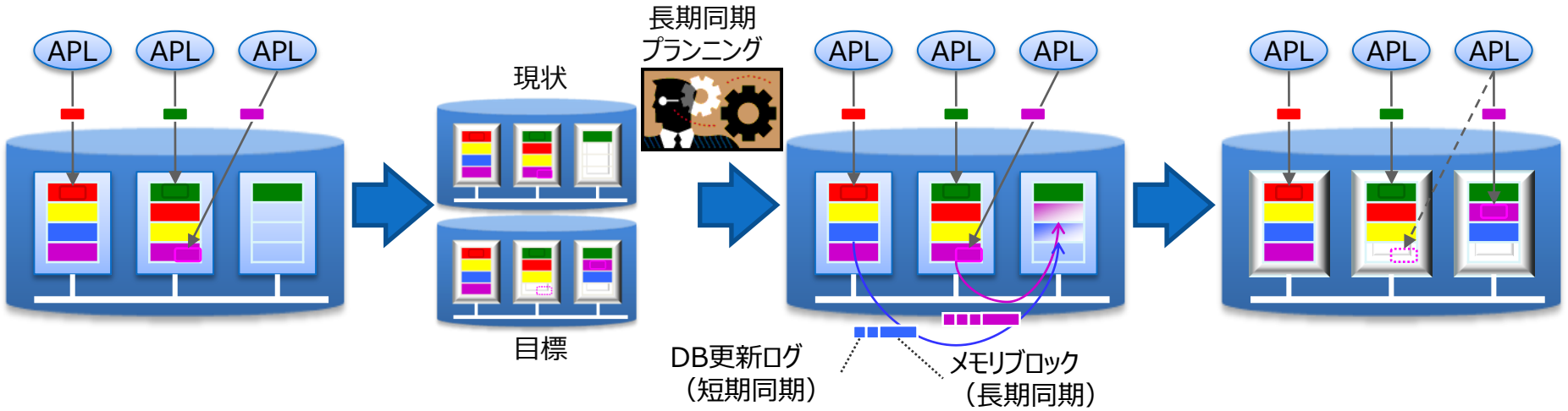


[1] Proc. 19th Ann. ACM Symp.Principles of Distributed Computing (PODC 00), ACM, 2000, pp. 7-10;

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

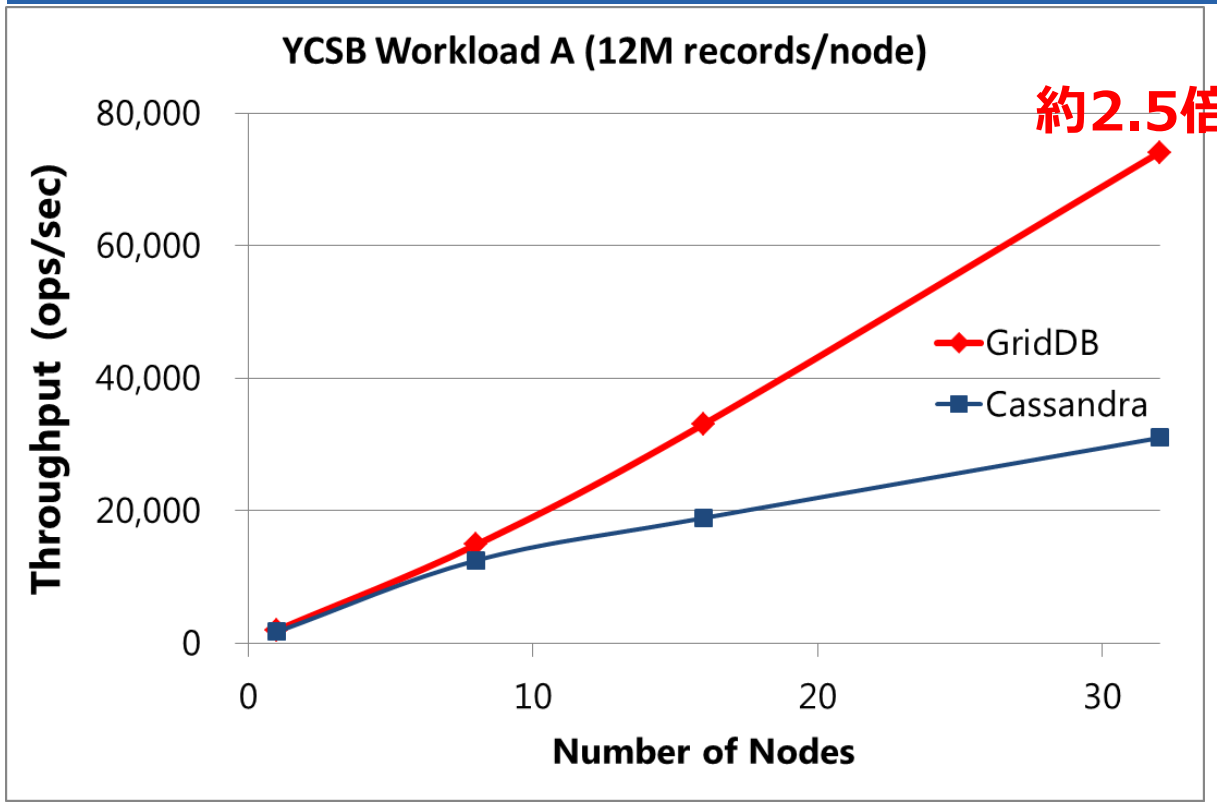


①負荷インバランス検知　②長期同期プランニング　③長期同期実行　④アクセス切替

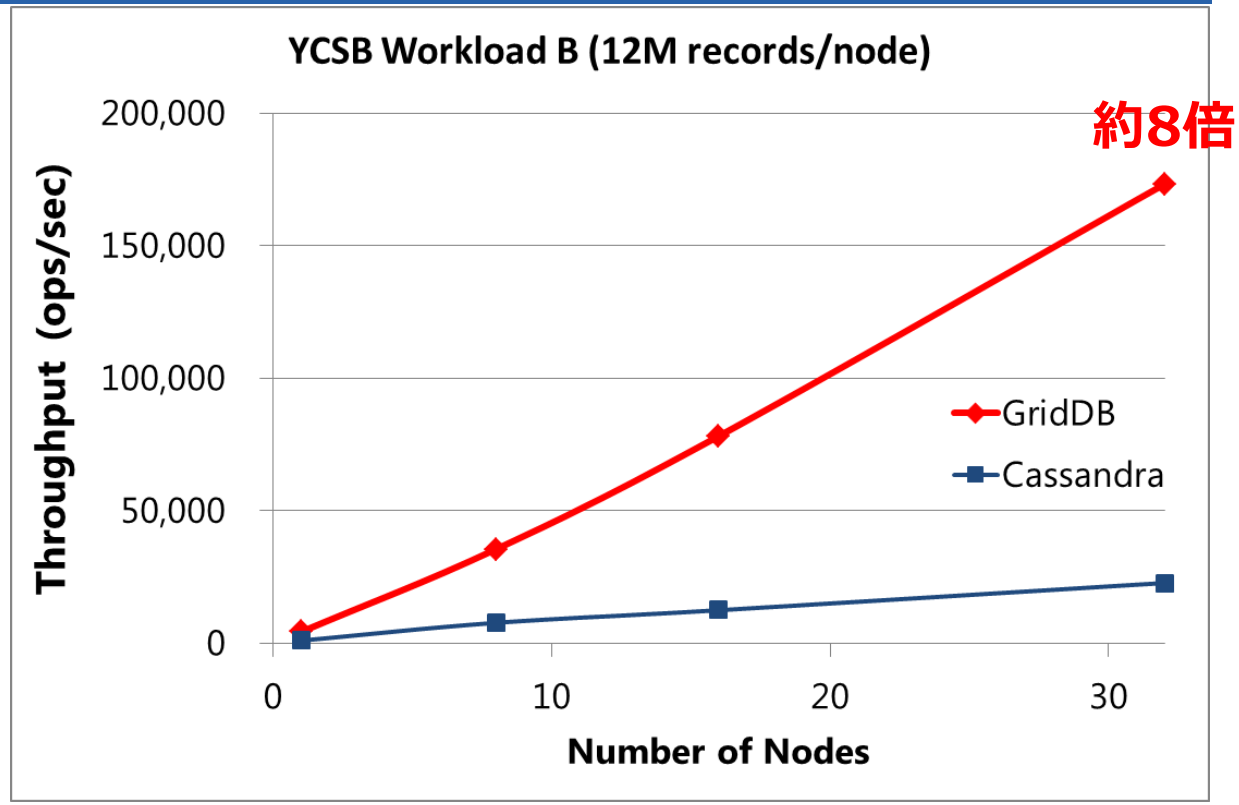
Cassandraとの性能比較（YCSB）

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

高性能を売りにするCassandraと比較しても、GridDBの方が圧倒的に高性能



Read 50% + Write 50%

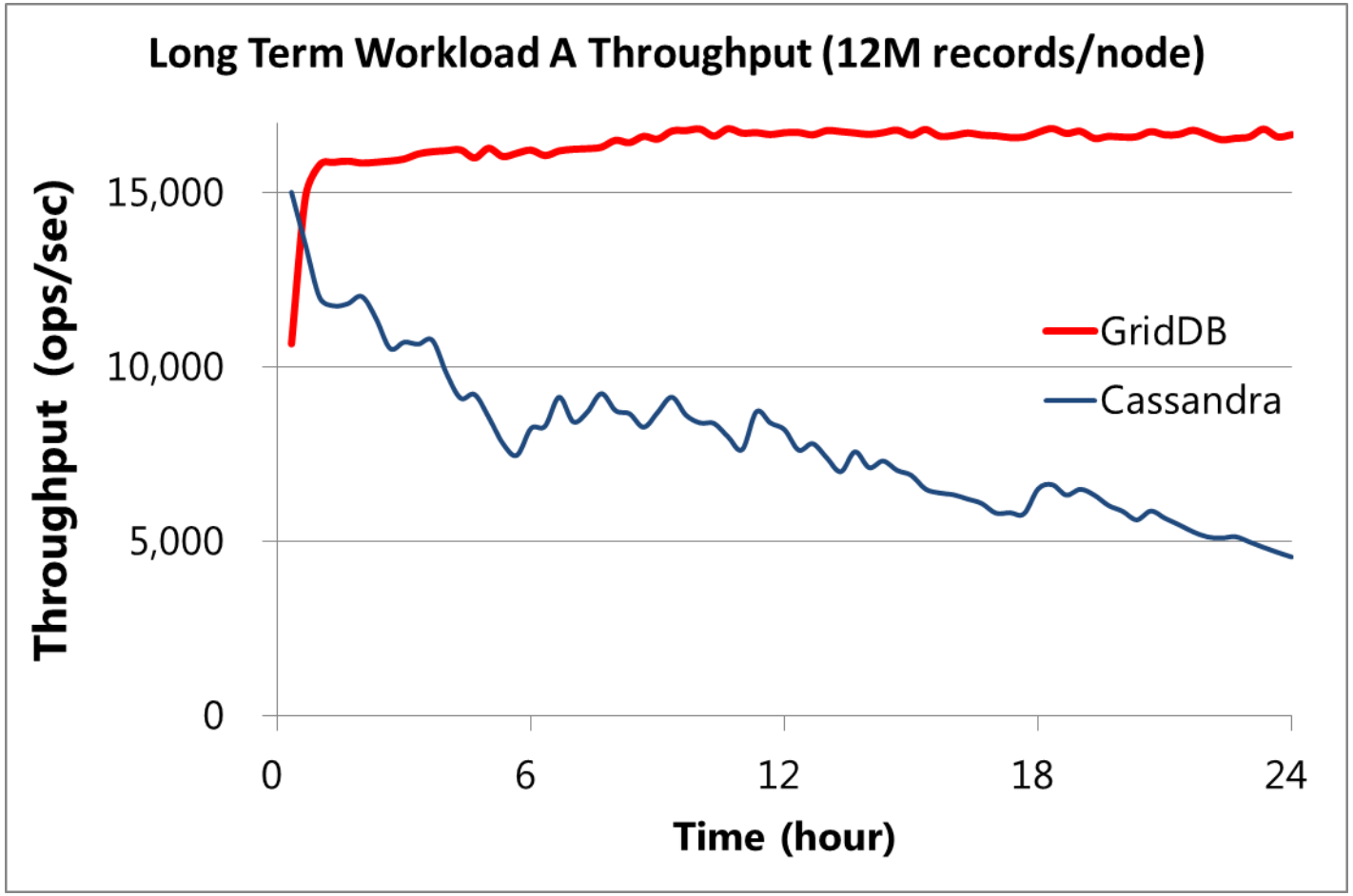


Read 95% + Write 5%

※フィックスターズ社によるYCSBベンチマーク結果

Cassandraとの性能比較（YCSB）

長時間実行してもGridDBは性能劣化が少ない



※フィックスターズ社によるYCSBベンチマーク結果

時系列DB (Time Series DBMS)

- 最近最も注目されているDBカテゴリ
  - ブログ「Time Series DBMS are the database category with the fastest increase in popularity」(2016/7/4) http://db-engines.com/en/blog_post//62
- InfluxDBが時系列DBのランキング(2018/2)でトップ1

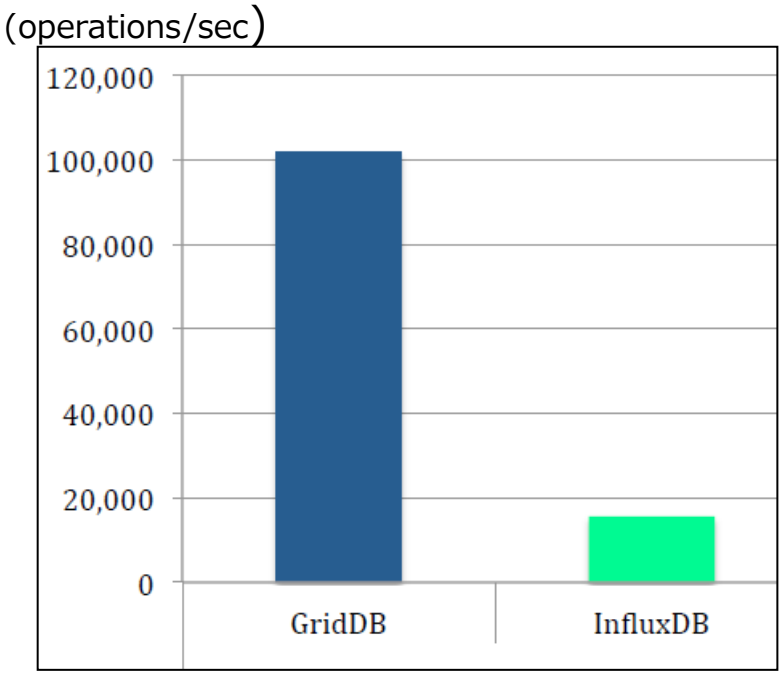
  https://db-engines.com/en/ranking/time+series+dbms

InfluxDBとの性能比較（YCSB-TS）

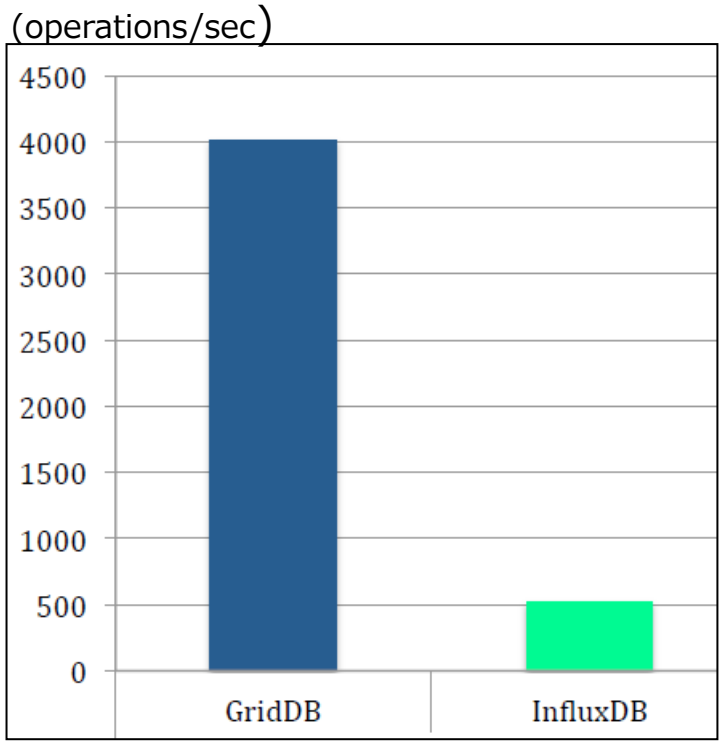
YCSB-TS：YCSBの時系列DB(Timeseries database)版のベンチマーク
https://github.com/TSDBBench/YCSB-TS

高速な時系列DB InfluxDBと比較しても、GridDBの方が圧倒的に高性能

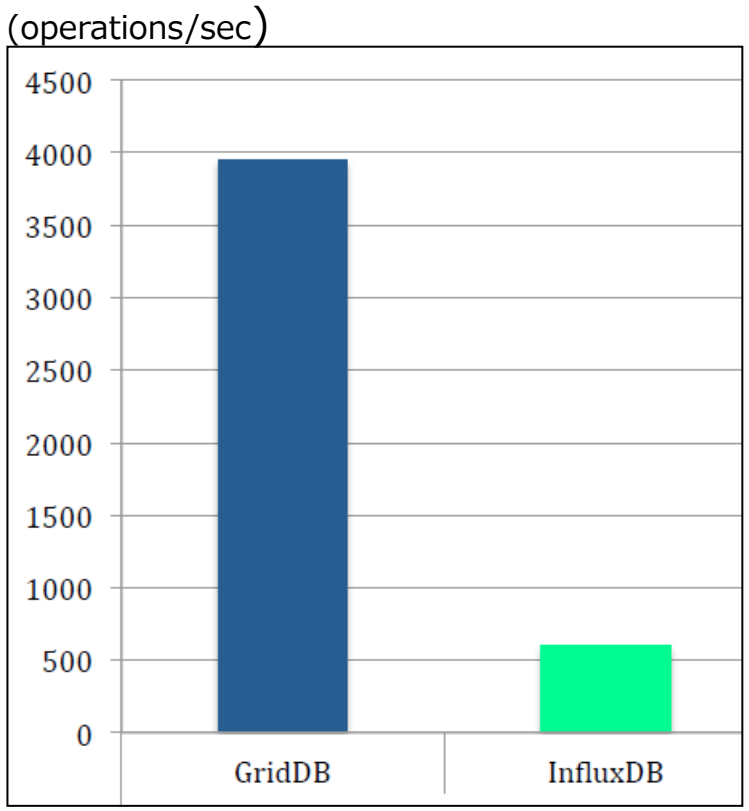
Insert Throughput
(single node, 100M records)



Workload A Throughput
(Read Only)



Workload B Throughput
(Scan, Count, Average, and Sum)



GridDB導入事例

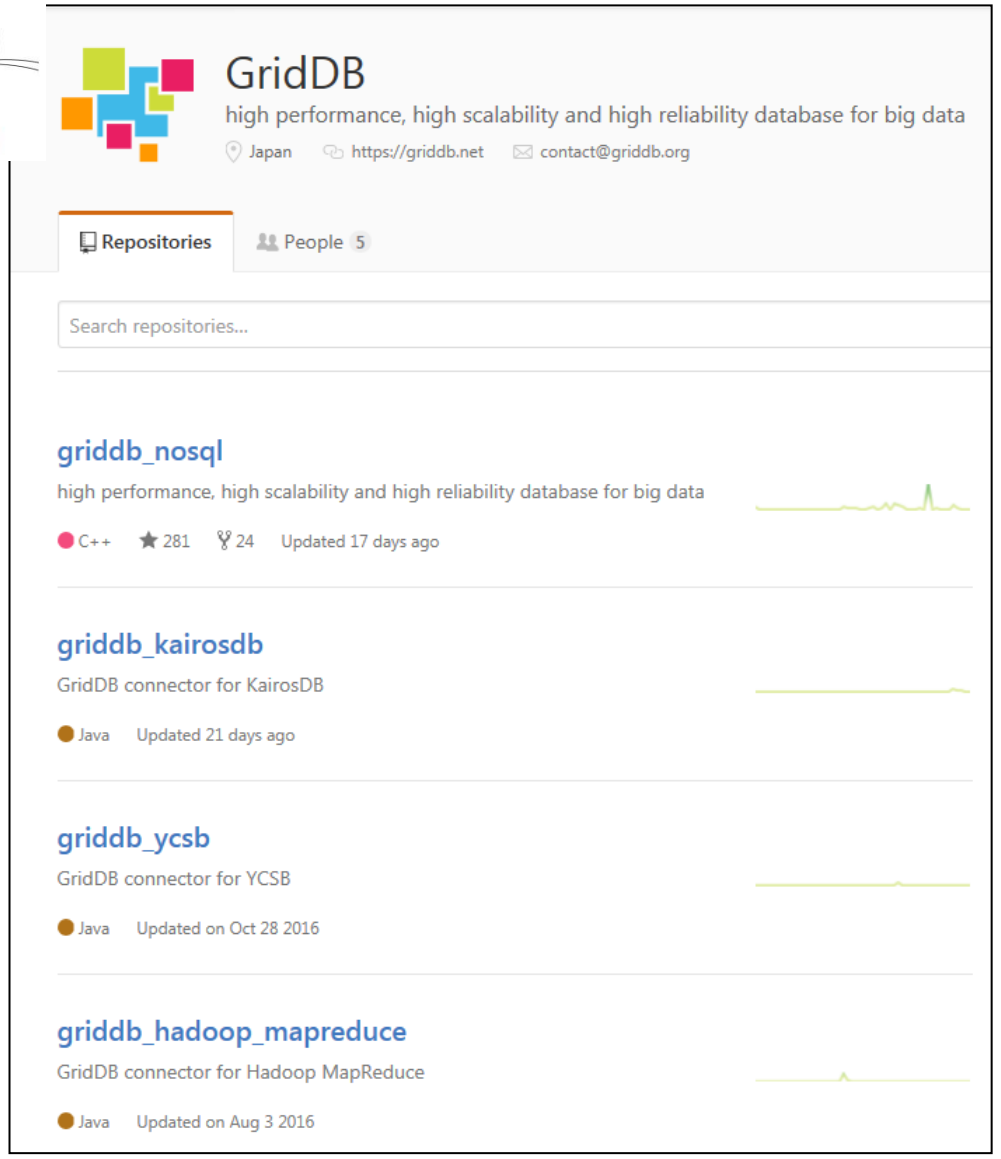
- **フランス リヨン 太陽光発電 監視・診断システム**
  - 発電量の遠隔監視、発電パネルの性能劣化を診断
- **クラウドBEMS**
  - ビルに設置された各種メータの情報の収集、蓄積、分析
- **石巻スマートコミュニティ プロジェクト**
  - 地域全体のエネルギーのメータ情報の収集、蓄積、分析
- **電力会社 低圧託送業務システム**
  - スマートメータから収集される電力使用量を集計し、需要量と発電量のバランスを調整
- **神戸製鋼所 産業用コンプレッサ稼働監視システム**
  - グローバルに販売した産業用コンプレッサをクラウドを利用して稼働監視
- **DENSO International Americaの次世代の車両管理システム**

https://griddb.net/ja/blog/griddb-automotive/

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

OSSサイト

- **GitHub上にNoSQL機能をソース公開 (2016/2/25)**
  - https://github.com/griddb/griddb_nosql/
- **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
- **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**

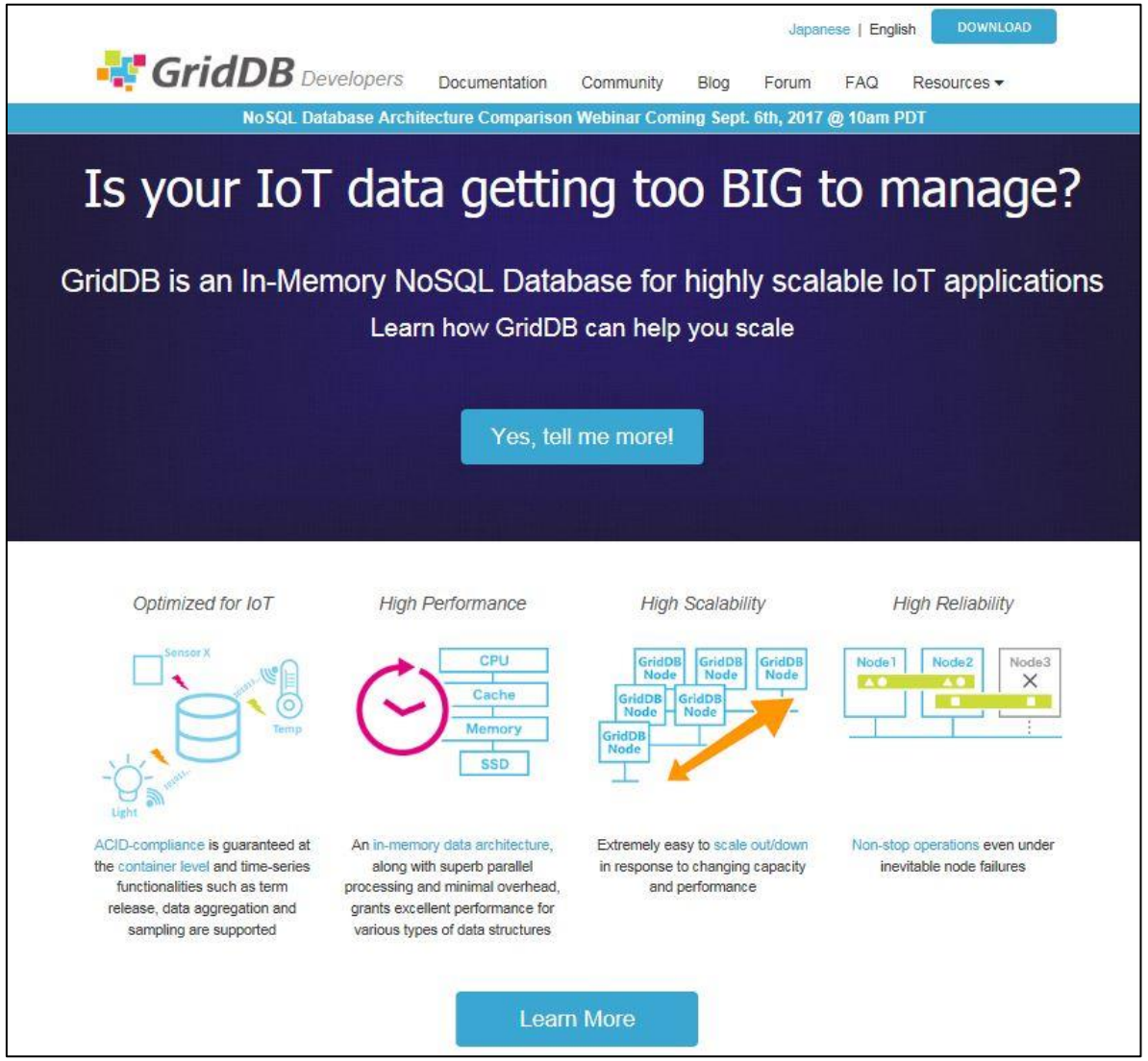


デベロッパーズサイト

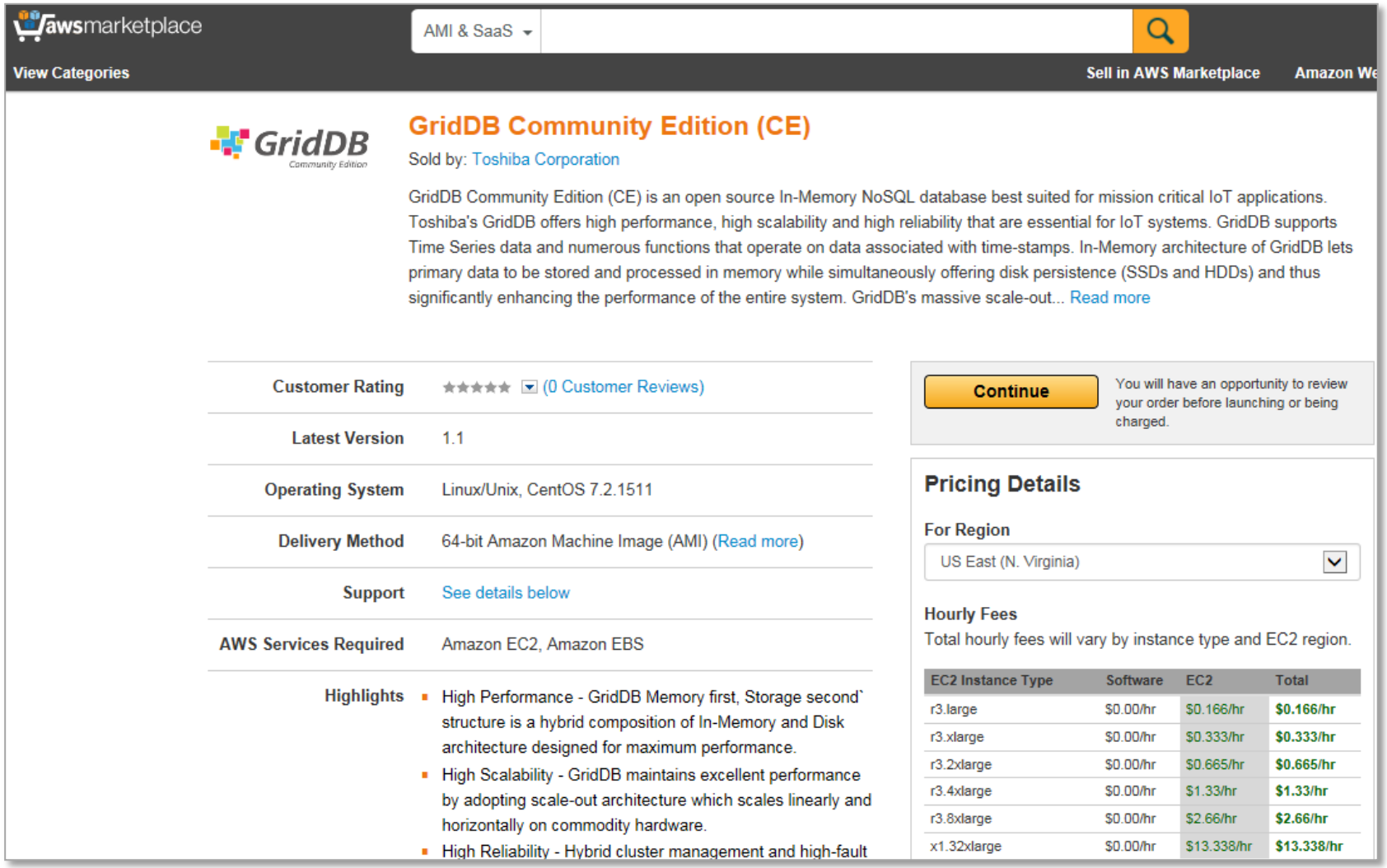
- **アプリケーション開発者向けのサイト**

  https://griddb.net/
- **コミュニケーションの場(フォーラム)を提供**
- **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



AWS Marketplaceで、すぐにGridDBを使用可能



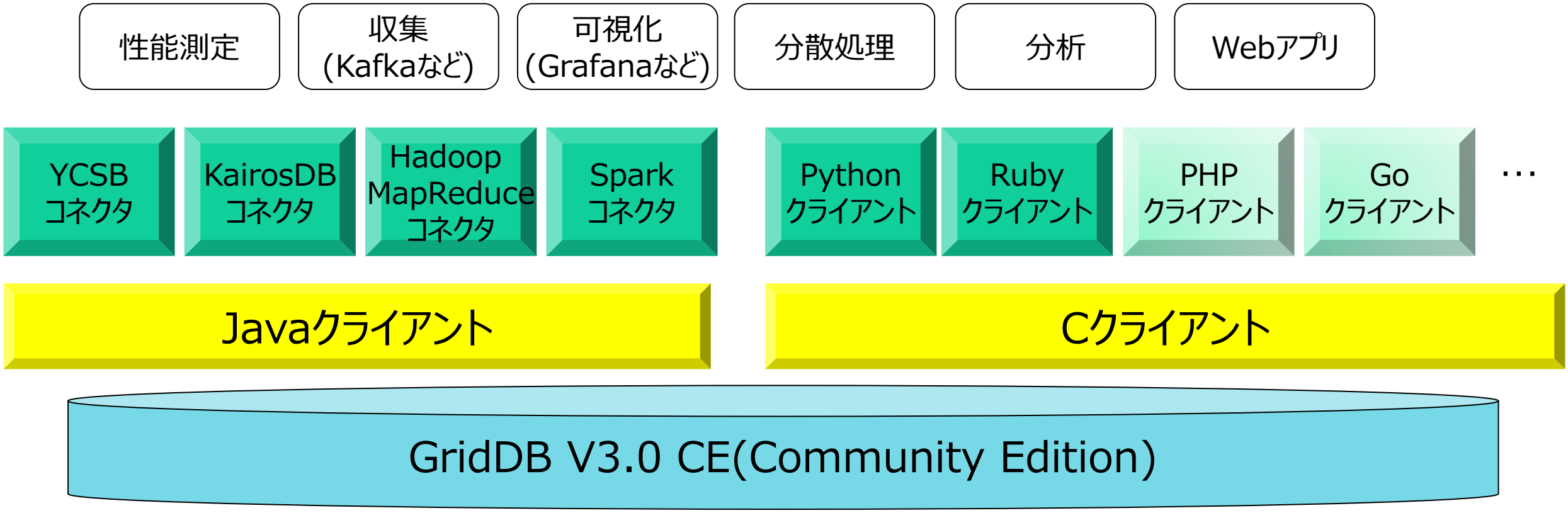
https://aws.amazon.com/marketplace/pp/B01N5ASG2S

Marketplace： パブリックIaaSの上で、各社のソフトウェアが時間単位で使えるようになっている

Pythonクライアント

GridDBのコネクタ、クライアント群

主なOSSとのコネクタや様々な開発言語のクライアントも公開



人工知能(AI)ソフトとの連携

DeepLearning/機械学習/ニューラルネット処理にPythonが主に使われている

DeepLerningフレームワーク
(Chainer, TensorFlow, Caffeなど)
・操作言語はPython
・分散処理用にSparkと連携

分析・AI向けライブラリ
・Python用のライブラリが豊富にある。
NumPy：N次元配列の数値演算ライブラリ
SciPy：数値演算アルゴリズム群
MatPlotLib：グラフ描画ライブラリ
Pandas：データ解析の支援ライブラリ
SciKit-learn：機械学習ライブラリ
NLTK：自然言語処理ライブラリ
など

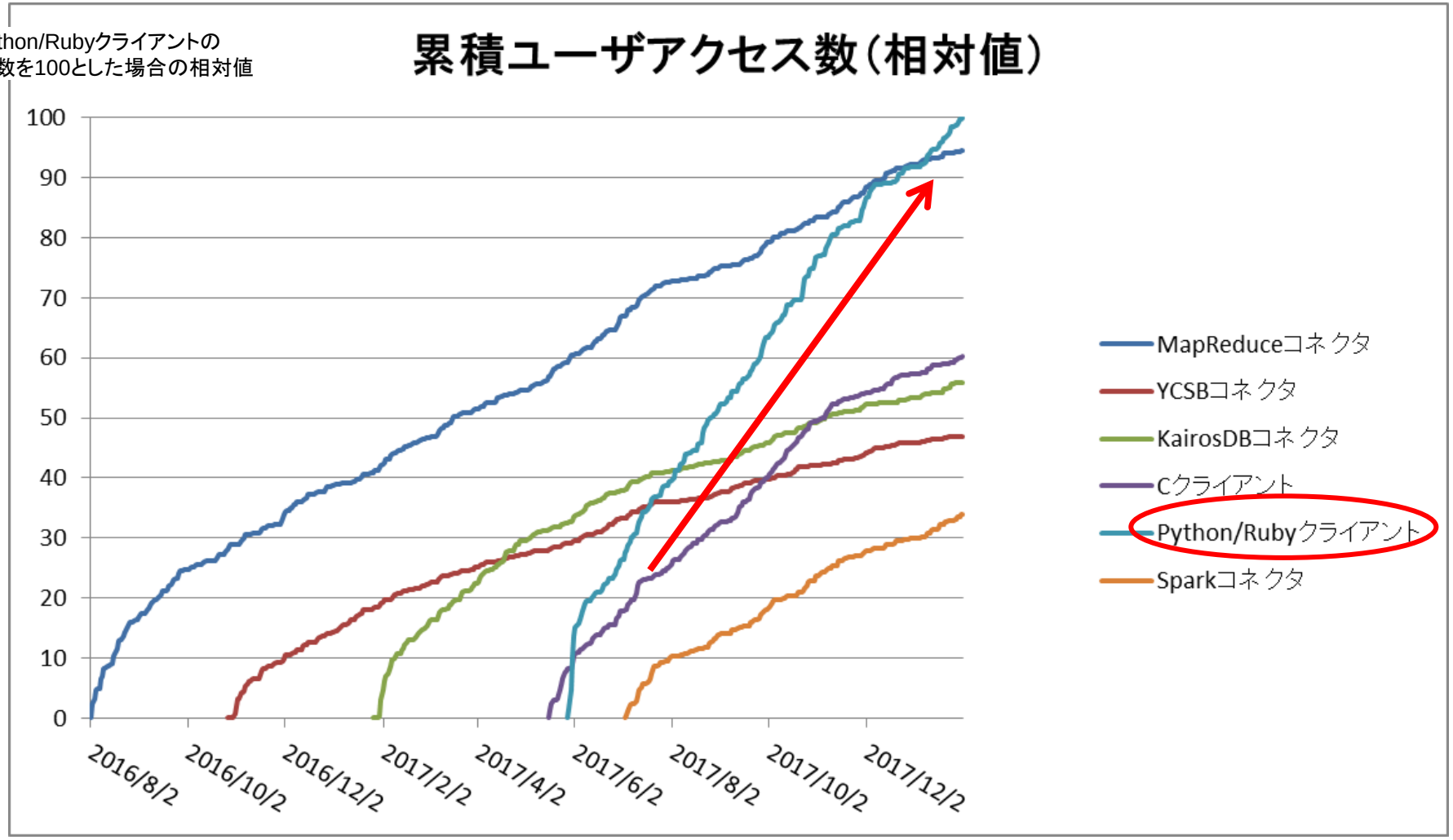
Spark

Sparkコネクタ

Pythonクライアント

GridDB V3.0 CE(Community Edition)

2018/1/31時点のPython/Rubyクライアントの
累積ユーザアクセス数を100とした場合の相対値



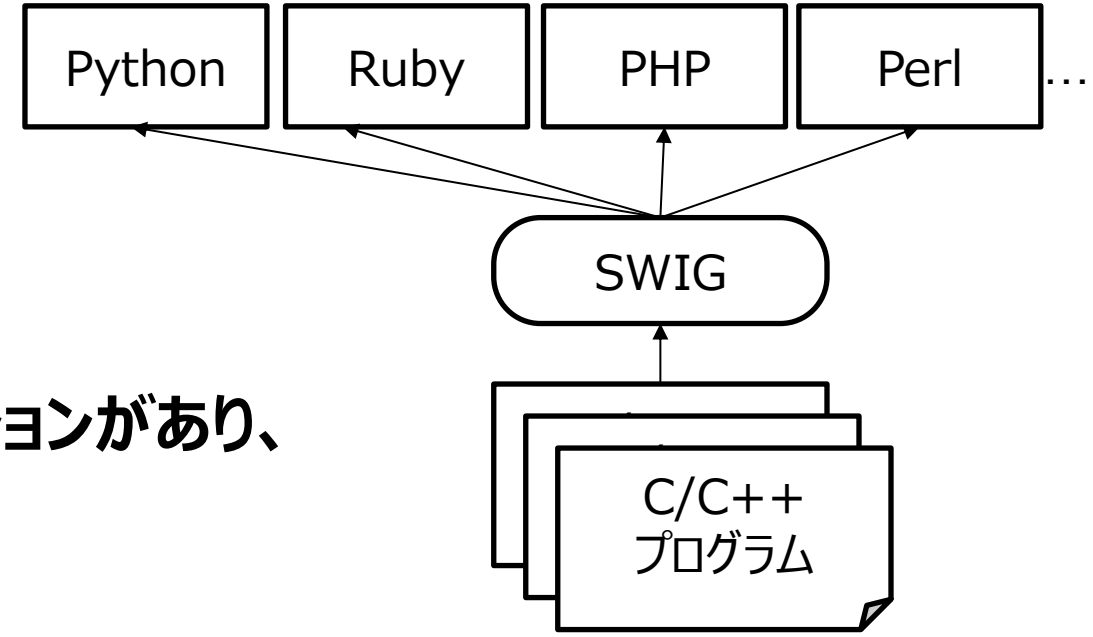
GridDBのクライアント開発

- **様々なプログラム言語に対応するためにCクライアントとSWIGを活用する。**
- **従来：性能第一**
  - Cクライアントと１：１対応のインタフェース

SWIG (Simplified Wrapper and Interface Generator)

- **C/C++ で書かれたプログラムやライブラリを、Pythonなど他のプログラミング言語に接続するためのオープンソースのツールである。**
  - サイト　http://www.swig.org/
  - ソース　https://github.com/swig/swig

  ※JavaScriptのテンプレートエンジンとは別物です。
- **1995年からDave Beazleyが開発。実装言語はC/C++。**
- **現在、Python, Ruby, PHP, Perlなど 20言語以上をサポート。最新版は3.0.12**
- **Subversionのpythonバインディングなどに利用されている。**
- **Pythonのsetup.pyに--swig-optsのオプションがあり、標準対応している。**



SWIG リリース状況

- …
- 2010/4 Software Freedom Conservancy(SFC) のメンバ･プロジェクトになる
- 2010/6 v2.0.0リリース
- 2010/10(v2.0.1) Goサポート
- 2014/3 V3.0.0リリース
- 2014/5(v3.0.1) Javascript(JavascriptCore, v8, node.js)サポート
- 2015/2(v3.0.5) Scilabサポート
- 2015/12(v3.0.8) std::array for Python
- 2016/5(v3.0.9) Python‘s implicit namespace packagesサポート
- 2016/12(v3.0.11) PHP7サポート
- 2017/1(v3.0.12)
- 現在、v4開発中

※http://www.swig.org/news.php

SWIG サンプル

```
/* File: example.h */
extern int fact(int n);
```

```
/* File : example.c */
int fact(int n) {
  if (n <= 1) return 1;
  else return n*fact(n-1);
}
```

Cファイル

```
/* File: example.i */
%module example %{   モジュール名
#inlude “example.h”  ヘッダ
%}

extern int fact(int n);   C宣言
```

インタフェースファイル

```
% swig -python example.i
    exsample.pyファイルとexample_wrap.cファイルが生成される
% gcc -c example.c example_wrap.c -I/usr/local/include/python2.1
% ld -shared example.o example_wrap.o -o _example.so
    _example.soファイルが生成される

% python
>>> import example
>>> example.fact(5) 120
```

※http://www.swig.org/tutorial.html

GridDBのクライアント開発

- **様々なプログラム言語に対応するためにCクライアントとSWIGを活用する。**
- **従来：性能第一**
  - Cクライアントと１：１対応のインタフェース
- **今回：ユーザビリティの向上**

  https://github.com/griddb/python_client
  - データ型を意識しないインタフェース
  - Pandasライブラリとの連携強化
  - その他
    - 日付、エラー処理の扱い

（A）データ型を意識しないインタフェース

- **リスト形式のロウデータによる操作**

  <従来>

  – フィールド単位で値を設定・取得するメソッドをデータ型別に使う必要があった
    - 例：Row.set_field_by_long(columnNo, longVal)

  <今回>

  – カラム順に並べたリスト形式のロウデータを用いる
    - 例：Container.put([1, “value1”, False])

- **暗黙的なデータ型変換**
  – Float型(Python)⇒TIMESTAMP型(GridDB) の例：Container.put([1421729699.000000, val])
  – String型(Python)⇒TIMESTAMP型(GridDB) の例：Container.put([“2013-05-31T20:33:20.000Z”, val])

		example	Converted data-type									
			BOOL	BYTE	SHORT	INTEGER	LONG	FLOAT	DOUBLE	STRING	TIMESTAMP	BLOB
Input data-type	int	3	✓	✓	✓	✓	✓	✓	✓			
	string	″yamada″								✓	✓	✓
	float	0.1						✓	✓		✓	
	boolean	True	✓									
	datetime	datetime.utcnow()									✓	
	bytearray	bytearray([65, 66, 67])										✓

（B）Pandasライブラリとの連携強化

- **Pandas DataFrameからGridDBへの登録**
  - 予めDataFrameに合致するスキーマのコンテナを作成しておく
  - DataFrameからリスト形式のデータを取得して、GridDBの登録メソッドを呼ぶ

```
df = …
Container.multi_put(df.values.tolist())
```

- **GridDB検索結果からPandas DataFrameの作成**
  - GridDB検索結果のRowSetからlist()関数を使ってDataFrameを作成する

```
rs = …
df = pd.DataFrame(list(rs), colums=rs.get_column_names())
…
```

Pythonクライアントのサンプルプログラム

【従来】

```
import griddb_python_client as gd
factory = gd.StoreFactory.get_default()

# Storeオブジェクト取得
store = factory.get_store({"notificationAddress": “239.0.0.1”,
  "notificationPort": “31999”, "clusterName": “myCluster”,
  "user": “admin”, "password": “admin”})

# コンテナ生成
col = store.put_container("col01",
  [("name", gd.GS_TYPE_STRING), ("status", gd.GS_TYPE_BOOL), ("count",
gdGS_TYPE_LONG)],
  gd.GS_CONTAINER_COLLECTION)


# 登録
row = col.create_row()
row.set_field_by_string(0, "name01")
row.set_field_by_bool(1, False)
row.set_field_by_long(2, 1)
# 検索
query=col.query("select * where name = 'name02'")
row2 = col.create_row()
rs = query.fetch(False)
while rs.has_next():
  rs.get_next(row2)
  name = row2.get_field_as_string(0)
  status = row2.get_field_as_bool(1)
  count = row2.get_field_as_long(2)
```

【今回】

```
import griddb_python as gd #モジュール名を変更
factory = gd.StoreFactory.get_instance()

# Storeオブジェクト取得
store = factory.get_store(host=“2319.0.0.1”,
  port=“31999”,  cluster_name=“myCluster”,
  username=“admin”, password=“admin”) #キーワード付きでプロパティを与える

# コンテナ生成
conInfo = gd.ContainerInfo("col01",
  [["name", gd.TYPE_STRING], ["status", gd.TYPE_BOOL], ["count",
gd.TYPE_LONG]],
  row_key=True)
col = store.put_container(conInfo)

# 登録
col.put_row([“name01”, False, 1])



# 検索
query=col.query("select * where name = 'name01'")
rs = query.fetch()
while rs.has_next():
    row = rs.next()
    # [“name01”, False, 1]
```

Pythonクライアントの展開状況（1/2）

- **PyPI (the Python Pakcage Index)でのPythonクライアントのパッケージ配布**

  ※PyPI：Python言語に関連するソフト群が登録されているサイト https://pypi.python.org/

  – pipコマンドでPythonクライアントを簡単にインストール可能

    % pip install griddb_python_client

- **OpenStack関係者によるMonasca用GridDBドライバ開発**

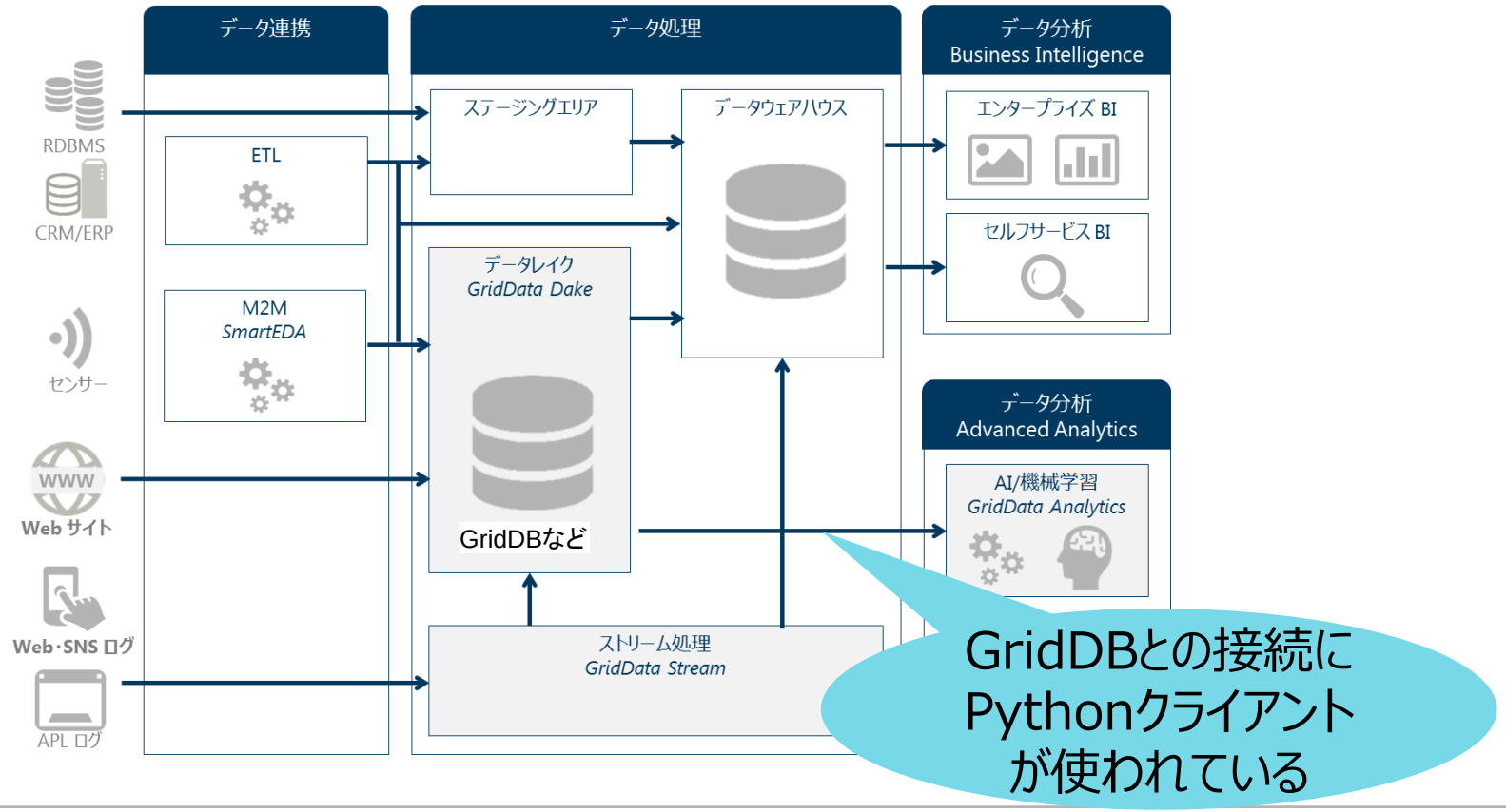
  – https://review.openstack.org/#/q/project:openstack/monasca-persister

  ※Monasca：OpenStackの監視基盤ソフト

  – Monascaのバックエンドとして、GridDBを使うためのドライバ開発にPythonクライアントが活用されている

Pythonクライアントの展開状況（2/2）

- **ビッグデータ(AI/機械学習)プラットフォーム *GridData Platform***
  - データ分析基盤、データレイク基盤、ストリーム処理基盤の３つから構成される製品
  - データ分析基盤の主な特徴：AI、機械学習を活用したデータ分析を行うためのスタンダードな分析基盤製品
    - ✓ ノートブック機能
    - ✓ 豊富な分析アルゴリズム
    - ✓ ディープラーニング
    - ✓ 分析の知見を共有するギャラリー
    - ✓ ビッグデータ対応
    - ✓ 外部DBとの接続：GridDBなど



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。**
- **OSSサイト、デベロッパーズサイト、AWS Marketplace上のサービス、などを公開・提供しています。**
- **ビッグデータの分析や機械学習の実現によく使われるPython言語用のクライアントが使いやすくなりました。**

オープンソースのGridDBを是非とも使ってみてください。

GridDBに関する情報

- **GridDB お問い合わせ**
  - デベロッパーズサイトのフォーラム、OSSサイトのGitHubのIssue、もしくはcontact@griddb.orgをご利用ください
- **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/
- **AWS Marketplace: GridDB Community Edition (CE)**
  - https://aws.amazon.com/marketplace/pp/B01N5ASG2S
- **Twitter: GridDB Community**
  - http://twitter.com/GridDBCommunity/
- **Facebook: GridDB Community**
  - http://fb.me/griddbcommunity/

GridDB™
Highly Scalable Database for IoT

- **OSSを利用したビッグデータ分析環境** GridData Analytics Cloud
  - https://www.griddata-analytics.net/

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：

- **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**
- **CAP 定理と GridDB**
- **GridDB Azureクラスタの構築**
- **GridDB's C/Python/Ruby APIsを使ってみよう**
- **YCSB向けGridDBコネクタを使ってみよう**
- **Apache SparkのためのGridDBコネクタ**

など

TOSHIBA

Leading Innovation