

TOSHIBA

Leading Innovation >>>

オープンソースのIoT向けスケールアウト型 データベース



GridDB™

Highly Scalable Database for IoT

～性能ベンチマーク結果と
OSSを利用したビッグデータ分析環境～

株式会社 東芝
野々村 克彦

目次

1. GridDBの概要

- 技術的特長
- YCSBによる性能測定結果
- 導入事例
- 公開サイト

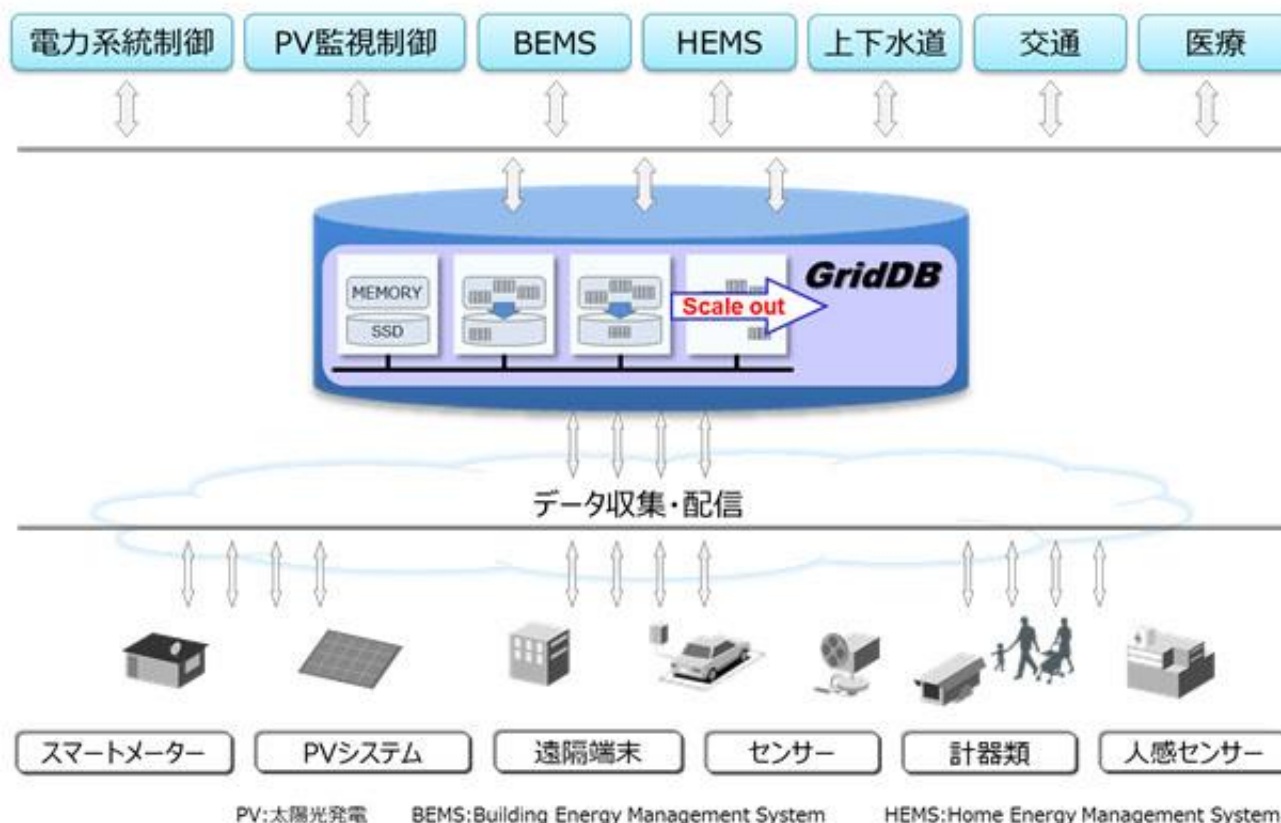
2. KairosDBコネクタ

3. OSSを利用したビッグデータ分析環境 *GridData Analytics Cloud*

4. まとめ

GridDB

- 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（2010年～）、製品化（2013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一貫性を保証

高性能

High Performance

- 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ー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

High Scalability

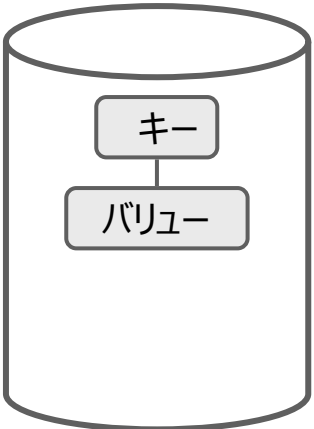
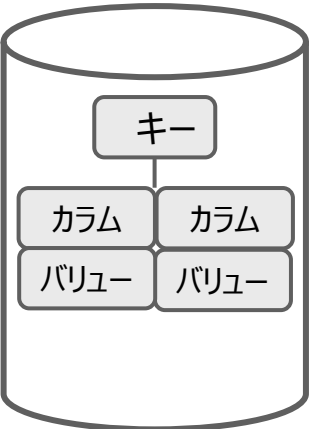
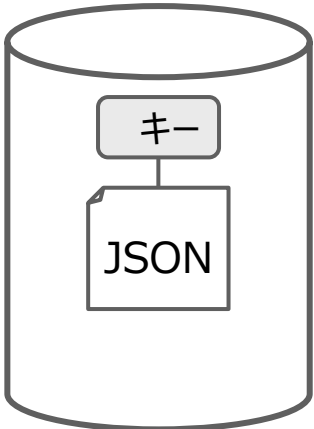
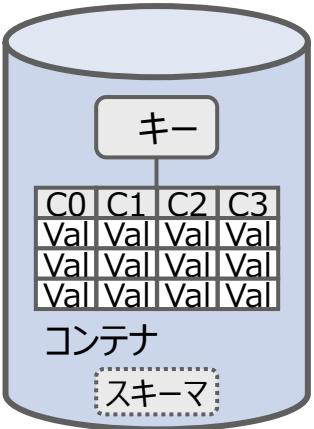
- データの少ない初期は少ないサーバで初期投資を抑え、データが増えるにしたがってサーバを増やし性能・容量を高めるスケールアウト型アーキテクチャ
- コンテナによりサーバ間通信を少なくし、高いスケーラビリティを実現

高い信頼性と 可用性

High Availability

- 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

データモデルの比較

	キーバリュー型	列指向型	ドキュメント型	キーコンテナ型
データモデル				
NoSQLの例	Riak	Cassandra	MongoDB	GridDB

• コンテナの種類

- コレクションコンテナ：レコード管理用
- 時系列コンテナ：時刻で並べられたレコード集合。時系列データ管理用
 - 期限解放機能、サンプリング機能など

IoT指向のデータモデル

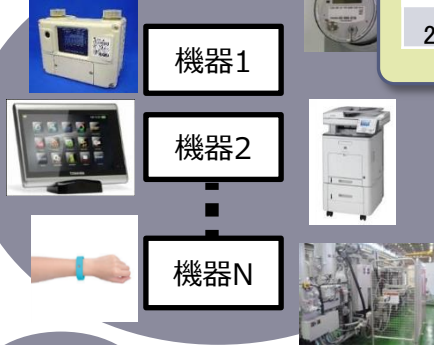
キーコンテナ型のデータモデル

- キーバリューをグループ化するコンテナ（テーブル）
- コンテナのスキーマ定義が可能。カラムにインデックスを設定可能
SQLライクなクエリ(TQL)が利用可能
- レコード単位でトランザクション操作（コンテナ単位でACID保証）

※ACID : Atomicity、Consistency、Isolation、Durability

IoTデータ

機器センサー



機器1のレコード

日時	センサA	センサB
2015/01/01 0:00	7.788683	0.648364

キー

機器 1

対象毎にIoTデータを格納

データ格納

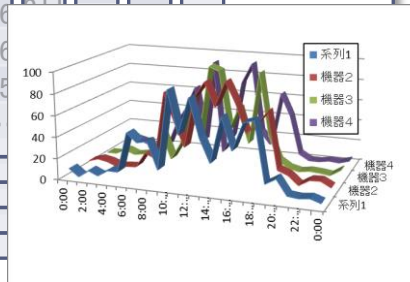
日時	センサA	センサB
2015/01/01 0:00	7.788683	0.648364
2015/01/01 1:00	0.68874	0.353611
2015/01/01 2:00	7.677135	5.881216
2015/01/01 コンテナ 3:1816	2.511166	
2015/01/01 4:00	9.739242	0.655805
...

テーブル表現で管理

株価

ログ

履歴



単純なキーバリュー型とは異なり、使い慣れたRDBに近いモデリングが可能

Cassandraとの性能比較（YCSB）

- **YCSB(Yahoo! Cloud Serving Benchmark)**

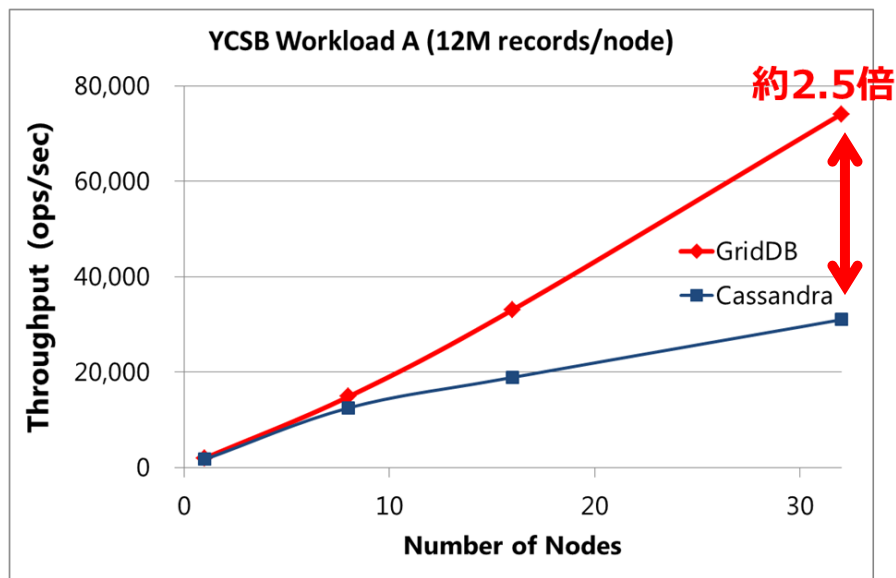
<https://github.com/brianfrankcooper/YCSB/wiki>

- NoSQLの代表的なベンチマーク。但し、必ずしもIoT向けではない
- Load/Runの2フェーズ、Runは6種のworkloadから成る

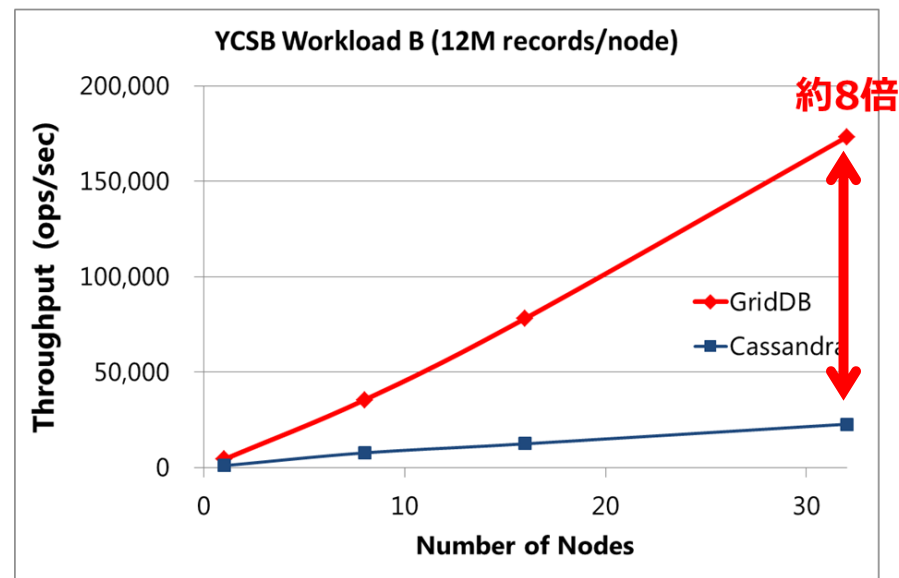
work load	type	insert	read	update	scan
A	Update heavy		50%	50%	
B	Read mostly		95%	5%	
C	Read only		100%		
D	Read latest	5%	95%		
E	Short ranges	5%			95%
F	Read-modify-write		50%	50% ※read-modify	

Cassandraとの性能比較 (YCSB)

- 高速性を売りにするCassandraと比較しても、GridDBの方が圧倒的に高速



Read 50% + Write 50%

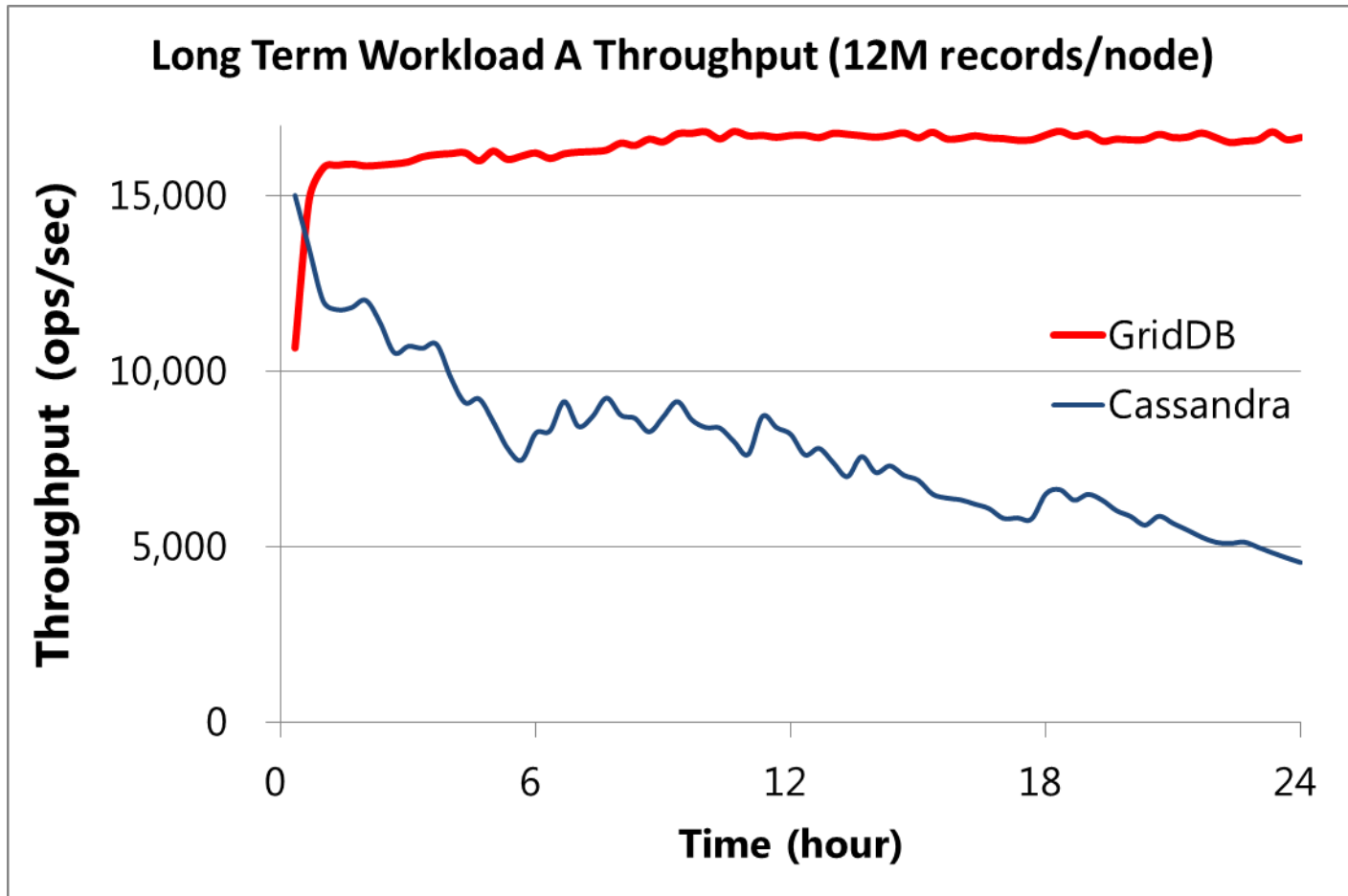


Read 95% + Write 5%

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。ベンチマークの詳細をホワイトペーパーとして配布中！
https://www.griddb.net/en/docs/Fixstars_NoSQL_Benchmarks.pdf

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

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

- **サーバとJavaドライバ、各種コネクタを公開中**

- Hadoop MapReduceコネクタ：並列分散処理
- YCSBコネクタ：性能測定
- **KairosDBコネクタ (2017/1/31公開)：**
メトリック、タグを使った時系列DB (後程説明)

The screenshot shows the GitHub profile for GridDB, which includes the text "high performance, high scalability and high reliability database" and "Japan". Below the profile, there is a "Repositories" section with a search bar and a list of repositories. The first repository listed is "griddb_nosql", described as "high performance, high scalability and high reliability database", with 281 stars and 24 forks, updated 17 days ago. Other repositories listed include "griddb_kairosdb" (GridDB connector for KairosDB, updated 21 days ago), "griddb_ycsb" (GridDB connector for YCSB, updated on Oct 28 2016), and "griddb_hadoop_mapreduce" (GridDB connector for Hadoop MapReduce, updated on Aug 3 2016).

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

- ホワイトペーパー
 - マニュアル
 - サンプルコード
- など

Japanese | English

GridDB Developers ドキュメント ダウンロード コミュニティ ブログ FAQ **フォーラム**

IoT指向モデル 高い性能 高い拡張性 高い信頼性と可用性

GridDB Is Now Available On AWS!

クイックスタート ダウンロード

GridDBとは？

IoTとビッグデータに最適な高い拡張性をもつ
NoSQL型データベースです

AWS Marketplaceで、すぐにGridDBを使用可能

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

GridDB Community Edition (CE)
Sold by: Toshiba Corporation

GridDB Community Edition (CE) is an open source In-Memory NoSQL database best suited for mission critical IoT applications. Toshiba's GridDB offers high performance, high scalability and high reliability that are essential for IoT systems. GridDB supports Time Series data and numerous functions that operate on data associated with time-stamps. In-Memory architecture of GridDB lets primary data to be stored and processed in memory while simultaneously offering disk persistence (SSDs and HDDs) and thus significantly enhancing the performance of the entire system. GridDB's massive scale-out... [Read more](#)

Customer Rating	★★★★☆ (0 Customer Reviews)
Latest Version	1.1
Operating System	Linux/Unix, CentOS 7.2.1511
Delivery Method	64-bit Amazon Machine Image (AMI) (Read more)
Support	See details below
AWS Services Required	Amazon EC2, Amazon EBS

Highlights

- High Performance - GridDB Memory first, Storage second structure is a hybrid composition of In-Memory and Disk architecture designed for maximum performance.
- High Scalability - GridDB maintains excellent performance by adopting scale-out architecture which scales linearly and horizontally on commodity hardware.
- High Reliability - Hybrid cluster management and high-fault tolerant system of GridDB is exceptional for mission-critical applications.

Pricing Information

Use the Region dropdown selector to see software and infrastructure pricing information for the chosen AWS region.

For Region:

Pricing Details

Software pricing is based on your chosen options, such as subscription terms and AWS region. Infrastructure prices are estimates only. Final prices will be calculated according to actual usage and reflected on your monthly report.

1 Software Pricing

The data below shows pricing per instance for services hosted in US East (N. Virginia).

GridDB Community Edition (CE) - Hourly

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

KairosDBコネクタ

KairosDB

- <https://github.com/kairosdb/kairosdb>
 - **代表的な時系列DB**
 - 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
 - **メトリック、タグを用いた操作**
 - **連携性が高い**
 - Telnet/REST API、Java Client
 - collectd、Grafana、Kafka
 - **バックエンドが選択可能**
 - H2, Cassandra, HBase
- ⇒今回、KairosDBコネクタでGridDBを選択可能にした
https://github.com/griddb/griddb_kairosdb

データモデル

- メトリックとタグ集合で構成される

Tags : (タグ名,タグ値)の集合

	tag1	tag2	...	tagM
metric1				
metric2				
...				
metricN				

(例)

(siteNo,hostNo)= (1,1),..., (1,30),
(2,1),..., (2,20),
...
(2000,1),..., (2000,50)

メトリック

	siteNo	hostNo	
cpu			
memory			
diskio			
network			

Cassandraでのデータの持ち方

Data schema

ColumnFamilyName	RowKey	ColumnType	ValueType
"data_points"	metricName+ rowStartTime+ valueType+tags	double	Byte[]

Data image

	Columns				
	上限3週間				
RowKey	0	1	2	3	...
"cpu-1421729699000-double-siteNo:1-hostNo:2"	0.00	0.20	0.80	0.40	...

※UTC:"2013-05-31T20:33:20.000Z" ⇔ timestamp:1421729699000(ms)

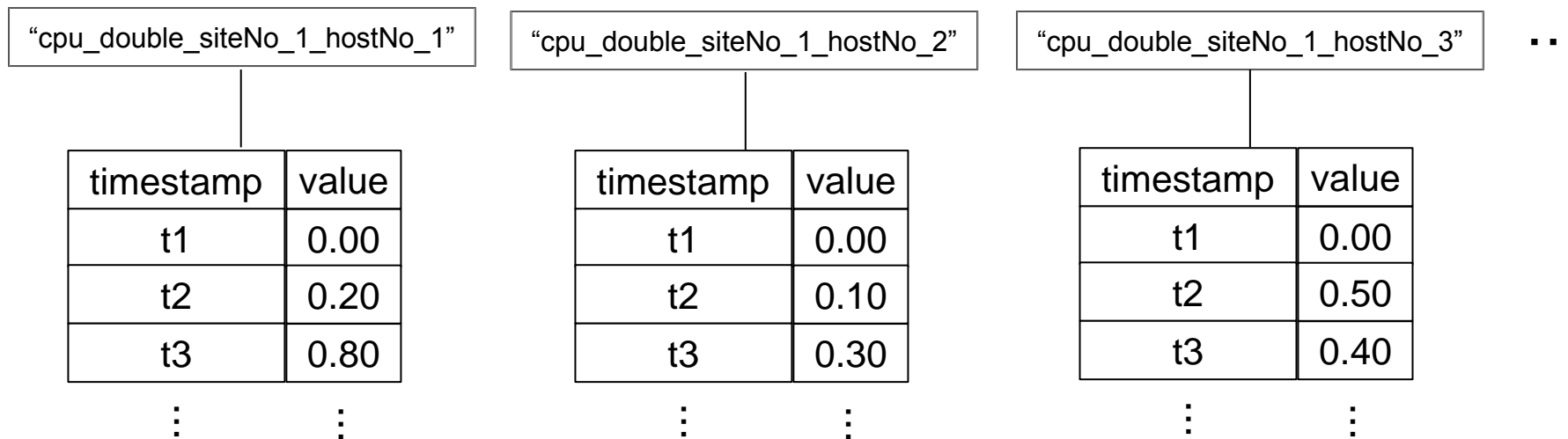
GridDBでのデータの持ち方

Data schema

DataType	ContainerKey (ContainerName)	RowKey	ValueType
"data_points"	metricName+ valueType+tags	timestamp	Byte[]

Data image

ContainerKey



ビルド・実行方法

• ダウンロード

```
$ curl -O -location  
https://github.com/kairosdb/kairosdb/archive/v1.1.1.tar.gz  
$ tar xfvz v1.1.1.tar.gz  
$ git clone https://github.com/griddb/griddb_kairosdb.git  
$ cp -r griddb_kairosdb/src/main/java/org/kairosdb/datastore/griddb  
kairosdb-1.1.1/src/main/java/org/kairosdb/datastore  
$ cd kairosdb-1.1.1
```

• 編集

- Ivy.xmlファイル
 - `<dependency org="org.apache.commons" name="commons-pool2" rev="2.0"/>`を追加
- src/main/resources/logback.xmlファイル
 - `<logger name="com.toshiba.mwcloud.gs.GridStoreLogger" level="INFO"/>`を追加

ビルド・実行方法(2)

● 編集(2)

- src/main/resources/kairosdb.propertiesファイル

```
#kairosdb.service.datastore=org.kairosdb.datastore.h2.H2Module  
kairosdb.service.datastore=org.kairosdb.datastore.griddb.GriddbModule
```

```
...
```

```
kairosdb.datastore.griddb.cluster_name=<GridDB cluster name>
```

```
kairosdb.datastore.griddb.user=<GridDB user name>
```

```
kairosdb.datastore.griddb.password=<GridDB password> マルチキャスト方式でサーバと接続する場合
```

```
kairosdb.datastore.griddb.notification_address=<GridDB notification address(default is 239.0.0.1)>
```

```
kairosdb.datastore.griddb.notification_port=<GridDB notification port(default is 31999)>
```

```
kairosdb.datastore.griddb.notification_member=
```

```
kairosdb.datastore.griddb.notification_provider_url=
```

```
kairosdb.datastore.griddb.consistency=
```

```
kairosdb.datastore.griddb.container_cache_size=
```

```
kairosdb.datastore.griddb.data_affinity_pattern=
```

```
kairosdb.datastore.griddb.failover_timeout=
```

```
kairosdb.datastore.griddb.transaction_timeout=
```

```
kairosdb.datastore.griddb.datapoint_ttl=0
```

```
kairosdb.datastore.griddb.max_data_cache_size=50000
```

```
kairosdb.datastore.griddb.max_write_buffer_size=500000
```

```
kairosdb.datastore.griddb.number_request_concurrency=100
```

```
kairosdb.datastore.griddb.write_delay=1000
```

```
kairosdb.datastore.griddb.write_buffer_datapoint=8
```

● 設定

- GridDB Javaドライバ gridstore.jarをlibフォルダに置く

ビルド・実行方法(3)

- **GridDBサーバのサービス開始**

※https://github.com/griddb/griddb_nosql/README_ja.md参照のこと

- **GridDB版KairosDBのビルド・起動**

```
$ export CLASSPATH=tools/tablesaw-1.2.2.jar
```

```
$ java make run
```

しばらく待つと「KairosDB service started」が表示される

```
20:38:17.744 [main] INFO [Main.java:306] - -----
```

```
20:38:17.745 [main] INFO [Main.java:307] -     KairosDB service started
```

```
20:38:17.745 [main] INFO [Main.java:308] - -----
```

• Telnet API

- put <metric name> <timestamp> <value> <tag> <tag>...
- 例 : echo "put cpu 1421720699000 0.8 siteNo=1 hostNo=1" |
nc -w 30 10.45.100.4 4242

※UTC:"2013-05-31T20:33:20.000Z" ⇔ timestamp:1421729699000(ms)

• REST API

- Method POST
- Request http://[host]:[port]/api/v1/datapoints
- Bodyの例 ※Telnet APIの例の100秒後のデータを登録

```
[{ "name": "cpu",  
  "timestamp": 1421720799000,  
  "type": "double", "value": 0.6,  
  "tags": {"siteNo": "1", "hostNo": "1"}  
}]
```

検索

- **REST API**

- Method POST
- Request `http://[host]:[port]/api/v1/datapoints/query`
- Bodyの例 ※指定時刻から5日間について10分毎のcpu平均を求める

```
{ "start_absolute": 1421720000000,  
  "end_relative": { "value": "5", "unit": "days" },  
  "metrics": [  
    { "tags": { "siteNo": ["1"], "hostNo": ["1", "2"] },  
      "name": "cpu",  
      "limit": 10000,  
      "aggregators": [  
        { "name": "avg",  
          "sampling": { "value": 10, "unit": "minutes" }  
        }  
      ]  
    }  
  ] }
```

補足) 実行例

//1件目登録(telnet)

```
$ echo "put cpu 1421720699000 0.8 siteNo=1 hostNo=1" | nc -w 30  
10.45.100.4 4242
```

//2件目登録(REST)

```
$ curl -v -H "Accept: application/json" -H "Content-type: application/json" -X  
POST -d '[{ "name": "cpu", "timestamp": 1421720799000, "type": "double",  
"value": 0.6, "tags":{"siteNo":"1", "hostNo":"1"}}]'  
http://10.45.100.4:8080/api/v1/datapoints
```

//検索(REST)

```
$ curl -v -H "Accept: application/json" -H "Content-type: application/json" -X  
POST -d '{ "start_absolute": 1421720000000, "end_relative": { "value": "5",  
"unit": "days" }, "metrics": [ { "tags": { "siteNo": ["1"], "hostNo": ["1", "2"] },  
"name": "cpu", "limit": 10000, "aggregators": [ { "name": "avg", "sampling": {  
"value": 10, "unit": "minutes" } } ] ] }'  
http://10.45.100.4:8080/api/v1/datapoints/query
```

...

```
{ "queries": [ { "sample_size": 2, "results": [ { "name": "cpu", "group_by": [ { "name": "t  
ype", "type": "number" } ], "tags": { "hostNo": ["1"], "siteNo": ["1"] }, "values": [[1421  
720699000, 0.7]] ] } ] } //検索結果
```

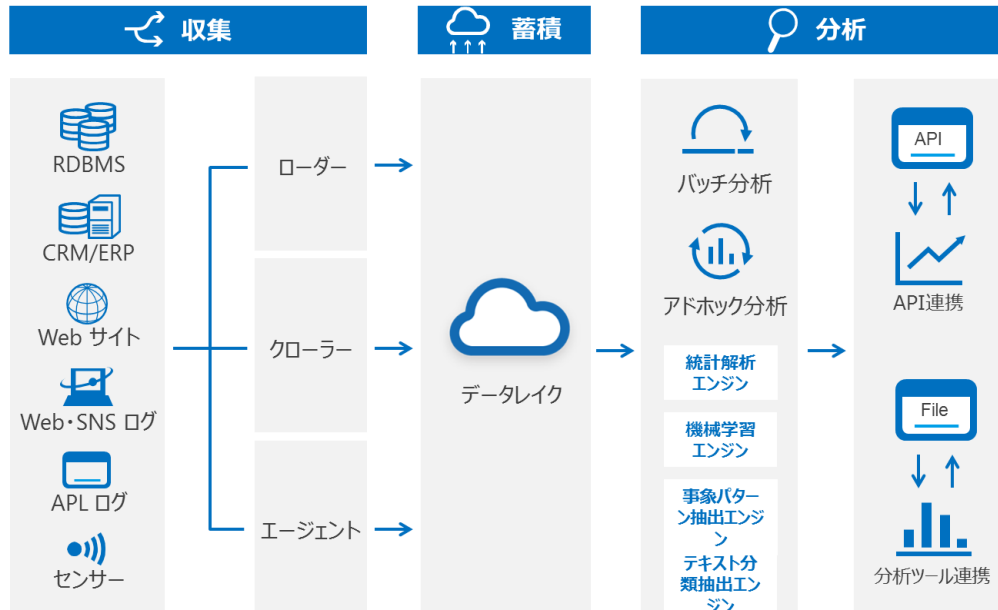
OSSを利用したビッグデータ分析環境

GridData *Analytics* Cloud

<https://www.griddata-analytics.net/>

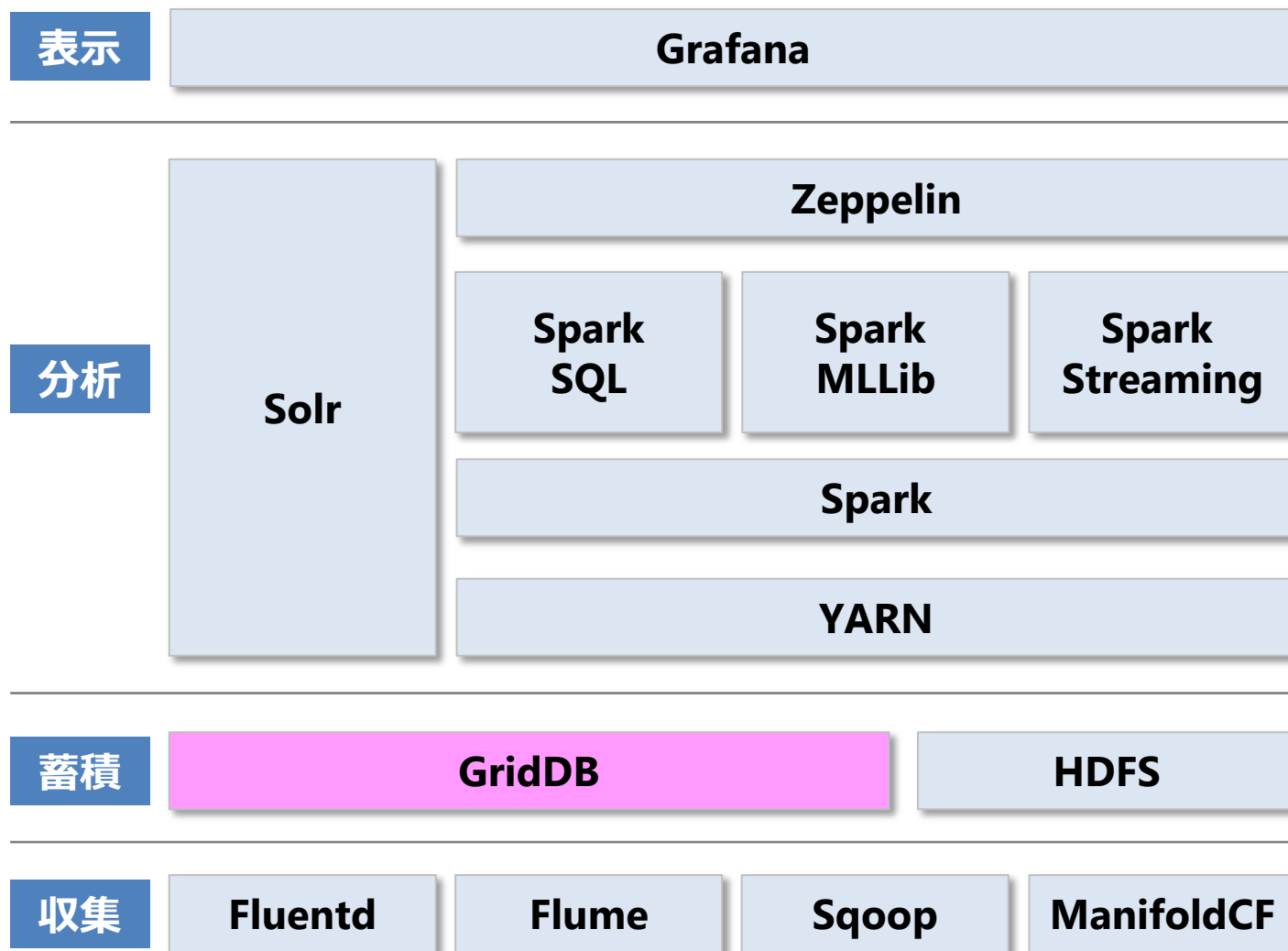
GridData Analytics Cloud

ビッグデータの収集・蓄積・分析（AI / 機械学習）をクラウドで簡単、かつ賢く実現



- クラウドだから初期コストを抑えて、気軽にスタート
- データコレクターを活用して効率よく収集
- 機械学習ライブラリを使用して賢く分析
- インタラクティブな環境で、分析の試行錯誤も簡単に

GridData Analytics Cloudを構成するOSS



注意：一部、予定を含む



デモ

機械学習による与信審査

①学習モデル生成 ②学習モデル評価 ③学習モデルによる予測

デモデータ

カリフォルニア大学アーバイン校が公開している
ポルトガルの定期預金のキャンペーンのデータ (約4万件)

Yの値が1 = 契約した、0 = 契約しなかった

1 - age (numeric)

2 - job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')

3 - marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'university.degree')

5 - default: has credit in default

6 - housing: has housing loan

7 - loan: has personal loan? (categorical: 'cellular', 'telephone')

8 - contact: contact communication channel

9 - month: last contact month

10 - day_of_week: last contact day of week

11 - duration: last contact duration in seconds

(e.g., if duration=0 then y='no', if y is obviously known. Thus, the intention is to have a realistic dataset)

12 - campaign: number of consecutive days of advertising

13 - pdays: number of days that client was not previously contacted

14 - previous: number of contacts performed before this campaign (0 means client was not previously contacted)

15 - poutcome: outcome of the previous campaign

16 - emp.var.rate: employment variation rate

17 - cons.price.idx: consumer price index

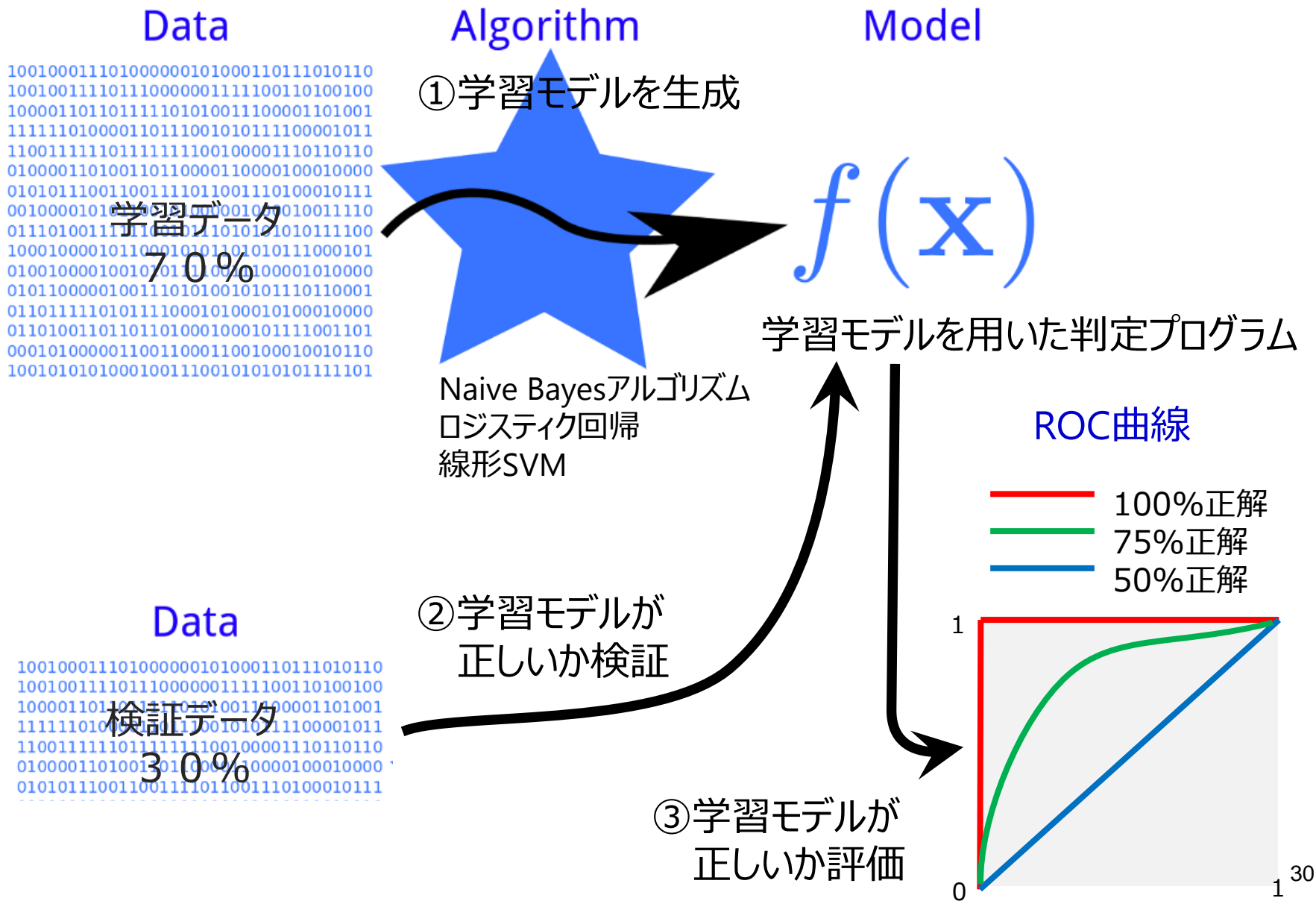
18 - cons.conf.idx: consumer confidence index

19 - euribor3m: euribor 3 month rate

20 - nr.employed: number of employed persons

	A	B	C	D	E	F	G	H	I	J	K	L	M
	age	job_str	marital_str	education_str	default_str	housing_str	loan_str	contact_str	month	day_of_week	duration	poutcome_str	y
1	44	blue-collar	married	basic.4y	unknown	yes	no	cellular	10	1	210	nonexistent	0
2	53	technician	married	unknown	no	no	no	cellular	4	3	138	nonexistent	0
3	28	management	single	university.degree	no	yes	no	cellular	1	1	339	success	1
4	39	services	married	high.school	no	no	no	cellular	5	3	185	nonexistent	0
5	55	retired	married	basic.4y	no	yes	no	cellular	10	3	137	success	1
6	30	management	divorced	basic.4y	no	yes	no	cellular	2	5	68	nonexistent	0
7	37	blue-collar	married	basic.4y	no	yes	no	cellular	3	1	204	nonexistent	0
8	39	blue-collar	divorced	basic.9y	no	yes	no	cellular	3	3	191	nonexistent	0
9	36	admin.	married	university.degree	no	no	no	cellular	1	2	174	success	1
10	27	blue-collar	single	basic.4y	no	yes	no	cellular	5	1	191	failure	0
11	34	housemaid	single	university.degree	no	no	no	telephone	3	3	62	nonexistent	0
12	41	management	married	university.degree	no	yes	no	cellular	10	1	789	nonexistent	0
13	55	management	married	university.degree	no	no	no	cellular	10	2	372	nonexistent	1
14	33	services	divorced	high.school	no	yes	no	cellular	3	5	75	nonexistent	0
15	26	admin.	married	high.school	no	no	yes	telephone	1	2	1021	nonexistent	0
16	52	services	married	high.school	unknown	yes	no	cellular	2	1	117	nonexistent	0
17	35	services	married	high.school	no	no	no	cellular	5	1	1034	nonexistent	1
18	27	admin.	single	university.degree	no	no	no	telephone	7	5	540	nonexistent	1
19	28	blue-collar	married	basic.9y	unknown	no	no	telephone	3	1	140	nonexistent	0
20	26	unemployed	single	basic.9y	no	yes	yes	cellular	2	2	104	nonexistent	0
21	41	unemployed	married	basic.9y	unknown	yes	no	telephone	5	3	246	failure	0
22	35	blue-collar	single	unknown	no	no	yes	telephone	1	3	1114	nonexistent	0
23	40	admin.	married	university.degree	unknown	yes	no	telephone	2	4	340	nonexistent	0
24	32	technician	single	professional.course	no	no	no	cellular	2	1	35	nonexistent	0
25	41	blue-collar	married	high.school	no	yes	yes	cellular	2	1	241	nonexistent	0
26	34	entrepreneur	single	university.degree	no	yes	no	cellular	3	5	168	nonexistent	0
27	49	technician	divorced	unknown	no	yes	yes	cellular	7	1	81	nonexistent	0

デモ（学習モデルの生成・評価）



デモ（学習モデルの生成・評価）

学習モデルを生成し、ROC曲線を出力

The screenshot displays the GridData Analytics Cloud interface. The main workspace, named "D10_NaiveBayes_roc", contains a Scala script for training and evaluating a Naive Bayes model. The script includes the following code:

```
row.getDouble(26),
row.getDouble(27)))

// データの 訓練データ (教師データ) (70%) と 評価データ (30%)への分割
val splits = data.randomSplit(Array(0.7, 0.3), seed = 11L)
val test = splits(1)

// 学習モデルのロード
val model = NaiveBayesModel.load(sc, "banking_model_bayes")

val predictionAndLabel = test.map(p => (model.predict(p.features), p.label))
val accuracy = 1.0 * predictionAndLabel.filter(x => x._1 == x._2).count() / test.count()

// 学習モデル評価指標の入手
val metrics = new BinaryClassificationMetrics(predictionAndLabel)
var roc = metrics.roc()
println("%table X\tY")
roc.collect.foreach(tuple => println(tuple._1+"\t"+tuple._2))
```

Below the script, a ROC curve plot is displayed. The plot shows the True Positive Rate (Y-axis) versus the False Positive Rate (X-axis). The curve starts at (0,0) and rises to approximately (0.15, 0.7) before leveling off towards (1,1). The plot is titled "Stacked Stream Expanded" and includes a legend with a blue dot labeled "Y".

On the right side of the interface, a "Table name list" is visible, showing a table named "education_event_data" with 0 rows. Below this, a table of data points is shown:

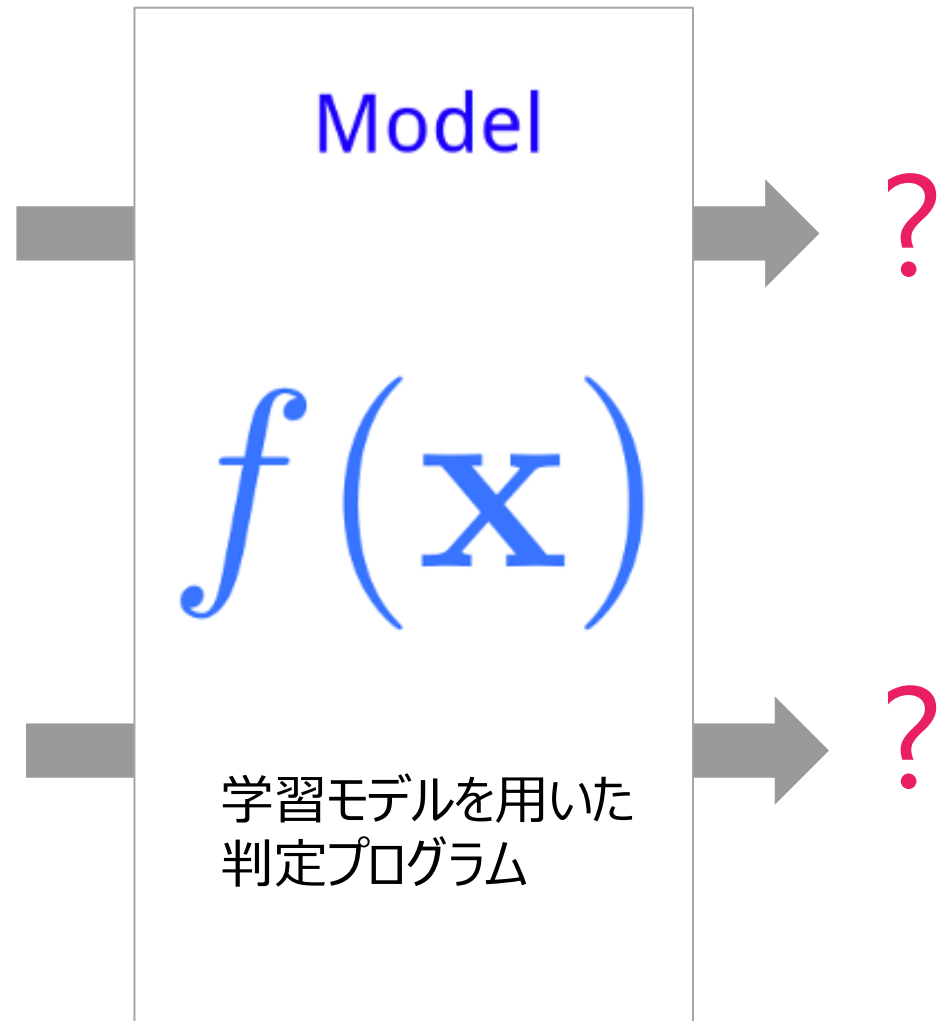
id	tstmp	TM	HU
1	2016-05-20T11:25:30.000Z	0to1636	
2	2016-05-20T11:26:15.000Z	0to1636	
3	2016-05-20T11:27:10.000Z	1636to1762	
1	2016-05-20T11:25:30.000Z		over
3	2016-05-20T11:27:10.000Z		over
1	2016-05-20T11:37:25.000Z		
3	2016-05-20T11:39:35.000Z		
1	2016-05-20T11:45:00.000Z		
2	2016-05-20T11:46:00.000Z		
3	2016-05-20T11:47:10.000Z		

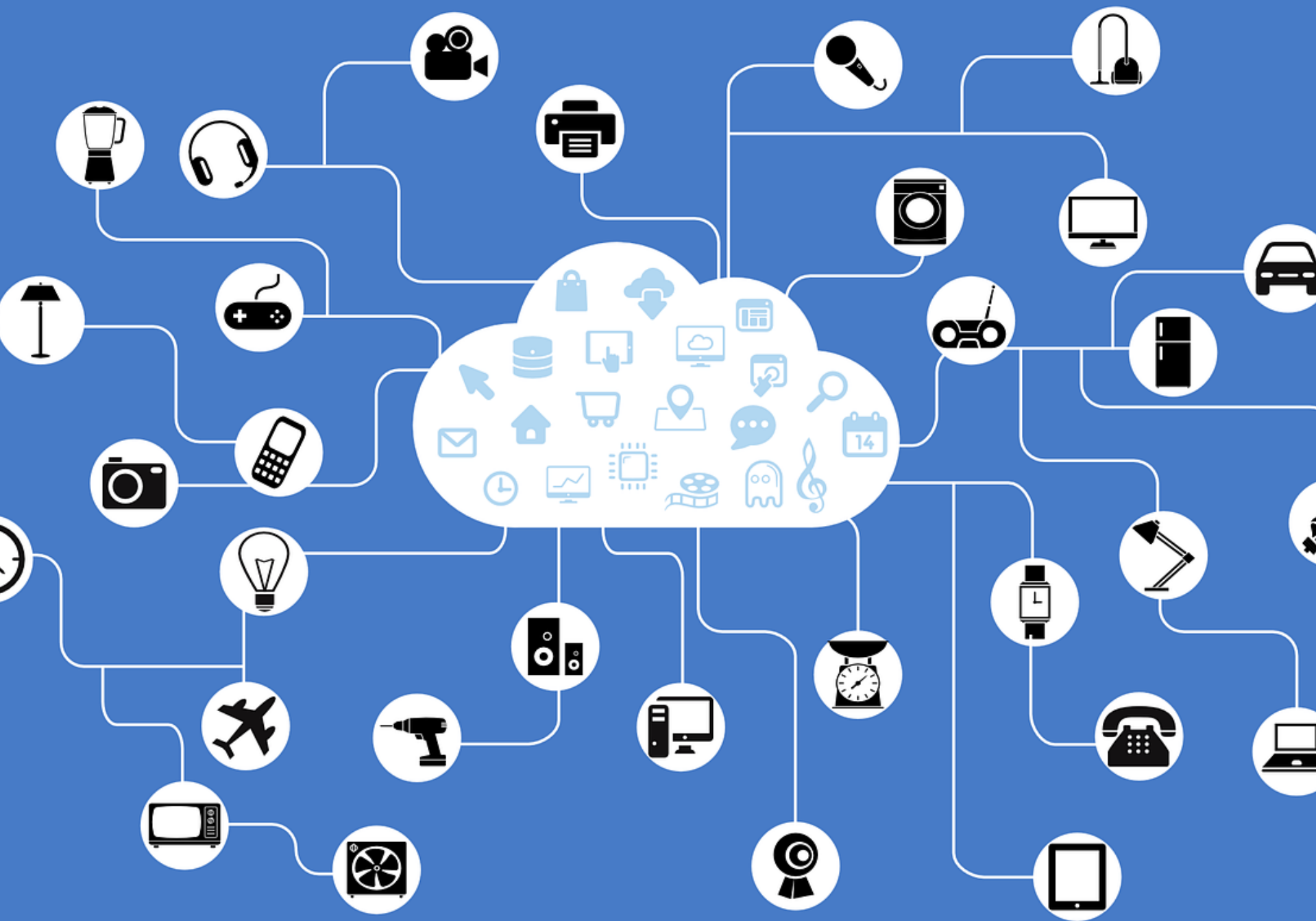
デモ（学習モデルによる予測）

未判定のデータを使い、学習モデルを用いた判定プログラムによる予測を試みる

name	value
age	39
job	blue-collar
material	divorced
education	basic.9y
default	no
housing	yes
loan	no
y	0

name	value
age	36
job	admin
material	married
education	university.degree
default	no
housing	no
loan	no
y	1





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™

Highly Scalable Database for IoT

■ GridDB OSSサイト

https://github.com/griddb/griddb_nosql/

■ **GridData Analytics Cloud**

<https://www.griddata-analytics.net/>

■ AWS Marketplace: GridDB Community Edition (CE)

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

■ Twitter <http://twitter.com/GridDBCommunity/>

■ Facebook <http://fb.me/griddbcommunity/>

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。**
 - IoT向けのデータモデルと3つのH(High Performance、High Scalability、High Availability)が特長
- **メトリック、タグを使う時系列DBのKairosDB経由で使うこともできます。**
- **OSSを利用したビッグデータ分析環境も提供しています。**
GridData Analytics Cloud

オープンソースのGridDBを是非とも使ってみてください。

● 本資料に掲載の製品名、サービス名には、各社の登録商標または商標が含まれています。

TOSHIBA

Leading Innovation >>>