

# ビッグデータ

ビッグデータ活用を支える技術

# 目次

## 第1章 ビッグデータを支える技術 クラウド、IoT

1-1.ビッグデータ周辺技術 クラウド .....	9
1-2.クラウドサービスの種類 .....	10
1-3.クラウドサービスのメリット・デメリット .....	11
1-4.クラウドコンピューティングとビッグデータ .....	12
1-5.ビッグデータ周辺技術 IoT .....	13
1-6. IoTとは .....	14
1-7. IoTとユビキタス社会 .....	15
1-8. IoTとM2M .....	17
1-9. IoTを構成する技術要素 .....	19
1-10. IoTを構成する技術要素 センサー .....	20
1-11. IoTを構成する技術要素 デバイス .....	23
1-12. IoTを構成する技術要素 IoTサービス .....	24

# 目次

## 第2章 ビッグデータを支える技術 データ収集・加工

2-1.ビッグデータの解析までの流れ .....	27
2-2.データ蓄積までの流れ .....	28
2-3.データ収集例 .....	29
2-4.同期通信と非同期通信 .....	32
2-5.ビッグデータの処理と保存 .....	33
2-6.ストリーム処理 .....	34
2-7.データ蓄積までの流れ（振り返り） .....	36
2-8.データ管理：クレンジング .....	37
2-8.データ管理：クレンジング具体例 .....	38
2-9.データ管理：ETLとELT .....	39
2-10.データ管理：従来型データ蓄積 .....	40
2-11.データ管理：データレイク .....	41
2-12.スループットとレイテンシ .....	42
2-13.従来型のデータベース：行指向DB .....	43
2-14.従来型のデータベース：列指向DB .....	44
2-15.従来型のデータベース：MPP .....	45

## 目次

### 第3章 ビッグデータコア技術 NoSQL

3-1. NoSQLとは .....	47
3-2. ビッグデータで扱うデータの種類 .....	48
3-3. NoSQLのメリット、デメリット .....	49
3-4. NoSQLの代表的な種類 .....	54
3-5. キー・バリュー型の特徴 .....	56
3-6. 列指向型の特徴 .....	59
3-7. ドキュメント型の特徴 .....	61
3-8. グラフ型の特徴 .....	62
3-9. 代表的なNoSQL製品 .....	63
3-10. NoSQLの基本的概念と技術（マスタ型・P2P型） .....	64
3-11. NoSQL時代の必要要件 .....	65
3-12. NoSQLの基本的概念と技術（整合性） .....	66
3-13. NoSQLの基本的概念と技術（データ分割） .....	73
3-14. NoSQLの基本的概念と技術（ストレージレイアウト） .....	77

## 目次

### 第4章 ビッグデータコア技術 分散処理

4-1.分散処理とは .....	80
4-2.分散処理のメリット・デメリット .....	82
4-3. Hadoopとは .....	83
4-4. Hadoopの基本構成 .....	85
4-5. HDFS (Hadoop Distributed File System) .....	86
4-6. HDFSはマスタ型 .....	87
4-7. HDFSの読み書き .....	88
4-8. MapReduce処理とは .....	89
4-9. MapReduceのアーキテクチャ .....	91
4-10.分散ファイルシステムとリソースマネージャー .....	95
4-11.分散データ処理とクエリエンジン .....	96
4-12.分散データ処理とSpark .....	99

## 目次

### 第5章 ビッグデータにおけるデータ解析

5-1.ビッグデータにおけるデータ解析 .....	101
5-2.アドホック分析ツール .....	102
5-3.ダッシュボードツール .....	103
5-4.データ活用：一般的な統計分析手法 .....	104
5-5.度数分布とヒストグラム .....	105
5-6.平均と標準偏差 .....	106
5-7.正規分布 .....	107
5-8.標本調査と標本平均 .....	108
5-9.相関関係 .....	109
5-10.相関係数 .....	110
5-11.相関関係と因果関係 .....	111
5-12.回帰分析と重回帰分析 .....	112
5-13.多項式回帰曲線のイメージ .....	113
5-14.テキストマイニング .....	114

## 目次

### 第6章 ビッグデータとAI、機械学習

6-1. AIの発展 .....	117
6-2. 機械学習 .....	118
6-3. 教師あり学習と教師なし学習 .....	119
6-4. 機械学習アルゴリズム .....	120
6-5. 画像分析手法 .....	124
6-6. 地図情報システムとの連動 .....	125

## 第1章

# ビッグデータを支える技術

クラウド、IoT

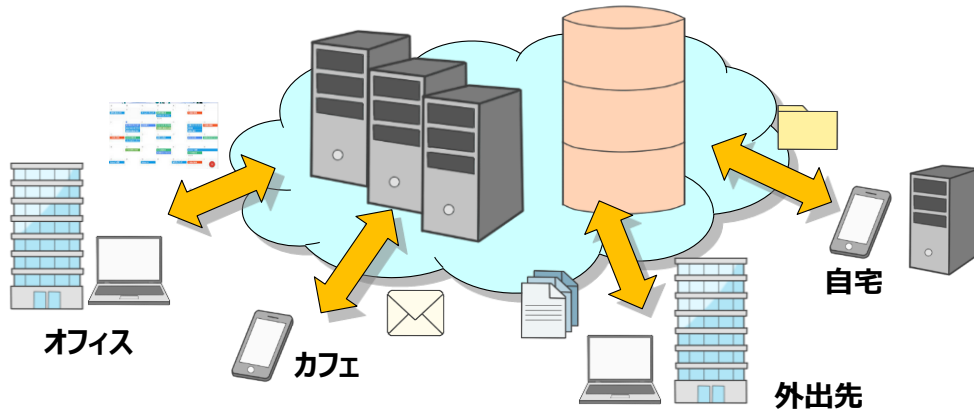


## ビッグデータ周辺技術 クラウド

### クラウドコンピューティングとは

メールやグループウェア、その他様々なサービスをインターネット上で提供し、インターネット上にデータまでも保存するようなサービス形態のことをいいます。

アプリケーションの実行場所が、雲（クラウド）のようにどこにあるのかが分からないモヤモヤとした場所にあることから、このように呼ばれています。



### ・説明の流れ

ビッグデータを支える周辺技術を、学びましょう。

クラウドサービスはビッグデータ登場の背景にも密接にかかわっており、ビッグデータのそもそもの置き場所などでも広く活用されています。

### ・ポイント（絶対に覚えてほしいこと、など）

クラウドはどこに存在するかわからないデータセンター。

データの共有や業務アプリサービスなど様々なサービスを、資産を持たずに提供できる。

### ・質問（問いかけ）

そもそもクラウドサービスはどのような用途で使われ、なぜクラウドと呼ばれているのだろうか。

## クラウドサービスの種類

### クラウドサービスの種類

- SaaS (Software as a Service)  
インターネットを經由してソフトウェアパッケージを提供するサービスのこと。  
アプリをPCにダウンロードしなくても、Webなどのブラウザ上で利用することができる。  
例・メール ・カレンダー ・チャット
- PaaS (Platform as a Service)  
インターネットを經由してアプリの開発・運用環境全体を提供するサービスのこと。システム管理者、開発者向けである。
- HaaS / IaaS (Hardware as a Service / Infrastructure as a Service)  
インターネットを經由してハードウェアや回線などのインフラを提供するサービスのこと。ユーザーはハードウェア資産を所有することなく、仮想サーバーやストレージ (外部記憶装置) を利用することができる。

### ・説明の流れ

クラウドには様々な利用形態の種類があります。  
どのようなものがあるのか見てみましょう。

### ・ポイント (絶対に覚えてほしいこと、など)

SaaSはアプリも提供するソフトウェアサービス。

IaaSは環境を貸し出して好きなようにシステムを構築できるサービス。

### ・補足説明

パブリッククラウドは、一般、不特定多数のユーザーに公開されるクラウド環境。

プライベートクラウドは1社のみに関じたクラウド環境

。

## クラウドサービスのメリット・デメリット

### メリット

・サーバー・ソフトウェアを購入する必要がない

・システム構築期間を短縮できる

・効率的なIT投資やリソース配分を実現できる

・メンテナンスが不要である

・IT部門の負担が軽減される

・カスタマイズが基本的にできない、もしくは難しい

・不特定多数が利用するため、安定して稼働できないリスクがある

・セキュリティー面のリスクがある

・利用するデータ量、時間によっては費用が増加するリスクがある

### デメリット

#### ・説明の流れ

クラウドのメリットとデメリットには、どのようなものがあるでしょうか。

サーバーの調達や、メンテナンスから解放される一方、カスタマイズの難しさや、不特定多数のユーザーの目にさらされる危険もあるため、個人情報の扱いがあるデータの運用には、注意が必要です。

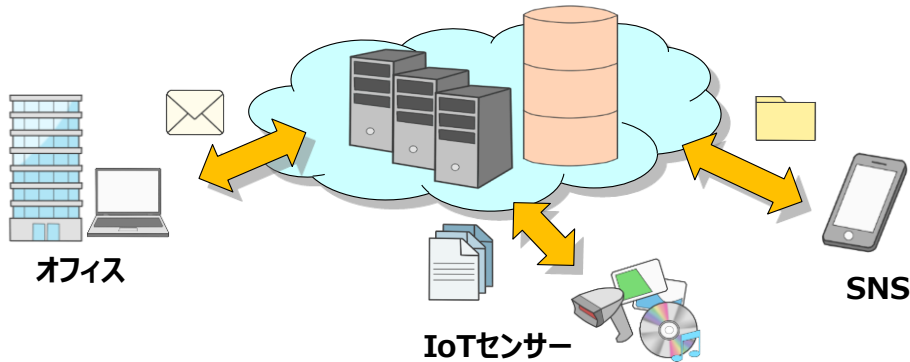
# クラウドコンピューティングとビッグデータ

## クラウドサービスに蓄積されるビッグデータ

クラウドサービスは、ビッグデータの蓄積場所に有力な選択肢です。

実際、以下はビッグデータがクラウドに保存されています。

- ・ GREE、mixi、Facebook、Twitterなどに代表されるSNSのユーザーデータ
- ・ IoTのようにセンサーなどで発生したデータ



### ・説明の流れ

ビッグデータの保存先として、サーバーサイジングを自由に変更できるクラウドサービスは有力な選択肢の一つとなります。

IoTセンサーデータのデータ収集もクラウドサービスベンダーが担う状況も整いつつあります。

## ビッグデータ周辺技術 IoT

### IoTとは

IoT = Internet of Things 「モノのインターネット」

ここで、「モノ」とは、ネットワークに繋がるあらゆる物のことです。

従来はインターネットとは関係のなかったもの、例えば、

- ・眼鏡 ・服 ・時計 ・冷蔵庫 ・電力メーター ・自動車
- ・太陽光パネル ・家 ・スマートフォン

もすべて「モノ」です。

IoTとは、モノがネットワークに繋がれることによって、モノとインターネットが相互に情報交換をできるようになった状態のことをいいます。

### ・説明の流れ

センサーデータを分析する際に耳にするIoTという言葉は何を意味する言葉でしょうか。

### ・ポイント（絶対に覚えてほしいこと、など）

モノが双方向でインターネットに繋がりデータのやり取りができる状態をIoTと呼ぶ。

## IoTとは

- Internet of Thingsという用語は1999年、ケビン・アシュトン（イギリス）によって初めて提唱されました。
- 当初はRFIDによる商品管理システムをインターネットに例えたものでした。
  - 徐々にスマートフォンやクラウドコンピューティングが普及。
  - IoTはモノ自体がインターネットを形作るという環境全体のことを表す概念として捉えられるようになる。
- IDC（ICT市場調査会社）による定義  
「IP接続による通信を人の介在なしにローカルまたはグローバルに行うことができる識別可能なエッジデバイスから成るネットワークのネットワーク」

### ・説明の流れ

IoTという言葉はいつ頃から呼ばれ、定着したのでしょうか。

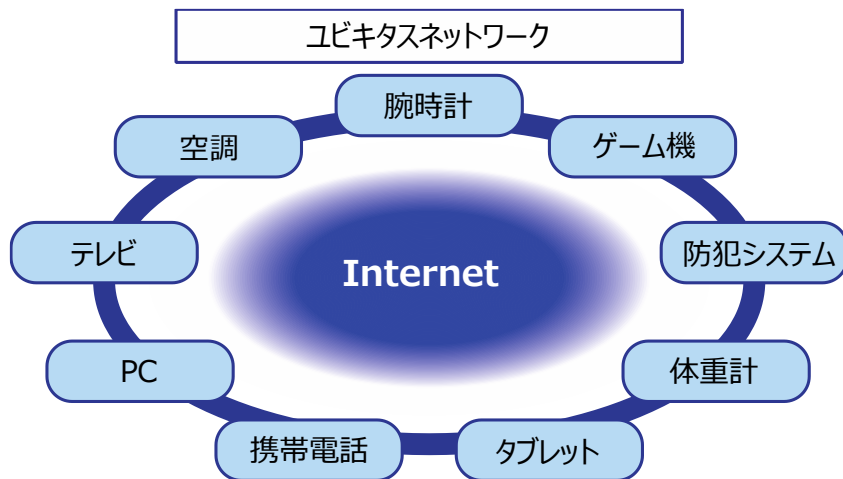
### ・ポイント（絶対に覚えてほしいこと、など）

IoTは2000年頃から呼ばれ、モノ自体がインターネットを形作る広い概念として捉えられている。

## IoTとユビキタス社会

### ユビキタスネットワークとは

いつでもどこでもインターネットを利用できるという概念のことをいいます。  
1988年ゼロックス社パロアルト研究所のマーク・ワイザーにより提唱されました。



### ・説明の流れ

2000年初頭から現れたユビキタスネットワークと  
は何でしょうか。また、IoTとの違いは何でしょうか。

### ・ポイント（絶対に覚えてほしいこと、など）

ユビキタスネットワークの中心は人であり、IoTの  
モノとモノとの相互制御とは異なる概念。

## IoTとユビキタス社会

### IoTとユビキタスの違い

- ・ IoT : モノとモノが相互に制御し合っている状態を表す。  
…「モノ」を中心とした概念
- ・ ユビキタス : ユーザーが時間や場所にとらわれずインターネットに繋がって様々なサービスを受けられる状態を表す。  
…ユーザーという「人」を中心とした概念

#### 参考

**ユビキタス** (ubiquitous) は、遍在 (いつでもどこでも存在すること) をあらわす言葉。

パロアルト研究所のマーク・ワイザーが、1991年の論文『The Computer for the 21st Century』にて、コンピュータやネットワークなどの遍在を表す意味合いで用いた。以来、ユビキタスコンピューティングやユビキタスネットワーク、さらにはそれらが当たり前になった社会を指す「ユビキタス社会」の意味で用いられるようになった。

Wikipedia

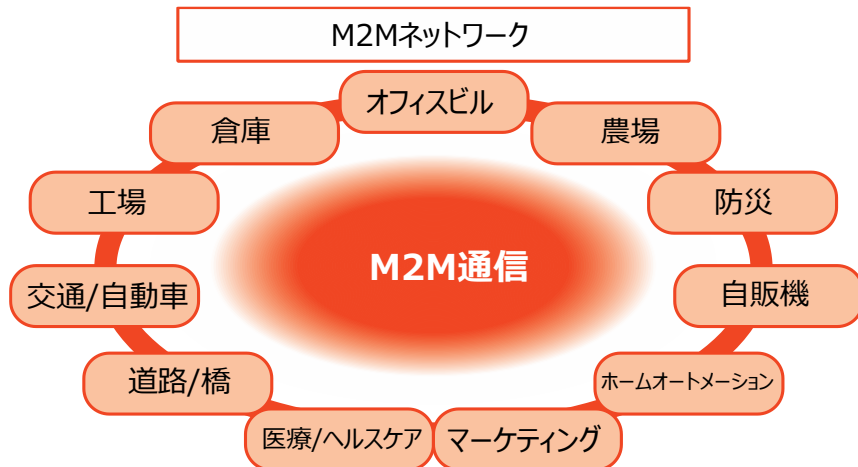
- ・ 説明の流れ  
前のスライドを補足する形で、ユビキタスについて説明を加える。



## IoTとM2M

### M2M = Machine to Machine

機械同士が、人を介在しないで情報をやり取りするシステムのことをいいます。



#### ・説明の流れ

M2MとIoTの違いは何でしょうか。M2Mは人が介在しない機械同士の連携です。

IoTは人も介在するセンサーも対象となるだけでなく、結果を見るのも人である場合もあるため、M2Mの概念だけではIoTを説明することはできません。

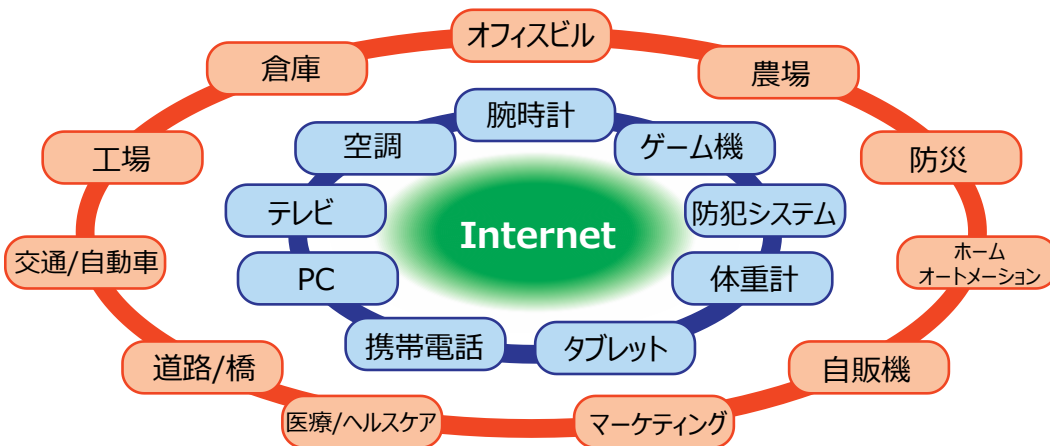
#### ・ポイント（絶対に覚えてほしいこと、など）

M2Mは人が介在しない機械同士の連携

## IoTとM2M

IoTはユビキタスとM2Mを包括する概念といえます

IoT社会  
Machine to Machine、Human to Machine、Human to Human



### ・説明の流れ

IoTはユビキタスとIoTを合わせたような概念です。  
人とモノ、モノとモノ、人と人を繋ぐ総合的な社会  
インフラになる可能性を秘めています。

## IoTを構成する技術要素

- ・センサー：物理的な現象を検知し、電気信号として出力する装置。
- ・デバイス：センサーが組み込まれることによって、ネットワークに接続された装置やモノ。  
例 スマホ、時計、メガネ
- ・ネットワーク：デバイスをIoTサービスに繋ぐ、あるいはデバイス同士を繋ぐことでデータを共有、処理するシステム。
- ・IoTサービス：①デバイスとのデータの送受信 … IoT  
②データの処理と保存 … ビッグデータの技術的守備範囲を行うサービス。
- ・データ分析：蓄積したデータについて統計分析や機械学習を行う。  
→最適な判断や行動方針を導き出す。

### ・説明の流れ

IoTとは何なのでしょう。まず、技術要素から切り込んでみましょう。

IoTには、基本的に、センサー、デバイス、ネットワーク、サービス、データ分析がセットになってIoTサービス全体を形作っています。

それぞれの技術要素は独立していることが多く、技術要素の組み合わせによって、IoTを実現しています。

データの処理と保存はまさにビッグデータの技術的守備範囲となり、IoTには欠かせない技術要素です。

### ・ポイント（絶対に覚えてほしいこと、など）

IoTの実現にはビッグデータ技術が必要。逆にIoTを知ることでビッグデータの現実を知ることになる。

## IoTを構成する技術要素 センサー

### センサー

物理的な現象を検知し、電気信号として出力する装置のことをいいます。  
多くの場合、一つのデバイスに対して複数のセンサーが埋め込まれています。

- ・ 画像センサー：光を捉えて処理することで、画像や動画を撮影する。  
赤外線を検知して画像処理するものもある。
- ・ 光センサー：光の強度を測定する。
- ・ 温度センサー：温度を測定する。
- ・ 湿度センサー：湿度を測定する。
- ・ 振動/速度/加速度センサー：機器の振動や速度、加速度を測定する。
- ・ 地磁気センサー：地磁気を検出することで、方角を計測する。
- ・ ジャイロセンサー：デバイスの傾きを検知する。
- ・ 音声マイク：機器が発する音や、人の声などの音声を収集する。

### ・ 説明の流れ

IoTのセンサーにはどのようなものがあるか見てみましょう。

データの中には、画像データや、音声データのよ  
うな非構造データや準構造化データ、バイナリーデー  
タが存在します。

## IoTを構成する技術要素 センサー

センサーの代表的なデータフォーマットとしては、  
・XML ・JSON ・MessagePack  
があります。

### XML

```
<xml>
  <info>
    <id>12996</id>
    <name>RoomSensor</name>
    <date>20170123112255</date>
  </info>
  <data>
    <temperature>27.8</temperature>
    <humid>72</humid>
  </data>
</xml>
```

人が読んで分かりやすい  
データ量が多い

### JSON

```
{
  "info": {
    "id": 123,
    "name": "RoomSensor",
    "date": "20170123112255"
  },
  "data": {
    "temperature": 27.8,
    "humid": 72
  }
}
```

データ量が少ない

いずれも各言語のライブラリが充実していますが、文字データであることから、  
パース（解析）をしないとプログラムで利用できません。  
MessagePackは、バイナリデータをそのまま扱いたい場合、有利です。

### ・説明の流れ

センサーデータの基本的なフォーマットを見てみましょう。

タグ構造のXMLや、Webサイトでよく使われるJSON形式もありますが、MessagePackのようなバイナリーデータも存在します。

### ・ポイント（絶対に覚えてほしいこと、など）

センサーデータは、構造化データでないことが多い。

## IoTを構成する技術要素 センサー

### MessagePack

- ・センサーの代表的なデータフォーマットの一つ。
- ・JSONと似た形式だが、値はバイナリのみである。
- ・軽量でプログラム間処理に向いている。

### MessagePackの特徴

- ・シリアライズ（データの直列化）、デシリアライズ：非常に高速
- ・シリアライズされたデータのサイズ：小さい
- ・フォーマット定義：不要
- ・ストリーム処理：可能

### JSONとMessagePackの比較

JSON `{"a":null,"b":10,"c":[20],"d":"30"}` … 35byte

→ MessagePack に変換すると…

`84 a2 61 c0 a2 62 0a a2 63 91 14 a2 64 a2 33 30` … 16byte

- ・説明の流れ

MessagePackの形式を見てみましょう。

人が見て理解できる形式ではありませんが、データ量が小さく、変換が高速であることが魅力です。

- ・ポイント（絶対に覚えてほしいこと、など）  
IoTデータにはバイナリーデータも存在する。

## IoTを構成する技術要素 デバイス

### デバイス

センサーが組み込まれることによって、ネットワークに接続された装置、モノのことをいいます。例えば、スマホ、時計、メガネはいずれもデバイスです。

### デバイスの2つの機能

- ・ センシング：センサーを利用して、デバイス自身や周りの環境の状態を収集し、IoTシステムに通知すること。
  - 例・画像センサーによる人の有無の検知
  - ・スマホの位置情報や加速度の計測
- ・ フィードバック：システムからの通知を受け、指示や動作をもとのシステムに返すこと。次のような方法がある。
  - ・ 可視化：センシング結果表示、デバイスの管理、画面表示
  - ・ 通知：システムが判断した結果を画面に表示
  - ・ 制御：デバイス自身や環境の状態そのものを変更

### ・ 説明の流れ

IoTにおけるデバイスとは何でしょうか。

スマホやウォッチに搭載されていて、センサーとネットワーク通信機能もデバイスとして捉えられます。

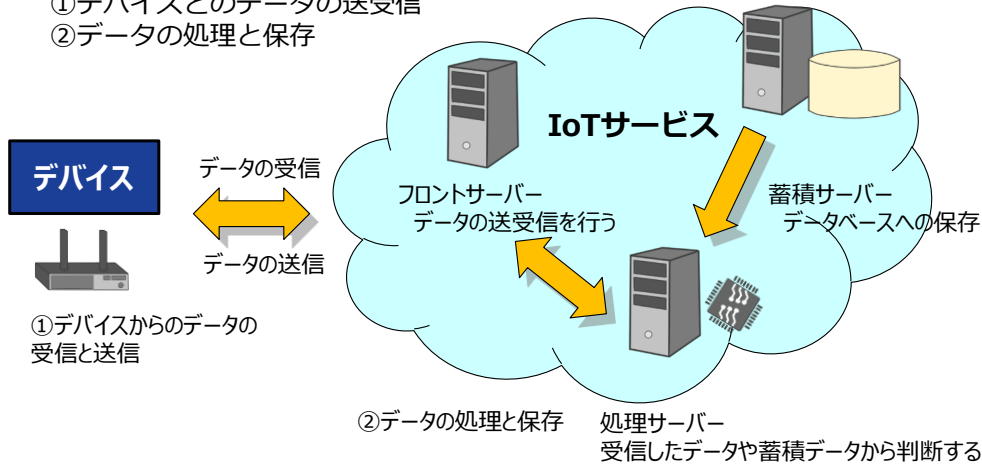
センサーとネットワーク接続を併せ持った機器を指します。

# IoTを構成する技術要素 IoTサービス

## IoTサービス

以下を行うサービスのことを指します。

- ①デバイスとのデータの送受信
- ②データの処理と保存



### ・説明の流れ

IoTサービスはデバイスから送られてきたデータを受信したり、蓄積、分析したり、サービスを行ううえで必要な処理を行います。

ビッグデータを処理することも役割の一つになります。

### ・ポイント（絶対に覚えてほしいこと、など）

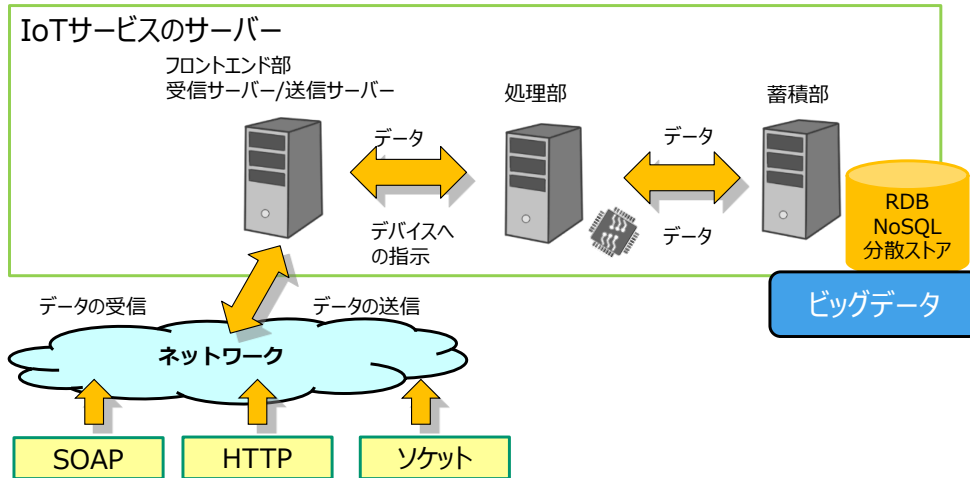
IoTサービスでは、バックエンドでビッグデータも扱う。



## IoTを構成する技術要素 IoTサービス

### サーバー構成

IoTサービスの役割は、フロントエンド、処理、蓄積の大きく3つに分けられます。蓄積されるデータは膨大な量となります。



### ・説明の流れ

IoTサービスをもう少し細かく見てみましょう。

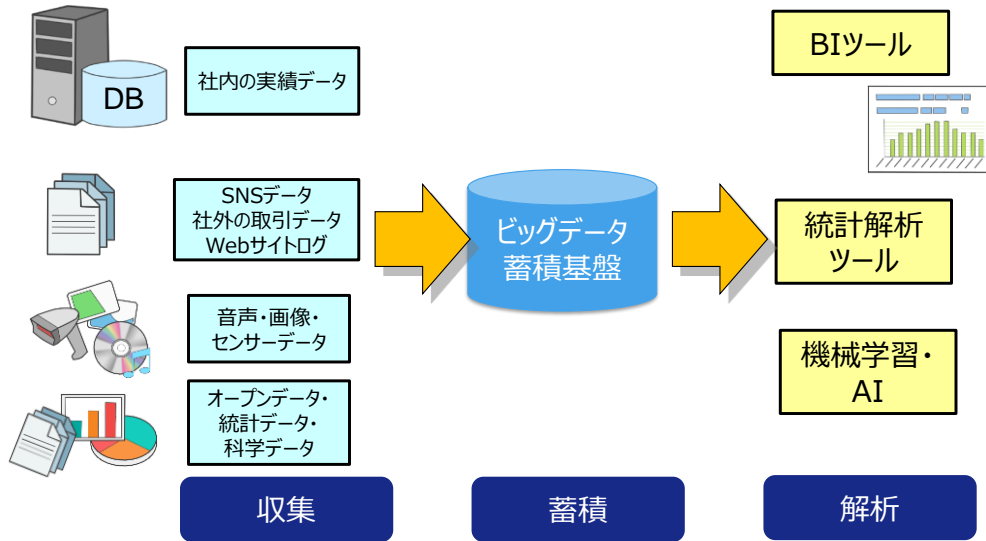
サービスの奥にある蓄積部では、NoSQLや分散処理システムなど、非構造化データを大量に処理できる基盤が利用されていることが分かります。

## 第2章

# ビッグデータを支える技術

## データ収集・加工

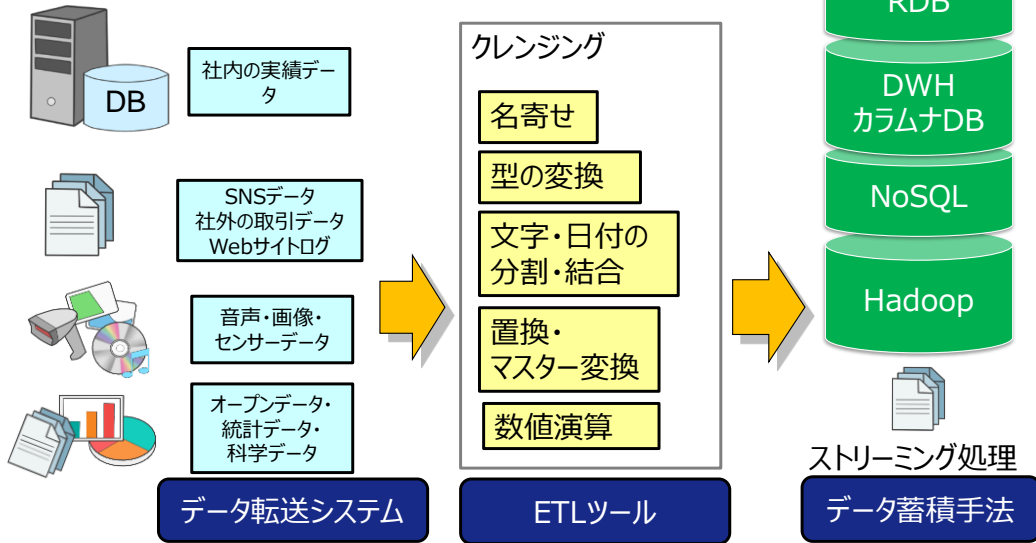
## ビッグデータの解析までの流れ



- ・説明の流れ  
ビッグデータの収集から解析までの流れを見てみましょう。  
解析部分には、BIツール（ Business Intelligence ） や統計ツール、機械学習、AIなどが利用されています。
- ・ポイント（絶対に覚えてほしいこと、など）  
ビッグデータ利活用には、蓄積だけでなく、前処理とアウトプット部分が必要。

## データ蓄積までの流れ

以下のようにして、ビッグデータは分析可能な状態に処理されます。



### ・説明の流れ

データを蓄積する部分までを詳しく見てみましょう。

データを蓄積据える前にデータをIoTサービスまで転送する仕組みが必要になります。

また、データを活用するにあたり、そのままでは使えないデータが混入する場合もあり、これらを正規のデータ、もしくは分析で扱いやすい易い形に加工する処理が必要になります。

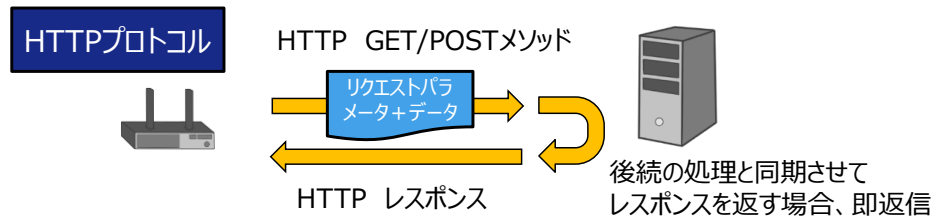
この加工処理をクレンジングと呼びます。

## データ収集例

### データ収集と通信方法

IoTの受信サーバの通信方法は3種類に分類できます。

- HTTPプロトコル  
通常のWebシステムと同様、HTTPプロトコルを利用したWeb APIを利用してデバイスからアクセスを行う。
- WebSocket  
音声や動画のリアルタイム通信を行う。
- MQTT  
送受信を媒介する第三者の存在により、柔軟な通信を可能にするメッセージ・キュー方式を利用する。



### ・説明の流れ

データ転送には3つの方法があります。HTTPプロトコルは最も単純ですが、通信は一方的です。

WebSocketは双方向通信が可能ですが、通信を確立するためのアプリが双方に必要になります。

MQTTはメッセージを媒介する、ブローカーが間にあれば、柔軟にデータのやり取りを実現することが可能です。

HTTPプロトコルは通信は、Webシステムと同じHTTPプロトコルを使うため単純ですが、デバイス側からの通信を待つ必要があります。

サーバーからの制御も通信があった一回限りになります。

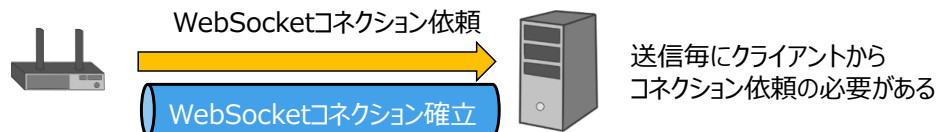
## データ収集例

### WebSocket

WebブラウザとWebサーバー間でデータを双方向かつ連続的に送受信するための通信プロトコルです。IoTでWebSocketを用いることで、インターネット上でソケット通信が可能になります。

HTTPプロトコルを用いると送信毎に接続の依頼が必要ですが、WebSocket通信を行うと、接続を継続したままにすることが可能となります。

#### HTTPプロトコルの場合 WebSocketコネクション確立



#### WebSocket通信



#### ・説明の流れ

WebSocketの通信イメージを見てみましょう。

#### ・ポイント（絶対に覚えてほしいこと、など）

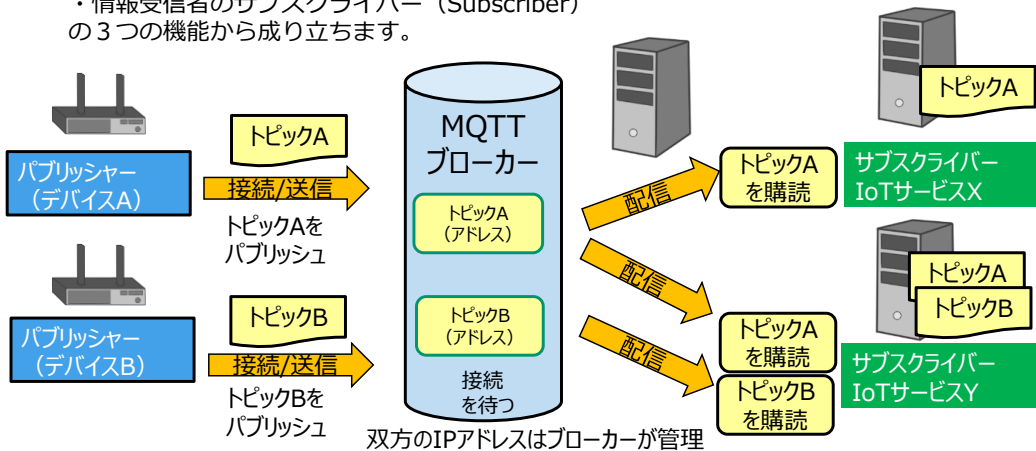
WebSocket通信は双方向の通信に強いいため、画像などの大きなデータ通信にも向いている。

## データ収集例

### MQTT (MQ Telemetry Transport)

IBM社により提唱され、現在はオープンソースとなっているプロトコルです。

- ・ 仲介役のブローカー (Broker)
  - ・ 情報発信者のパブリッシャー (Publisher)
  - ・ 情報受信者のサブスクライバー (Subscriber)
- の3つの機能から成り立ちます。



#### ・ 説明の流れ

MQTTはIoTに特化したメッセージ・キューシステムです。

メッセージブローカーが、センサーデバイスと、IoTサーバーの間に立ち、データの配信を行っています。

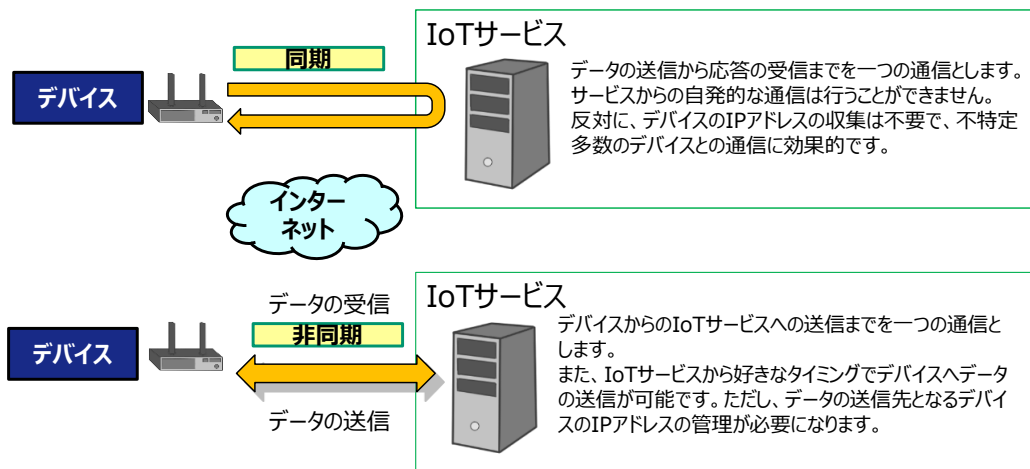
#### ・ ポイント (絶対に覚えてほしいこと、など)

MQTTは非同期通信が可能。

障害があった際のメッセージの送り直しなど、多彩な機能が用意されている製品もある。

## 同期通信と非同期通信

デバイスとの通信には同期通信と非同期通信があります。



### ・説明の流れ

デバイスとの通信には同期通信と非同期通信があります。

同期通信の場合、デバイスからの通信に対してレスポンスを送るため、デバイスのIPアドレスを知る必要があります。

非同期通信の場合は、サーバーから好きなタイミングでデバイス制御を行えますが、デバイスのIPアドレスを管理する必要があります。



## ビッグデータの処理と保存

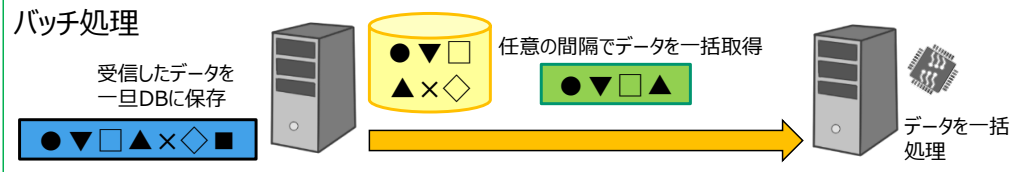
### ビッグデータの処理

受信したデータは、データベースや分散ファイルシステムなどに保存されます。また、受信したデータからデバイス制御の判断を行います。

処理と保存の方法には、ストリーム処理とバッチ処理の2種類があります。



データ内容を常に判断し、即座にデバイスを制御したい場合に利用



記録とデバイス制御にタイムラグがあっても問題ない場合に利用

### ・説明の流れ

IoTやビッグデータ処理では、データを一括で処理するバッチ処理と流れてくるデータを蓄えず、リアルタイムで処理していくストリーミング処理に分かれます。

### ・ポイント（絶対に覚えてほしいこと、など）

即時応答が必要かどうか、要件によっていずれかを選択する必要がある。

例えばセンサーデータを監視し続け、故障、危険などにより異常値を検出した場合は、アラートを出したり、関連機器に指令を出して調節などをする必要がある場合は、ストリーム処理が適しています。

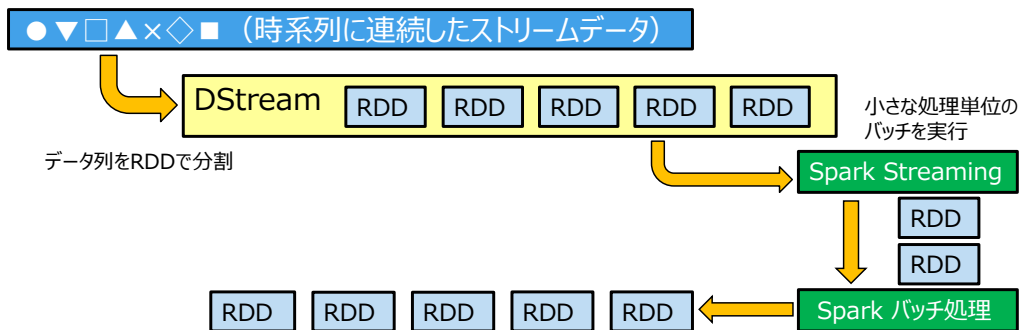
## ストリーム処理

### ストリーム処理

ストリーム処理はデータを保存せずに、処理サーバーに到着したデータを逐次処理する方法です。与えられたデータにリアルタイムで反応することができます。

### Spark Streaming

Spark Streamingは、ストリーム処理を行うためのSparkのライブラリです。時系列的に連続したデータ列をRDDで分割し、分割されたRDDに対して小さな処理単位のバッチを実行します。



### ・説明の流れ

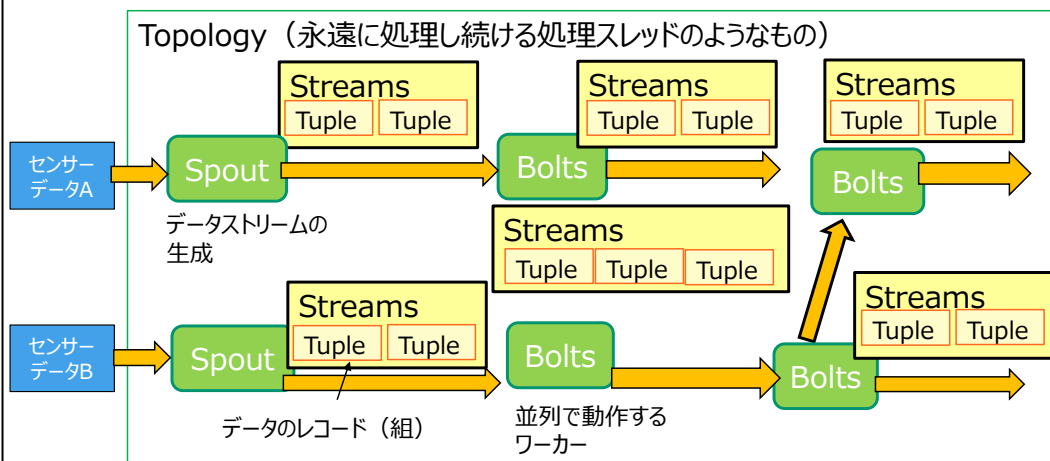
ストリーミング処理の技術にはどのようなものがあるでしょうか。

Spark StreamingはRDDという小さな処理単位に分けることで常に処理を流し続けることが可能です。

# ストリーム処理

## Apache Storm

Apache Stormとは、ストリーム処理のためのフレームワークです。



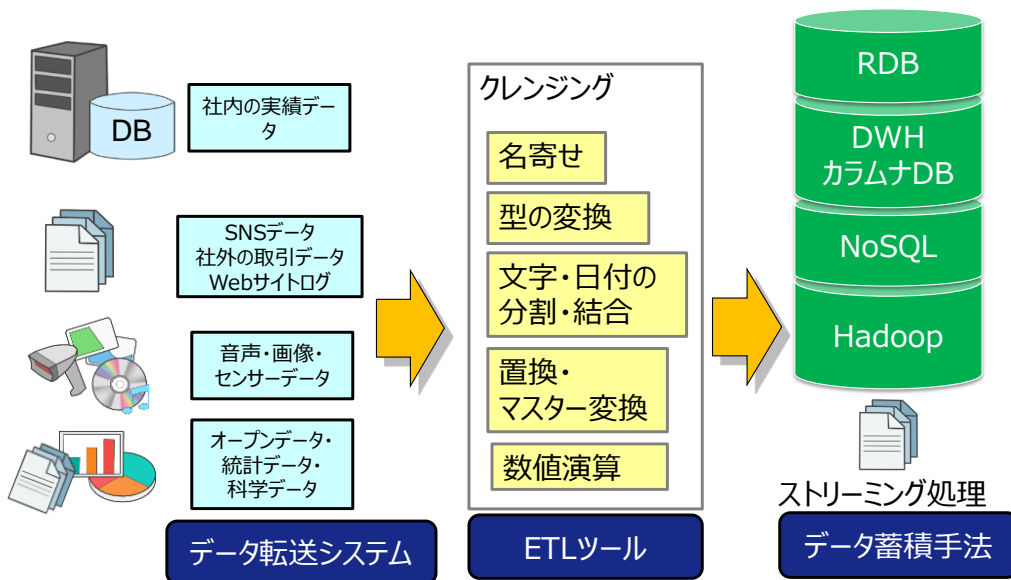
### ・説明の流れ

ストリーミング処理で有名なフレームワークとしてApache Stormがあります。

Spoutから入ってきたデータをタプルというデータ単位に分けBoltsというワーカーで順次処理をしていきます。

## データ蓄積までの流れ

ビッグデータを分析可能な状態に処理する流れは以下のようになっています。



### ・説明の流れ

次にデータ加工、クレンジングの部分を見ていきましょう。

このページは「データ蓄積までの流れ」の振り返りです。

## データ管理：クレンジング

### クレンジング

データの整理や加工を行うことで、効率的に分析できるようになります。これをクレンジングもしくはデータ加工と呼びます。専用のツールやR、Python、GOなどのスクリプト言語で実施できます。

クレンジング対象	具体例	対処例
型の統一、日付、数値など	数値演算を行いたいデータに文字が入る場合など	基準を用意し、基準に合わないデータ修正
書式の不適合・表記ゆれ	住所、会社名、電話番号、郵便番号などの表記の不統一、通貨や数値の単位、文字コードなど	表記方法や単位の基準を定義し、基準に従うようにデータを修正する
異常データ	ある条件下でのセンサーの誤動作、データの無記入による空データ、データ入力ミスによる意味のないデータ	回帰やクラスタリングにおいて、結果の精度を下げる原因に異常値を検知し、事前に外す
個人情報およびプライバシー情報の保護	氏名、住所、メールアドレスなど個人を直接特定できる情報	個人を直接特定できる情報を削除または匿名化処理を施す

### ・説明の流れ

データクレンジングでは、日付の形式の統一、データの分割、結合、マスターデータに紐づけなど、要件に応じたあらゆる加工処理を行います。

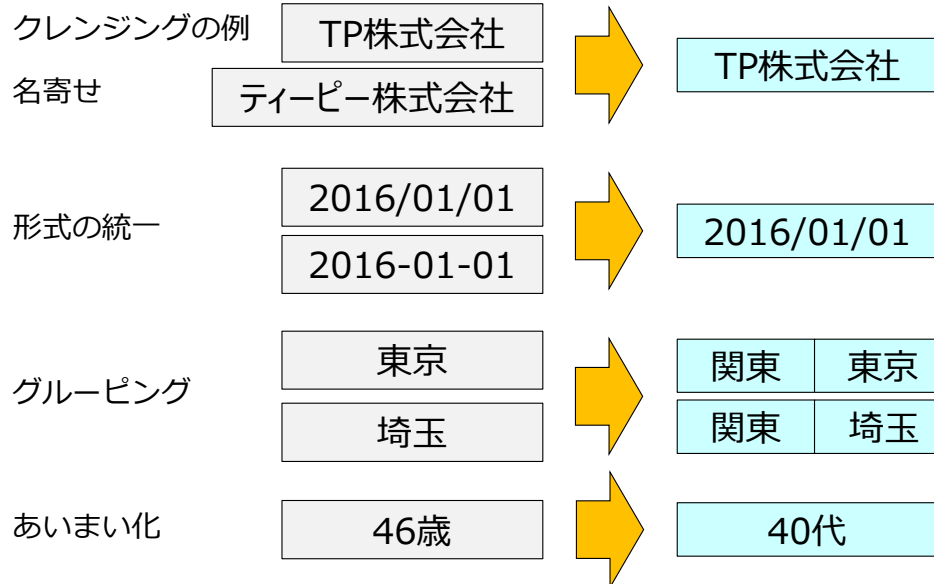
後の工程に回すと処理が複雑になったり、分析の処理速度が低下することが予想される場合は、極力前処理でデータをきれいにする（クレンジングする）事が重要となります。

### ・ポイント（絶対に覚えてほしいこと、など）

データの欠損、異常値などは分析結果に影響することがあるため、入り口で除去できるのであれば、除去した方が良い場合がある。

ただし、分析の目的が異常値検出などである場合は、除去してはいけない。

## データ管理：クレンジング具体例



- ・説明の流れ

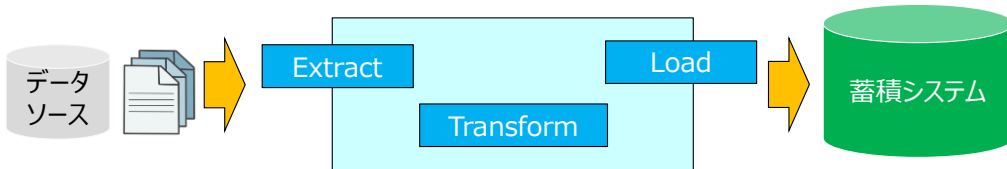
具体的なクレンジングの例を見てみましょう。

名寄せや形式変更、グルーピングといった従来から使われている変換の他、ビッグデータでは匿名化処理もクレンジングでの重要な加工処理となります。

## データ管理：ETLとELT

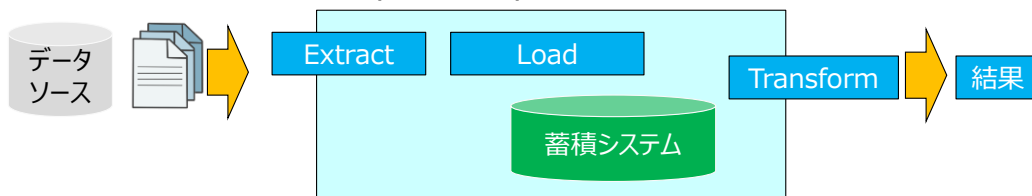
### ETL (Extract Transform Load)

収集したデータを抽出(Extract)した後、利用しやすいように加工(Transform)し、DWHなどの蓄積先に書き込む(Load)一連の処理を表します。大量データをバッチ処理する商用のETLツールなどが存在します。



### ELT (Extract Load Transform)

収集したデータを抽出(Extract)し、DWHなどの蓄積先に書き込み(Load)をした後、利用する時に初めて加工(Transform)を行う方式です。



### ・説明の流れ

データ加工は、処理の流れからETL (Extract Transform Load) と呼ばれます。

データを抽出し、加工、DBなどにロードという流れです。

一方でビッグデータの場合は、収集時点では何に使うか決めず、蓄積してから利用する内容を決めることも多いのが現状です。

その場合は、Extractしてから蓄積環境にLoadし、使う際にTransformするという順番になります。

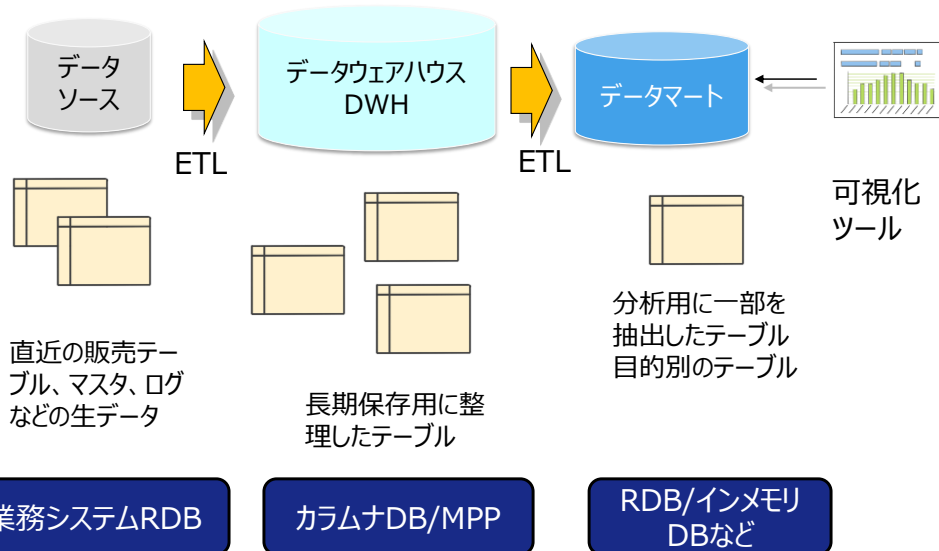
この処理方式をELTと呼びます。

### ・ポイント (絶対に覚えてほしいこと、など)

ETLとELTの違いを把握しましょう。場合によっては、ETLも使い、ELT処理もするケースもある。

変換するツールはどちらも同じツールになる場合も異なる場合もある。そのため、要件に応じた選定をする必要がある。

## データ管理：従来型データ蓄積



### ・説明の流れ

ビッグデータ以前に一般的に利用されているデータ蓄積方式を見てみましょう。

データウェアハウスは、データを長期に渡って蓄積するデータの倉庫として活用されてきました。

構造化DBであるカラムナデータベースや、RDBなどで運用されます。BIツールからのアドホック（随時）集計要求に応えるため、目的別に抽出したデータマートという結合済みテーブルを小さい規模のDBに入れて運用する形式も良く採用されます。

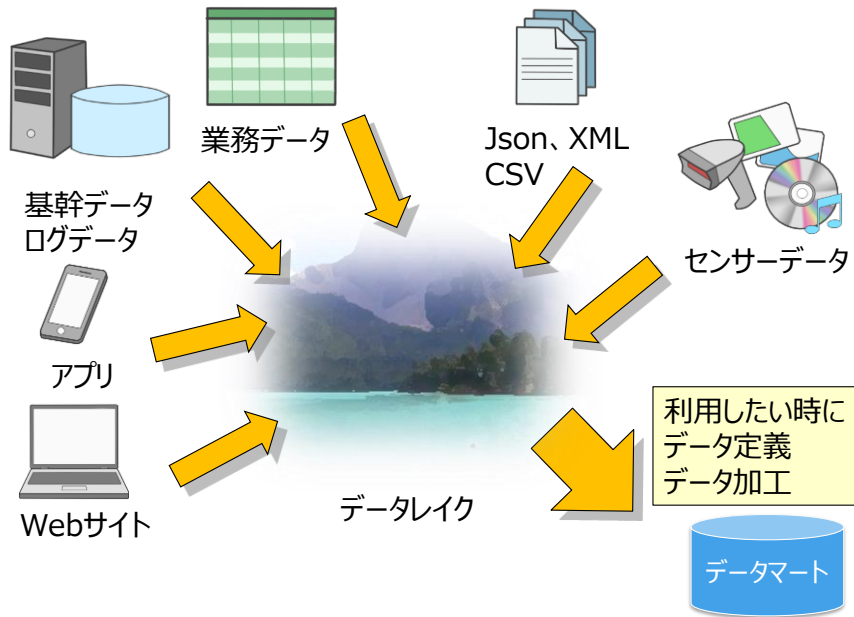
一方で、データ発生源となる基幹データベースは、日中業務で利用されているため、負荷をかけないようにDWH、データマート、BIからは直接アクセスさせない運用が一般的です。夜間、DWHにデータを移行します。

### ・ポイント（絶対に覚えてほしいこと、など）

基幹DBと、DWH、データマートの違いを理解する。



## データ管理：データレイク



### ・説明の流れ

DWHが構造化データを扱い、最初から構造を決めないと構築すらできないのに対し、ビッグデータは既にある情報を利用したり、そもそも構造化できないデータの場合が一般的です。

そのため、とりあえずデータを蓄積しておいて、利用する段になったら初めて構造を定義しようという発想が生まれました。

この、データをとらずに貯めこんで後で利用する形態を、データレイク、データの湖と呼びます。

### ・ポイント（絶対に覚えてほしいこと、など）

データレイクでは、データ構造の定義をいつ行うかを押さえておく。

## スループットとレイテンシ

### ビッグデータ処理性能を測る指標

- ・スループット  
一定時間に処理できるデータの総量。  
WHやデータレイクなど、データの量が多い処理を行う時に重視。
- ・レイテンシ  
データ処理が終わるまでの待機時間。  
アドホック集計、データマート、BIツールなどで重視。

スループットとレイテンシは両立しないことが多く、複数のシステムで役割を分担することがあります。

### ・説明の流れ

ビッグデータシステムの性能を図る指標は何でしょうか。

それは、処理の早さか、単位時間に処理するデータ量かということです。それらを前者はレイテンシ、後者はスループットと呼びます。

一見似たような指標のようですが、内容が異なりますのでご注意ください。

### ・ポイント（絶対に覚えてほしいこと、など）

レイテンシはアドホック集計能力、スループットは夜間のバッチ処理能力に関わる。

## 従来型のデータベース：行指向DB

### 行指向データベース

テーブルの行を一つのデータとしてディスクに保存する方法です。

追記は末尾に加えるだけなので容易です。  
トランザクションが多く発生する業務アプリなどで利用されます。



ディスク格納イメージ

2017-01-01	商品A	5000	10
2017-01-02	商品B	3750	1
2017-01-03	商品C	2160	5

高速化のためにIndexを利用することがありますが、分析対象となる列がわからないため、ディスクI/Oの軽減にならない場合が多いです。

### ・説明の流れ

RDB、リレーショナルデータベースは、ビッグデータ集計に向くのでしょうか。データ保存の仕組みから紐解いてみましょう。

行指向データベースでは、データの格納を行単位で行うため、列ごとの集計では、必要のない項目（カラム）も読み込んでしまい、高速に集計できない場合が多いです。

行指向データベースは追記や更新（内部的には追記）は高速なため、エントリー業務と参照業務で主に利用されます。

### ・ポイント（絶対に覚えてほしいこと、など）

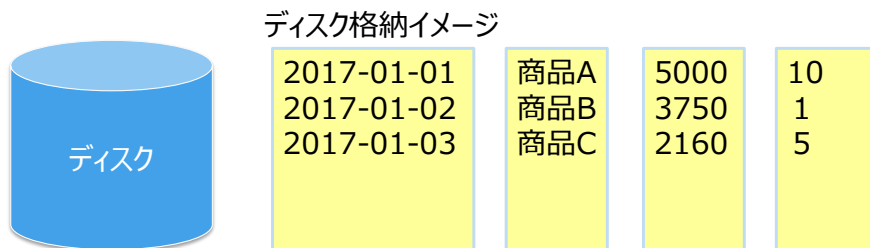
RDBに代表される行カラムDBは、入力業務には強いが、集計業務には向かない。

## 従来型のデータベース：列指向DB

### 列指向データベース

テーブルのデータを列単位にまとめて保存する方法です。

集計に必要な項目だけを読み込むことができ、ディスクI/Oの軽減ができます。



また、カラム単位での重複は集約が容易なため、圧縮効率が高いです。

### ・説明の流れ

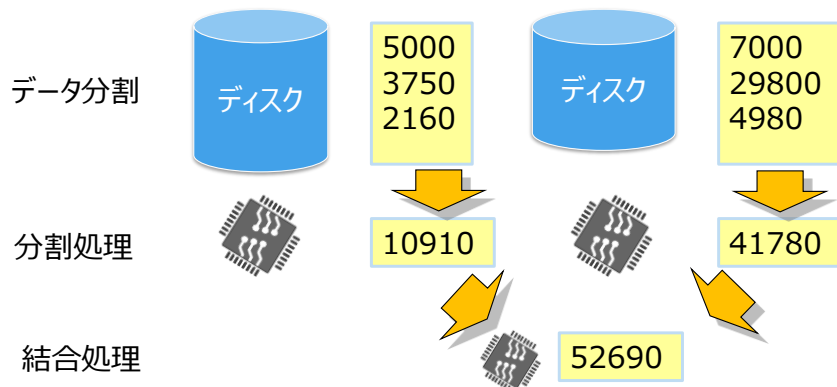
列指向データベースは列単位でデータを保存するため、集計は高速になります。

しかしながら、1台での運用を想定する場合が多く、ビッグデータの規模での運用は難しいことと、仕組み上構造化データしか扱えないため、ビッグデータの分野での活用は進んでいません。

## 従来型のデータベース：MPP

### MPPデータベース (Massively Parallel Processing)

一つのクエリを多数の小さなタスクに分解し、多くのCPUコアやコンピュータを並列的に稼働することで結果を得る手法です。



#### ・説明の流れ

列指向DBや行指向DBを複数台用意し大量のCPUで並列処理する、MPPデータベースが現れ、より大規模のデータを扱えるようになりました。

しかしながら扱うデータは構造化データであり、ソフトも高額であるため、Hadoopのような安価なサーバーを大量に配置することはできません。

そのため、データレイクのようなとりあえず貯めるという発想での運用は難しいと言えます。

## 第3章

# ビッグデータコア技術

## NoSQL

ビッグデータとその周りの技術の関係について学ぶ。

周辺技術ごとにビッグデータとどのように関連しているのか理解する。

## NoSQLとは

### NoSQLとは

- Not only SQL = RDBMS以外のデータベースの総称
- SQLを否定するものではない。
- 1998年にCarlo Strozzi氏が初めて名称を使用した。(NoREL)
- 非構造化データの蓄積や管理に利用される。

### BASE

- Basically Available Soft-state Eventual consistency
- 処理速度を優先するNoSQLで採用されているトランザクションの考え方。  
…「稼動が第一で、厳密な整合性は過程ではなく結果でのみ重要視する」

### ・説明の流れ

NoSQLはビッグデータシステムの話題でよく耳にする言葉です。

NoSQLのNoとは何を指すのでしょうか。

### ・ポイント（絶対に覚えてほしいこと、など）

NoSQLはSQLを否定する言葉ではない。

むしろNoSQLでもSQLを扱える製品もある。

## ビッグデータで扱うデータの種類

ビッグデータの種類	データの例
構造化データ	データベースに格納されたデータ など (顧客テーブルデータ、 受注テーブルデータ など)
準構造化データ	ログデータ、センサーデータ、SNSに書き込まれた データ など
非構造化データ	文書、音声、動画、画像 など

- ・ 説明の流れ  
ここでビッグデータの種類をおさらいしましょう。



## NoSQLのメリット、デメリット

### NoSQLのメリット

- ・ 前もったデータの構造の定義が不要で、柔軟な変更が容易である
- ・ 特定の形式に固執しないため、複数のサーバにデータの分散ができる
- ・ データ構造が単純で、更新や検索処理の速度が速い
- ・ 基本的にPutとGetのみを行えばよく、JOINがないため操作が明快

### NoSQLのデメリット

- ・ RDBMSと比較すると、データ間の整合性を維持する機能が弱い

### ・ 説明の流れ

NoSQLのメリットとデメリットを整理しましょう。

メリットは、データ構造の変更のしやすさなどです。

デメリットはDB間の整合性の維持の難しさです。

## NoSQLのメリット、デメリット

### NoSQLとRDBMSとの比較

	RDBMS	NoSQL
保存に適するデータ	構造化データ	非構造化データ
スキーマ定義	事前定義が必要 固定的で変更しにくい	事前定義不要 データ構造を変更しやすい
データの整合性	同時実行制御がある データ整合性を重視	比較的緩い 大容量データの高速処理を優先、結果整合重視
データ操作命令	SQL言語 (SELECT、INSERT、 UPDATE、DELETE)	Put(書き込み) Get(読み出し)
拡張方式	スケールアップ	スケールアウト

### ・説明の流れ

NoSQLとRDBMSを比較した資料です。構造定義とデータの整合性の扱いで大きく特徴が異なります。

## NoSQLのメリット、デメリット

### データベースの拡張方式の比較

RDBMS : スケールアップを行う

NoSQL : スケールアウトを行う

### スケールアップ

1台のサーバの処理性能やストレージを拡張します。

### スケールアウト

サーバを増設し処理性能を向上させます。

### ・説明の流れ

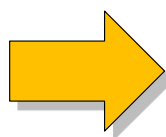
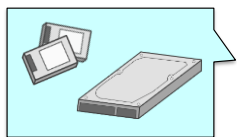
データベースの拡張方式の違いをみてみましょう。

スケールアップは、1台のサーバ処理性能のアップで処理能力を向上させるのに対し、スケールアウトでは、サーバ台数を増やすことにより処理性能を向上させます。

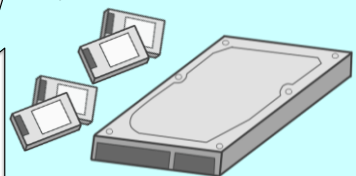
## NoSQLのメリット、デメリット

データベースの拡張方式の比較

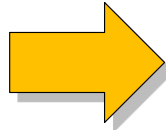
RDBMS → スケールアップ



より大容量のストレージ



NoSQL → スケールアウト



- ・説明の流れ

データベースの拡張方式の違いをみてみましょう。

スケールアップは、1台のサーバー処理性能のアップで処理能力を向上させるのに対し、スケールアウトでは、サーバー台数を増やすことにより処理性能を向上させます。

## NoSQLのメリット、デメリット

### スケールアップ

ハードウェアの性能(ディスク、CPU、メモリ)の増強により、特定の1台のサーバの性能を向上させます。

RDBは以下の特徴を持つため、スケールアップが効果的です。

- ・複数のサーバで運用すると、データ分割や配置作業が必要となる
- ・サーバ間の大規模トランザクションによりパフォーマンスが低下する

### スケールアウト

サーバの台数を増やし、全体での処理能力を向上させます。

高性能がサーバに求められないため、低コストで処理性能を上げることが可能です。

また、1台サーバの停止による影響は少なく、稼働率は向上します。

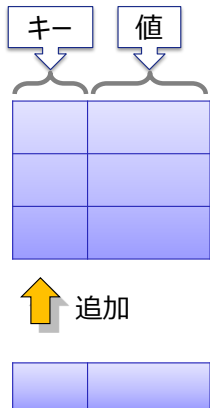
NoSQLは、データベースで扱うデータが非構造化データなため、複数のサーバへの分割・複製が容易、つまりスケールアウトの手法に向いています。

### ・説明の流れ

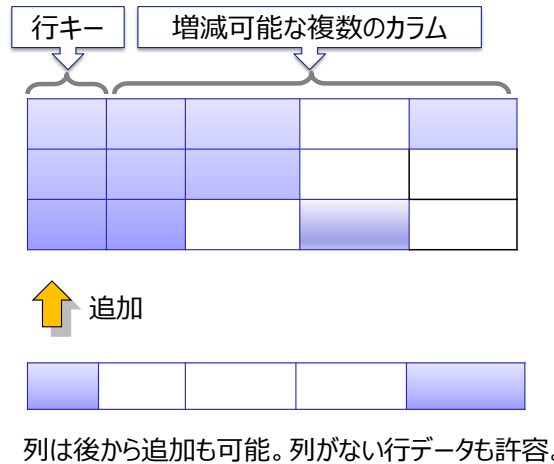
データベースの拡張方式の違いを詳しくみてみましょう。

## NoSQLの代表的な種類

### キー・バリュー型



### 列指向型 (ワイドカラム型)



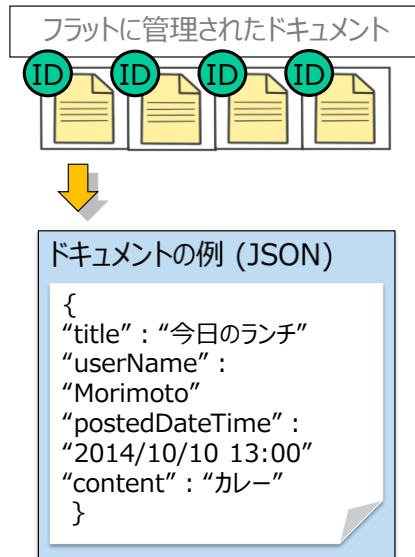
### ・説明の流れ

NoSQLには様々な種類があり、データの格納方式などで、大きく分けて4種類のNoSQLデータベースに分類されます。

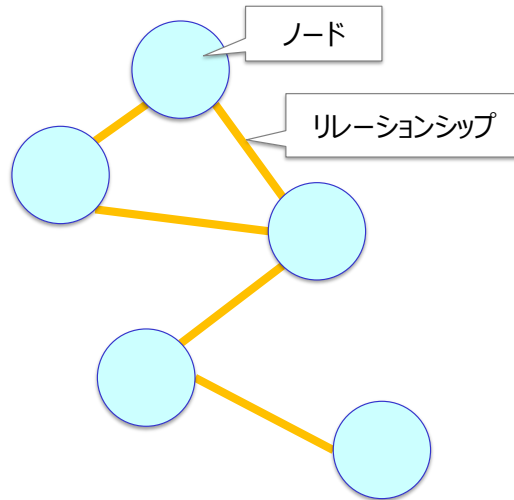
- ・ポイント (絶対に覚えてほしいこと、など)  
4つの種類のNoSQLを押さえる。

## NoSQLの代表的な種類

### ドキュメント型



### グラフ型



- ・ 説明の流れ  
4種類のNoSQLが存在します。

## キー・バリュー型の特徴

### キー・バリュー型

### キー・バリュー型



キーとバリュー(値)のセットでデータを整列します。  
キーは、値に対する識別子となるデータです。  
値はバイナリデータであれば、BLOB型の画像や音声も格納できます。

キーとキーに紐付けられたデータという単純な構造であるため、スキーマ定義が不要で、検索が高速化でき、またサーバの追加やデータの分割・配置作業が容易となります。つまり、簡単にスケールアウト環境が構築できます。

Dynamo、Voldemort、Riak、Hibari、Redis、Scalaris、TokyoCabinet/Tylantなどで採用されています。

### ・説明の流れ

キー・バリュー型のNoSQLDBの特徴を押さえましょう。

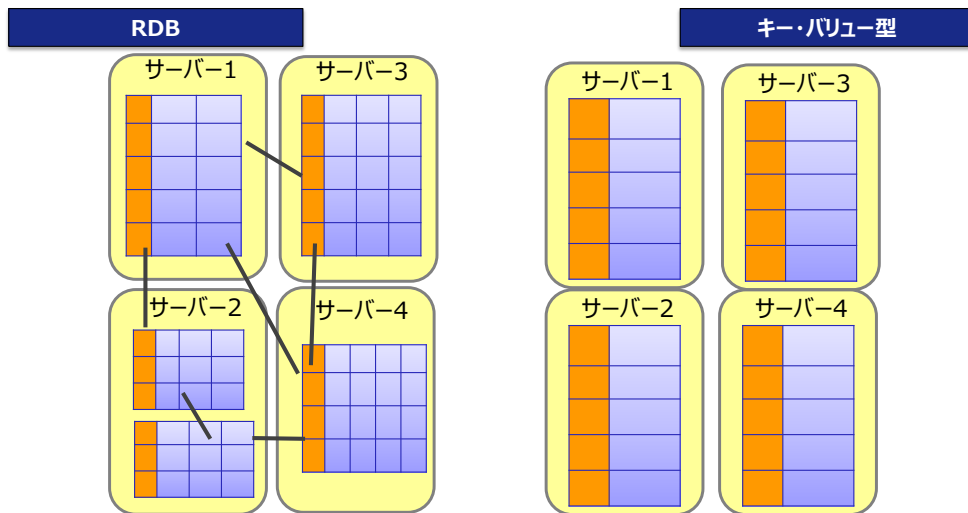
画像や音声も格納できること、スキーマ定義が不要なこと、スケールアウトが容易なことが特徴として挙げられます。



## キー・バリュー型の特徴

### RDBとキーバリュー型NoSQLのスケールアウト比較

キーバリュー型では、整合性を確認する必要がないので、キーの管理さえしていればサーバを増やすだけで問題はありません。



### ・説明の流れ

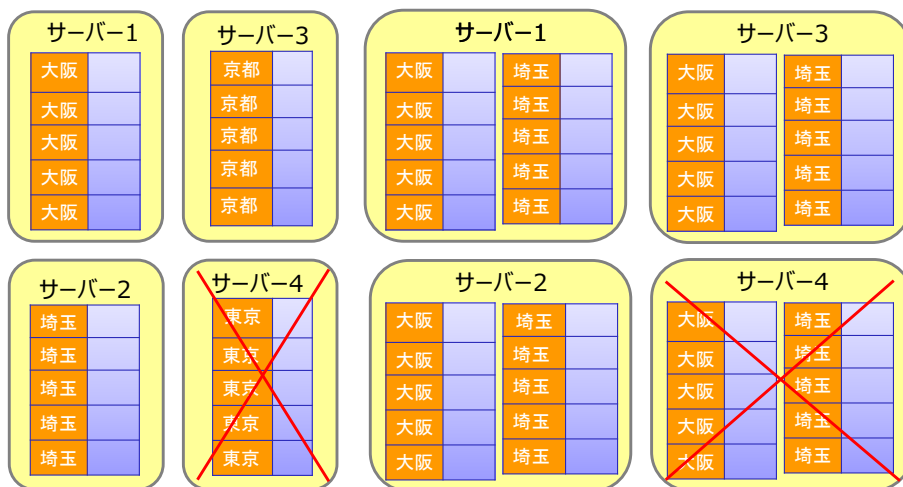
キーバリュー型NoSQLはキーと値という単純構造のため、ただサーバを増やして続きを格納していきます。

一方でRDBなどの場合は、マスターの配置も重要なため、あらかじめ構成を決めておく必要があります。

## キー・バリュー型の特徴

### シャーディング / Sharding

あらかじめサーバ毎に保存するデータのキーの範囲を設定しておくことで、検索や管理を容易にする機能です。



### ・説明の流れ

キーバリュー型では、あらかじめどのサーバに特定の範囲のデータを入れておくかを定める機能があります。

これをシャーディングといい、検索をする際には特定のサーバを探索すればよい形になるため、処理時間を短縮することができます。

ただしサーバダウンをした場合、ダウンしたサーバに格納されているデータは参照できなくなるため、サーバ相互に冗長化してデータを持たせて可用性を担保する方式がとられます。

## 列指向型の特徴

キー・バリュー型を機能強化したもので、各行のキーが複数の列(カラム)の値を持つことが可能です。

列単位でデータの保存や読み込み処理を行うことができ、列の集計処理などを高速に実行することができます。



一部の列のデータが欠けている行があっても問題ありません。また、行や列の数は自由に好きなだけ拡張可能です。

Bigtable、Cassandra、Hbase、Hypertableが採用

※カラムナデータベースとは別物です。

### ・説明の流れ

列指向型NoSQLの特徴を見ましょう。列指向型では、キーと複数のカラムを持つことができます。

複数の列項目は、項目数が一致している必要がなく、後から列の削除や追加を自由に行う事が可能です

## 列指向型の特徴

### Twitterの例

行キー：ユーザーネーム

カラムの名前：ユーザーID

カラムファミリー = カラムをまとめる入れ物

ユーザーネーム・カラムファミリー

行キー      カラム

ユーザーネーム      ユーザーID

Tanaka	A1234
--------	-------

ユーザータイムライン・カラムファミリー

行キー    カラム1    カラム2    カラム3    カラム4    カラム5

ユーザーID    a1234    a1235    a1236    a1237

a1238

A1234	t1234	t1238	t1276	t1287	t1299
-------	-------	-------	-------	-------	-------

横に増える

ユーザー・カラムファミリー

行キー      カラム1      カラム2

ユーザーID    ユーザーネーム    パスワード

A1234	Tanaka	sder5k
-------	--------	--------

ツイート・カラムファミリー

行キー    カラム1    カラム2    カラム3    カラム4    カラム5

ツイートID    ユーザーID    ユーザーネーム    ボディ    タイムスタンプ

t1234	A1234	Tanaka	やあ	a1234
-------	-------	--------	----	-------

縦に増える

### ・説明の流れ

Twitterの例でどのように柔軟に列指向型NoSQLが働くかを見てみましょう。

列指向型では用途に応じてカラムファミリーと呼ばれる入れものを定義できます。

ツイート・カラムファミリーでは、行を増やすイメージでツイート情報を管理します。

一方でタイムラインなどは、一人のユーザーのタイムラインで話された内容は横にカラムを増やして対応することができます。

さらに、列指向型は横にも縦にも情報を増やすことが可能です。

## ドキュメント型の特徴

フラットに管理されたドキュメント



ドキュメントの例 (JSON)

```
{  
  "title": "今日のランチ"  
  "userName": "Morimoto"  
  "postedDateTime":  
    "2014/10/10 13:00"  
  "content": "カレー"  
  "count": 2  
}
```

ドキュメント指向型

JSONやXMLなどのデータ記述書式のドキュメントを格納します。

階層構造を持たないフラットなドキュメントの形式でデータを管理します。各ドキュメントには管理のためのユニークIDが割り振られます。

スキーマレスです。(データ設計が不要です。)

ドキュメントの内部構造はDB、アプリケーションからわかり、任意の内容でクエリを作成できます。

MongoDB、CouchDBで採用されています。

### ・説明の流れ

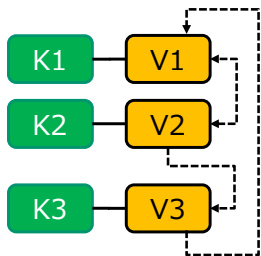
ドキュメント型の特徴は、JSONや、XMLといった準構造化データをドキュメントとしてフラットに格納することができます。

ドキュメント内の特定の情報も検索することが可能です。

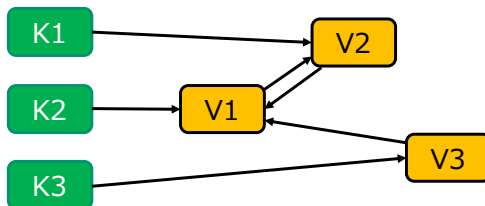
## グラフ型の特徴

### グラフ型

- キーバリュ型格納の例



- グラフ型格納の例



データ同士の入り組んだ関係を独立して表現します。  
グラフ型は以下の要素により構成されます。

ノード (node) … 他のノードと関係をもつ要素を表す。

リレーションシップ (Relationship) … ノード間における関係の有無と方向を矢印で示す。

プロパティ (Property) … ノードとリレーションシップの具体的な属性を示す。

- ・ 説明の流れ

グラフ型はデータ間の複雑な関係を表現しながらデータを格納することができます。

## 代表的なNoSQL製品

### 代表的なNoSQL製品

NoSQL製品はさまざまなものが多数存在します。

NoSQL分類	製品	ベンダー
キー・バリュー型	Amazon DynamoDB	Amazon
	Hibari	クラウドファン
	Riak	Basho Technologies
	memcached	(オープンソース)
列指向型	HBase	(オープンソース)
	Apache Cassandra	(オープンソース)
	Bigtable	Google
ドキュメント型	Apache CouchDB	(オープンソース)
	MongoDB	(オープンソース)
グラフ型	InfiniteGraph	Objectivity, Inc.
	Neo4j	オープンソース / 商用ライセンス(Neo Technology)

### ・説明の流れ

それぞれの型で代表的なNoSQL製品にはどのようなものがあるか見てみましょう。

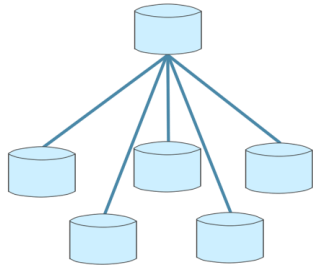
### ・ポイント（絶対に覚えてほしいこと、など）

それぞれの製品に特長があるため、試験的に利用しながら用途に応じて最適なものを選ぶ必要があります。

## NoSQLの基本的概念と技術

NoSQLのデータベースアーキテクチャ  
サーバの配置方法には、マスタ型やP2P型の2つがあります。

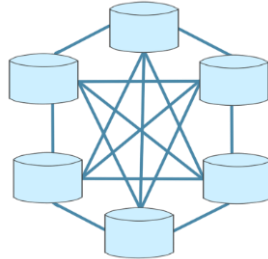
マスタ型



マスターノードはノード管理を役割を持ちます。  
単一障害点(SPOF)が存在します。  
※マスターノードがSPOFです

例 BigTable、CouchDB、  
Hbase、Hibari、MongoDB

ピア・ツウ・ピア(P2P)型



単一障害点はありません

例 Cassandra、Dynamo  
Riak、Voldemort

### ・説明の流れ

NoSQLのサーバー配置には2通りの種類があります。マスタ型とP2P型です。

マスター型は、処理サーバーを管理するマスターノードが存在します。

マスターノードはダウンした場合に単一障害点になってしまい、システム全体が利用不可能になってしまいます。

一方P2P型は障害が発生しても、サーバーが自動的に連携しながら応答する形をとるため、単一障害点は存在しません。



## NoSQL時代の必要要件

### 必要要件

NoSQLに求められることは様々であり、全てをみたすデータベースは存在しません。

- ・ 膨大なデータ : 複数のサーバを必要とするデータ量を扱う
- ・ 分散配置 : データを複数のサーバに分けて保存する
- ・ コスト : 性能は求めず、安価なハードウェアを多数用意する
- ・ 信頼性 : データの欠損や漏洩を防ぐ
- ・ 可用性 : 稼働率を高く維持する
- ・ 耐障害性 : 障害による影響を最小限にとどめる
- ・ 性能 : 処理や応答をできる限り早く行う

### ・ 説明の流れ

NoSQLで要求される必要要件を押さえましょう。  
これらすべてを満たすデータベースは存在しません。  
必ず何かをトレードオフする必要が発生します。

## NoSQLの基本的概念と技術（整合性）

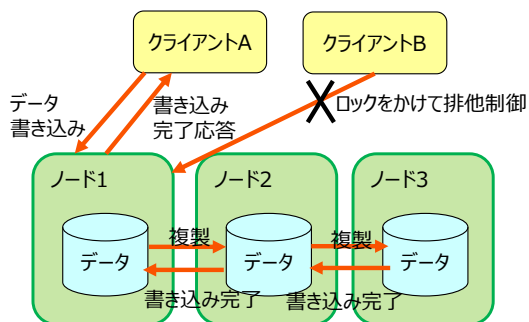
データの整合性（Consistency）とは

作業の実行後に複製(レプリケーション)されたデータ間に矛盾がないことを示します。

整合性には、「強い整合性」と「緩い整合性」があります。

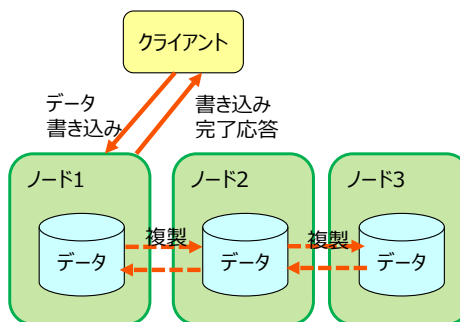
### 強い整合性

必ず最新の更新データが出力されることを保証します。特に、RDBでは重要視され、保持されます。



### 緩い整合性

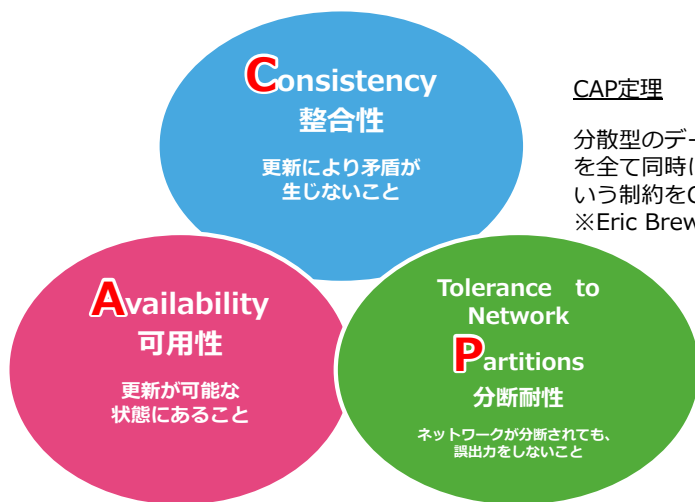
更新から少し時間をおいた後であれば、全てのノードで最新のデータが読み出せればよいとする考え方です。



書き込み完了確認がない状態でクライアントに完了通知を行う

## NoSQLの基本的概念と技術（整合性）

データベースの特性でも重要な以下の3つの要件を考えます。



### CAP定理

分散型のデータベースでは、3つの要件を全て同時に満たすことはできない、という制約をCAP定理と呼びます。  
※Eric Brewer氏により提唱されました。

### ・説明の流れ

データベース選定の上で、整合性、可用性、分断耐性は重要な選定ポイントです。

しかしながらそのすべてを完全に満たすデータベースは存在せず、3つの中の2つを満たすことが限界です。

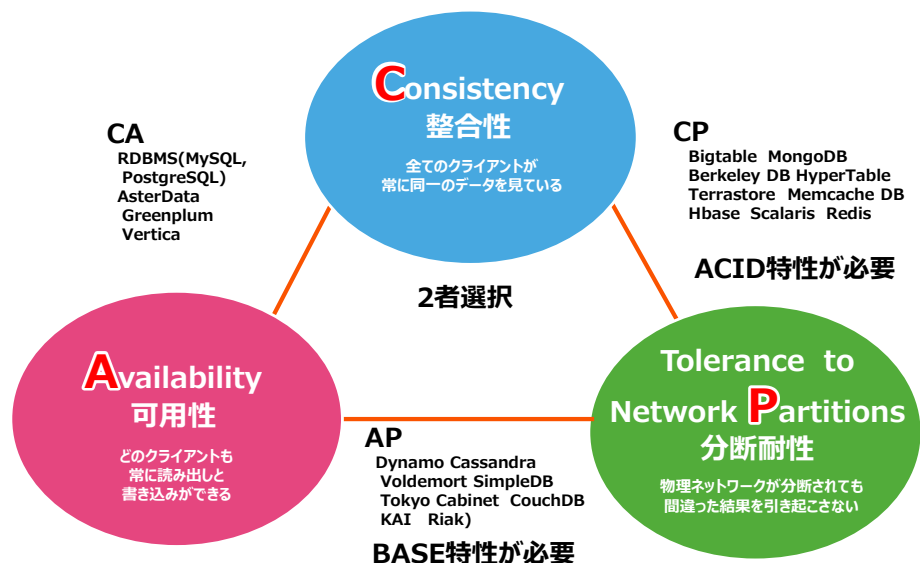
これをそれぞれの頭文字をとって、CAP定理と呼びます。

### ・ポイント（絶対に覚えてほしいこと、など）

CAP定理の考え方とDB選定

## NoSQLの基本的概念と技術（整合性）

NoSQLではCPかAPのいずれかが保証されることが基本です。



### ・説明の流れ

現存するデータベースで、CAP定理のいずれかを重視しているか見てみましょう。

RDBはやはり、CA、整合性と可用性を重視しています。

一方でNoSQLデータベースではCP、APを重視しているケースが多く、瞬間的な整合性を求めずに、とにかく動くことを重視しているケースが多いようです

## NoSQLの基本的概念と技術（整合性）

### ACIDとBASE

- ACID(Atomicity、Consistency、Isolation、Durability)

トランザクションは、全ての処理完了または処理以前の状態への復元のいずれかを行う仕組み。

Atomicity 原子性	: 手順が全て実行されるか、一つも実行されないか
Consistency 一貫性	: トランザクションの前後で整合性が保たれる
Isolation 独立性	: 実行中に他の処理に影響を与えない
Durability 耐久性	: データが正しく記録され、損失の可能性がない

- BASE(Basically Available、Soft-state、Eventual Consistency)

アプリは基本的に常に動作し（Basically Available）、強い整合性は持っていないが（Soft-state）、最終的に整合性をもつ（Eventual Consistency）という特性を備えているべきであるという考え方。

- 説明の流れ

強い整合性か、ネットワーク分断があってもとにかく動くことのどちらを重視するかが大きなポイントとなりますが、それぞれを表す言葉として、ACIDとBASEという言葉があります。

それぞれの中身を見ていきましょう。

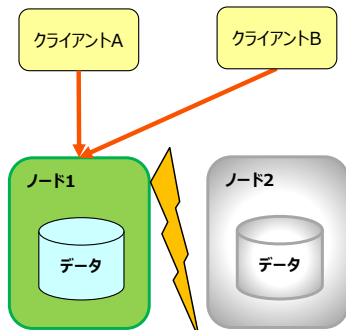
- ポイント（絶対に覚えてほしいこと、など）  
ACIDとBASEの違い、それぞれの特徴

## NoSQLの基本的概念と技術（整合性）

### ネットワーク障害時のCPとAPの対応

#### CP（整合性と分断耐性）

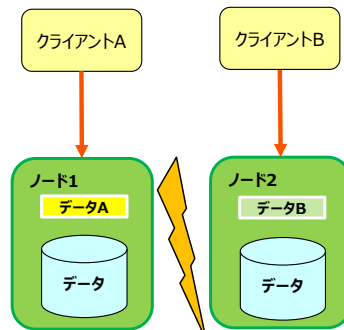
リクエストは片方のグループだけが受け付け、他のグループは自動的に停止します。データを受け付けないグループは更新が起きないため、整合性が乱れる心配はありません。



ネットワーク障害による分断が発生

#### AP（可用性と分断耐性）

全てのノードで更新を行えるようにするため、リクエストは全てのグループが受け付けます。情報の共有ができないノード間で不整合が発生します。



ネットワーク障害による分断が発生

#### ・説明の流れ

CPとAPでネットワーク分断が起こった際の挙動の違いを見てみましょう。

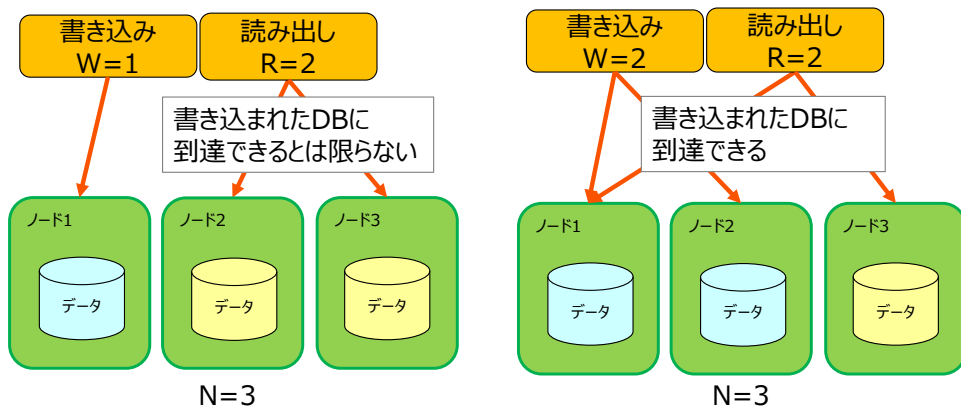
APではネットワーク分断により、ユーザー間で一時的に結果が異なっても結果を返し続ける方法を取ります。

## NoSQLの基本的概念と技術（整合性）

### 整合性と性能のトレードオフ

R(ノードから読み込む数)とW(ノードに書き込む数)の和がN(レプリカの数)より大きければ、整合性が保証することができます。

複数の書き込みと読み出しを行う事は、応答速度とのトレードオフになります。



### ・説明の流れ

APの場合、整合性をあきらめてしまうしかないかということではない場合もあります。

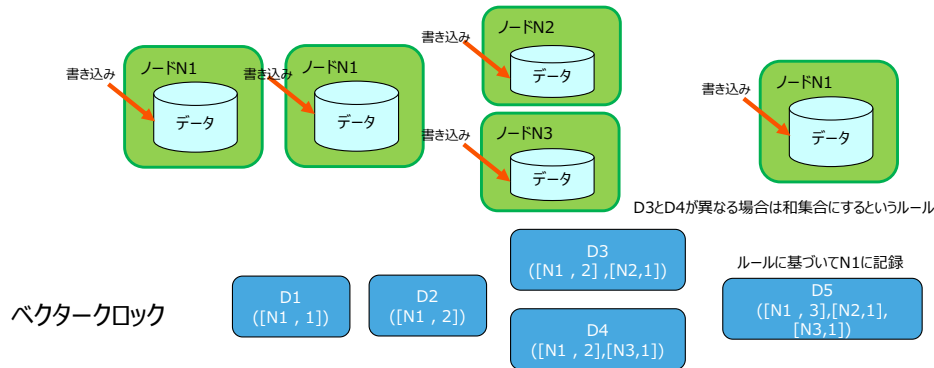
書き込みと読み出しを行う際に、複数のノードに処理を行う事で、正しい結果に到達できる仕組み（Quorum）を取ることができます。

ただし、この場合、書き込み、読み出しを複数行う必要があるため、応答性能は低下します。（トレードオフになります）

# NoSQLの基本的概念と技術（整合性）

## データのバージョン管理

- ・タイムスタンプ(TimeStamp)  
データの更新時間を記録する。  
新しいタイムスタンプを持つものを重要視することでバージョンを管理する。
- ・ベクタークロック(Vector Cloks)  
複数のバージョン間の仲裁をする仕組み。  
変更が行われたノードと、そのノードで変更が行われた回数を記録する。



## ・説明の流れ

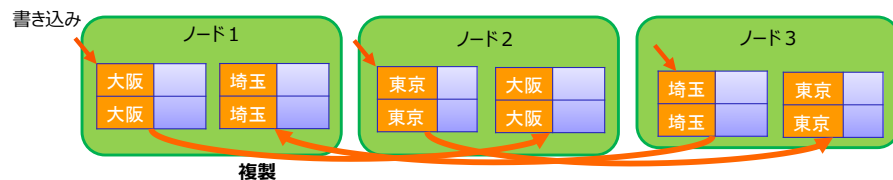
複数ノードで書き込まれたデータが最新化どうかを調べる単純な方法としては2つあり、更新時の更新時間（タイムスタンプ）を見る方法と複数バージョンの記録をとり、仲裁するベクターロックとい仕組みも用意されている方法があります。



## NoSQLの基本的概念と技術（データ分割）

### 整合性を後から修復する仕組み

- ・リードリペア（Read Repair）  
全てのノードで複製データの読み出し要求を行い、データに矛盾がないかを確認する。発見された矛盾はバックグラウンド処理で修復する。
- ・ヒントド・ハンドオフ（Hinted Handoff）  
故障が発生した場合、他の稼働中のノードに、故障の復旧後の更新手続きを指示する仕組み。
- ・マークル・ツリー（Markle Tree）  
Markle Treeという樹形図構造のアルゴリズムによりデータの整合性を確認し、整合性に問題がある場合は手動での修復を行うもの。



### ・説明の流れ

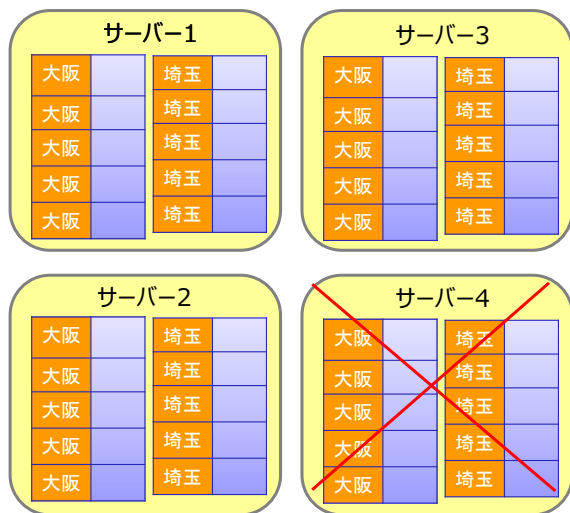
データ分割におけるNoSQLの機能を見ていきましょう。

NoSQLではデータに不整合が発生したり、ノードが故障していたりする場合には、緩やかにバックグラウンドでデータの補修を行う事が機能（リードペア）が搭載されている場合があります。

## NoSQLの基本的概念と技術（データ分割）

### シャーディング（Sharding）

あらかじめ、サーバ毎に保存するデータのキーの範囲を設定しておくことで、検索や管理を容易にする機能です。



他、アルファベットA~G、H~N、O~Z  
ID 1 ~ 1000、1001 ~ 2000...など

データの偏りがあった場合、適切な負荷分散ができません。なので、データ単位を小さくし、ノード数より多くすることなどにより、データの偏りを減らす工夫が必要です。

BigtableやHbaseは再分割、再配置を自動的に行います。

### ・説明の流れ

すでにご紹介したシャーディングもデータ分割の一手法です。

データの中身によって、偏りが生じる場合もあるため、最初から均等になる設計や再配置、再分割機能があるNoSQLを選定することも重要です。

# NoSQLの基本的概念と技術 (データ分割)

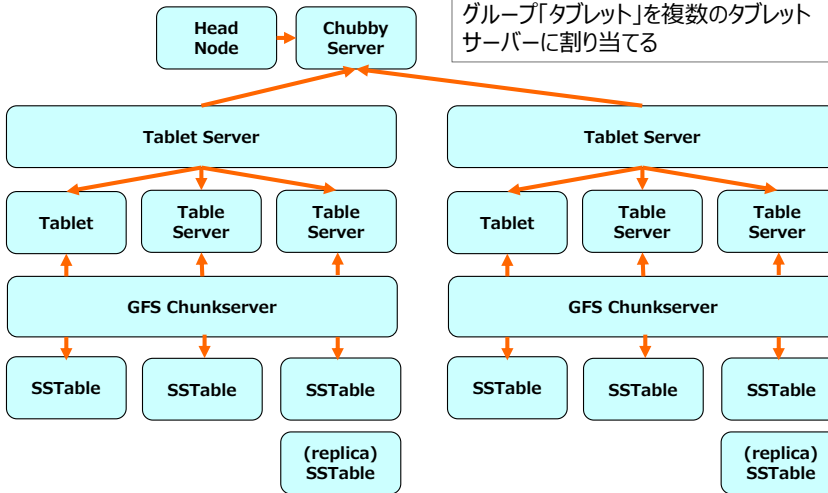
## Bigtableのデータ割り当て

辞書配列において連続するキーのグループ「タブレット」を複数のタブレットサーバーに割り当てる

論理領域容量が大きくなると自動分割

ガーベージコレクションとコンパクション

物理領域



### ・説明の流れ

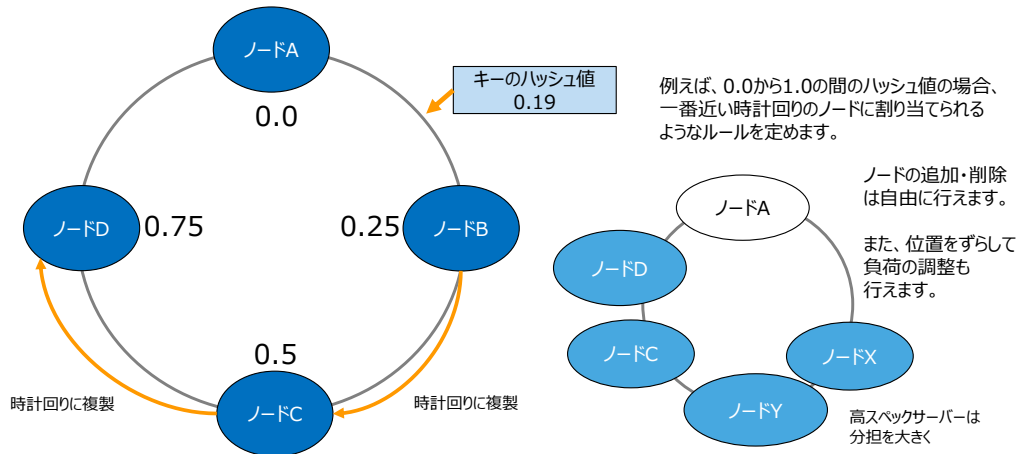
ビッグデータのデータ割り当てとして、Bigtableの仕組みを見てみましょう。

キー文字列の順番が連続するものをタブレットとしてグループ化する機能と不要となったデータを管理する機能が充実しています。

## NoSQLの基本的概念と技術（データ分割）

### コンシステントハッシング Consistent Hashing

分散型システムを構成するノードを、論理的に輪を形作るよう配置します。データのキーのハッシュ値をハッシュ関数によって求め、不規則な値であるハッシュ値を元に、データを保存するノードを選択します。ノードの位置を変更することで、負荷の代償を変更することも可能です。



### ・説明の流れ

データの分散をデータの内容に依存すると偏りが生じるため、データ内容には依存せず、特別に付与したハッシュコードによって均等に分散させる手法があります。

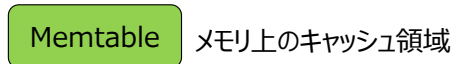
これをコンシステントハッシングと呼びます。

# NoSQLの基本的概念と技術

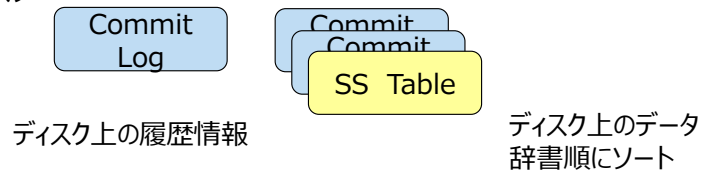
## ストレージレイアウトの技術

- ・ LSM-Tree (Log Structured Merge Tree)  
メモリを使って応答性能を高めつつ、ハードディスクにデータを格納することでデータの安全性を保つ機能。

サーバーメモリ



サーバーディスク



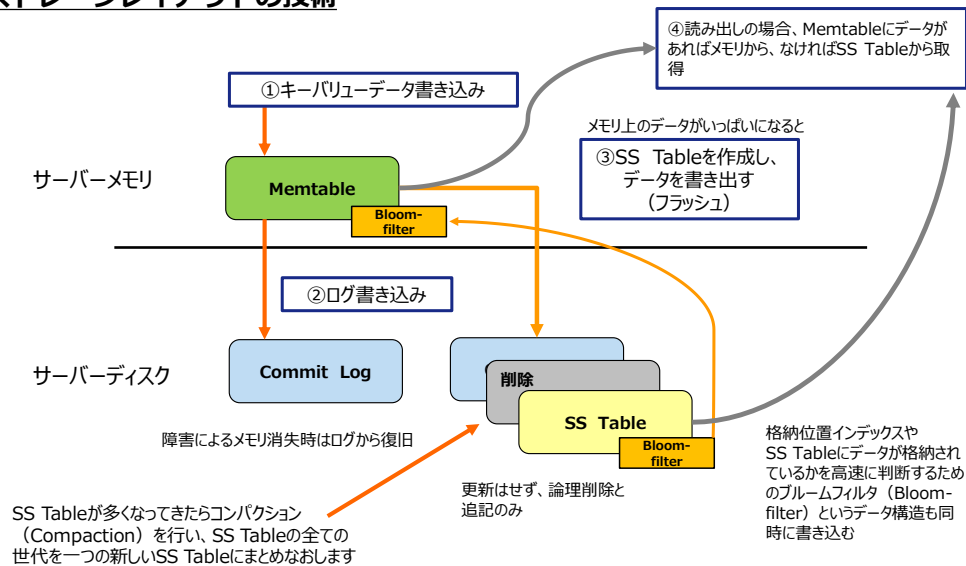
### ・説明の流れ

NoSQLのストレージはどのような仕組みになっているのでしょうか。

LSM-Treeという技術では、メモリを使って応答性能を高めながら、データをSS Tableにというメモリ上の入れ物に蓄積し、一定量たまった段階でディスクに記録する手法を取っています。

# NoSQLの基本的概念と技術

## ストレージレイアウトの技術



### ・説明の流れ

LSM-Treeをもう少し詳しくみましょう。

メモリ上のデータ操作時にCommitログも同時にディスクに記録し、障害時の復旧に利用します。

必要なSS Tableのデータはメモリにあればメモリから、なければディスクから読み込み、SS Tableの所在を記録したBloom-filterも同時に記録することで高速な応答性能を実現します。

## 第4章

# ビッグデータコア技術

## 分散処理

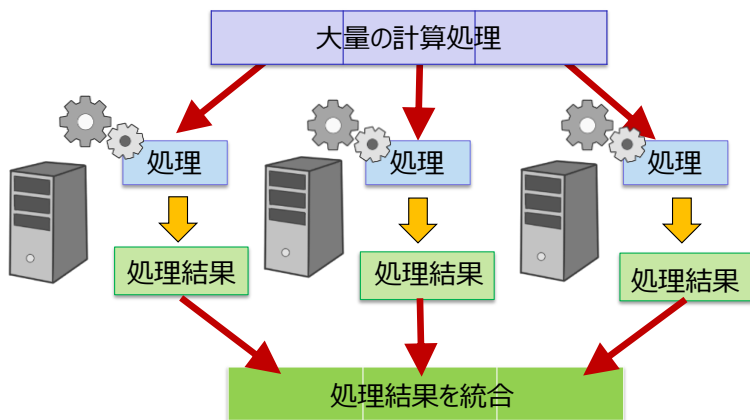
ビッグデータの基盤システムを学ぶ

基本的なビッグデータの基盤システムを理解する

## 分散処理とは

### 分散処理とは

ビッグデータを分割し、複数のサーバで同時に並行して処理した後、処理結果を統合することで全体の処理結果を生成する処理方法です。



### ・説明の流れ

ビッグデータの蓄積技術として、分散処理という技術があります。

分散処理とはどのようなものでしょうか。複数のサーバで処理を分担するという仕組みになります。

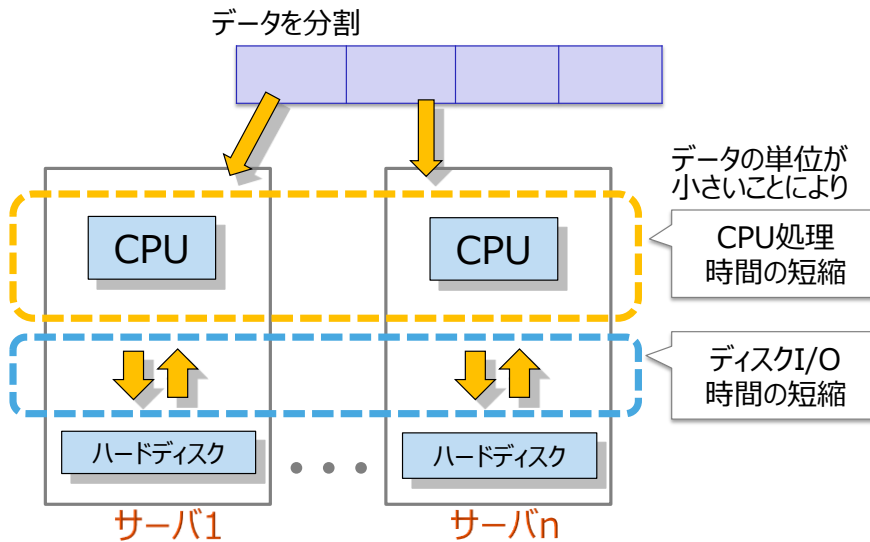
CPUやメモリ、ディスクを複数サーバで占有して利用できるため、個々の処理は高速になります。



## 分散処理とは

### 分散処理による処理時間の短縮

サーバでは、分割されたデータを処理するので、CPUでの処理時間や、ディスクからの読み込み、ディスクへの書き込みの時間が短縮されます。



### ・説明の流れ

分散処理を図示するとこのようになります。1台で処理するデータ量が少なくなるため一台当たりのCPU処理時間や、ディスクI/Oは短縮されます。

## 分散処理のメリット・デメリット

### 分散処理のメリット

ビッグデータの処理時間を短縮することができます。  
スケールアウトによる拡張が可能で、比較的 low コストになります。  
一台の故障が全体の停止には直結しないので、稼働率が高いです。

### 分散処理のデメリット

データが複数コンピュータにまたがるので、厳密な整合性が必要なオンライントランザクション処理に向きません。  
少量のデータ処理には効果が薄いです。  
システム全体の性能は、ネットワークの帯域幅や稼働率から大きな影響を受けます。

### ・説明の流れ

分散処理のメリットとデメリットを考えましょう。

大量データを扱うことができる、スケールアウトしやすいなどのメリットがある一方で、ノード間の整合性の担保や、ネットワーク通信量の増加、結合処理の負荷などが発生することは念頭に置く必要があります。

### ・ポイント（絶対に覚えてほしいこと、など）

分散処理も万能ではなく、トレードオフがあることを理解する。

## Hadoopとは



### Hadoop

Hadoopはデータベースではなく、複数のソフトウェアからなる分散処理フレームワークです。

一定期間のデータを一括処理する、バッチ向けのフレームワークです。

### 2つの主要プロダクト

- ・ファイルシステム(HDFS) : Hadoop Distributed File System  
バッチ処理向けに高いスループットを分提供する分散ファイルシステム。
- ・分散処理アルゴリズム(MapReduce):Hadoop MapReduce  
複数のサーバでの並列分散処理するプログラミングのためのソフトウェア。  
HDFS上のファイルを読み込み、集計結果を別のHDFS上に書き込む。

### ・説明の流れ

分散処理技術の中核となるHadoopというオープンソースの製品を紹介します。

Hadoopは分散処理を実現するためのフレームワークです。

2つの主要プロダクトを中心として様々な関連ソフトが存在します。

# Hadoopとは

## Hadoopの特徴

オープンソースソフトウェアフレームワークで、非構造化であるビッグデータの解析や大量のバッチ処理に用いられます。Googleによる開発の後、Apache Software Foundationが現在は開発、公開を行っています。

HadoopはLinuxで動作します。また、システムはJavaで構築されているため、JVMが必要です。

MapReduceフレームワークやHDFSを操作するためのJavaのAPI（クラスライブラリ）が提供されています。

専門スキルが導入に必要なため、商用のサポートサービスを提供している会社があります。（クラウドラ、ホートンワークス、MapRなど）

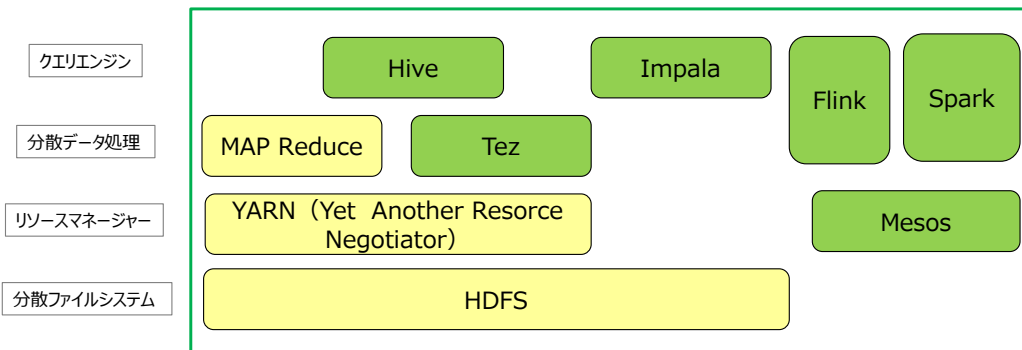
- 説明の流れ  
Hadoopの特徴を押さえましょう。

## Hadoopの基本構成

Hadoopの基本コンポーネントは、

- ・分散ファイルシステムのHDFS
- ・リソースマネージャーのYARN
- ・分散データ処理を行うMapReduce

です。その他のシステムは、Hadoopから独立して開発されたものです。



- ・説明の流れ

Hadoopを構成する製品群を確認しましょう

。

リソース管理のYARNや、高速集計の、Impala、Tez、SQLを扱えるHiveなどもあります

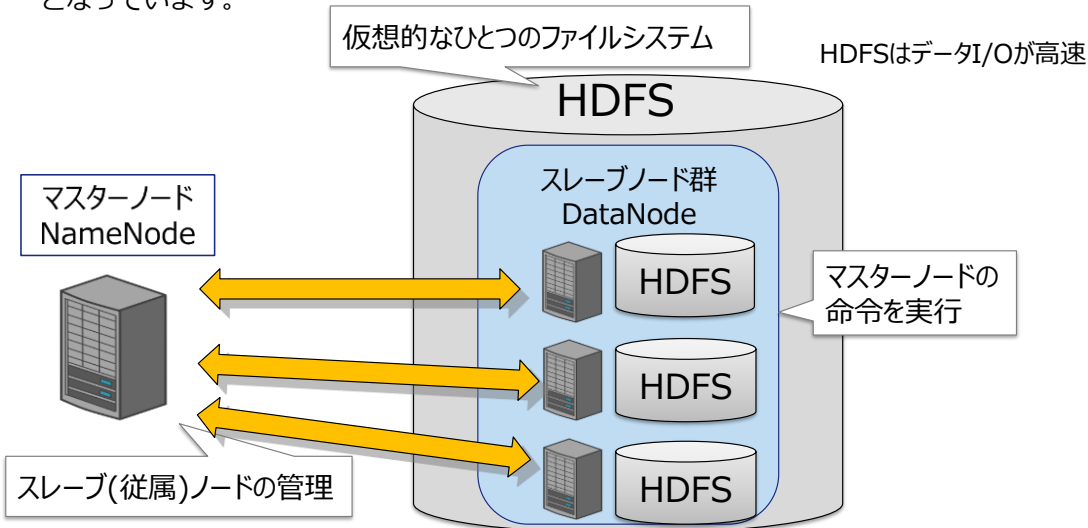
。

機械学習のライブラリが充実したSparkもよくHadoopと共に利用されます。

# HDFS (Hadoop Distributed File System)

## HDFS

HDFSはマスタ型で、データを管理するスレーブノード群と、スレーブノードを管理するマスターノードから構成されています。全体で、仮想的な一つのファイルシステムとなっています。



## ・説明の流れ

HDFSの仕組みを理解しましょう。

HDFSはデータの位置を管理するマスターノードとデータを格納するスレーブノードで構成されています。

利用する側からは、複数の分散データが一つの巨大ファイルシステムとして見えます。

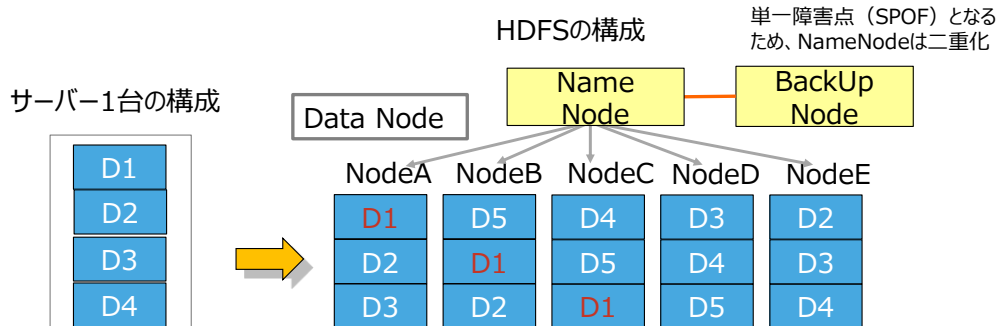
## HDFSはマスタ型

### マスタスレーブ型

NameNode、DataNodeに役割が分かれます。

NameNodeはメタ情報(他のノードのデータやブロックの対応関係など)を保持するマスターノードです。単一障害点となるため、バックアップノードを作成します。

DataNodeがデータ(ブロック)を保持します。障害対策として、同じデータを複製し、ノードを跨って保存されます。



データ処理の分散と障害対策のために、あらかじめ決めた数(例:3つなど)の複製を作成し、異なるノードに分散して保存する。

大きなサイズのファイルは任意のブロックサイズに分割して格納。(デフォルト64MB)

### ・説明の流れ

HDFSはノードが破損しても、データ格納時にデータを分割し複数サーバーに配置することで、復元を可能としています。

自動でデータ配置を催事調整し、復元します

ただし、マスターノードだけは単一障害点となるため、冗長化構成をとる必要があります。

## HDFSの読み書き

### HDFSへの書き込み・読み出し

データは、クライアント側で任意のサイズのブロックに分割され、NameNodeがどのDataNodeに格納するか指示し、書き込みを行います。  
読み出しは、NameNodeがクライアントに対して目的のデータが保存されているDataNodeを示し、直接ブロックを読み出します。

### データの複製と耐障害性

データ処理の分散と障害対策のために、一定数複製を作成し、複数のノードにまたがってデータを保存します。障害時は、障害のあるノードをクラスタから切り離し、切り離されたノードにあった複製の分を、他のノードに複製します。

### リバランシング

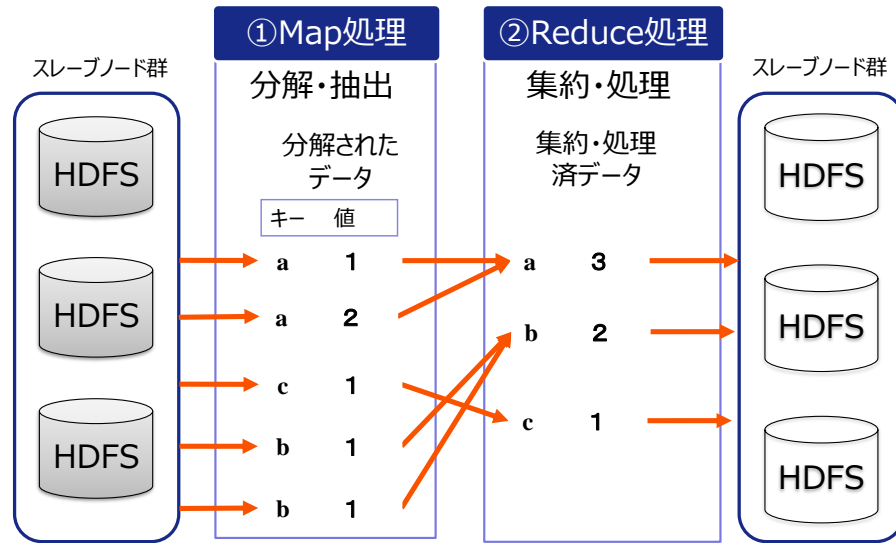
ノードの追加や削除された場合に、データの分散に偏りが生じないようにデータの再配置を行います。

### ・説明の流れ

HDFSはデータの複製と再配置を自動で行う事ができます。



# MapReduce処理とは



## ・説明の流れ

Hadoopでデータを集計する場合はどのように行うのでしょうか。

Map処理でデータをキーと値に分割、抽出し、Reduce処理で同じキーのデータを集約、集計します。

## MapReduce処理とは

### MapReduce

大量データを分割し、複数のサーバで並列処理するための方法です。複数のスレーブノードで同一の処理を実行します。次の2段階の処理に分かれます。

#### ①Map処理

データを小さく分割し、必要な情報を選んで出力します。出力データはキーバリュー型の構造となります。

#### ②Reduce処理

キーごとにデータを集約し加工を行なった上で、HDFSに出力します。

Map処理とReduce処理の開発には、Hadoopが提供するJavaのAPI(クラスライブラリ)を利用します。複雑な分散処理アルゴリズムや障害対応処理などの実装は不要で、開発者は業務ロジックの開発に集中できます。

#### ・説明の流れ

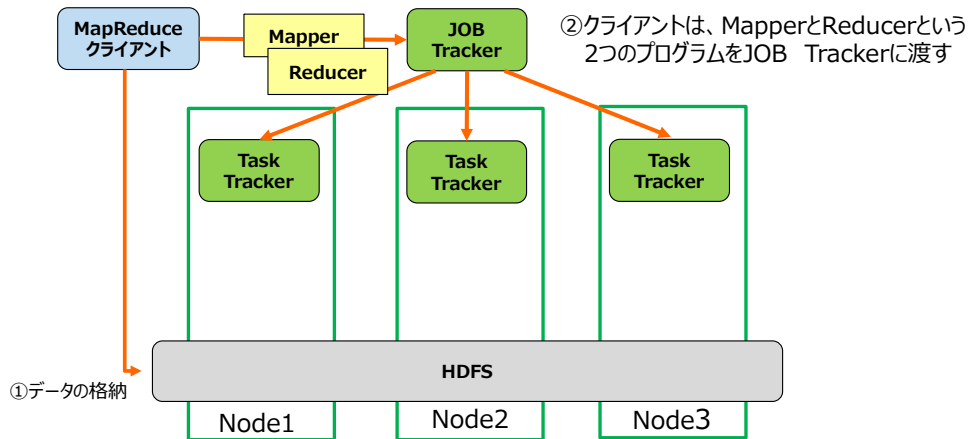
MapReduce処理は、複数サーバ上で動作します。

実際には小さなJavaプログラムが、各サーバノードで実行されます。

# MapReduceのアーキテクチャ

## MapReduceのアーキテクチャ

MapReduceのマスターはJobTrackerと呼ばれ、全てのMapReduce処理工程の調整・管理とスレーブとなるTaskTrackerの維持管理を行います。



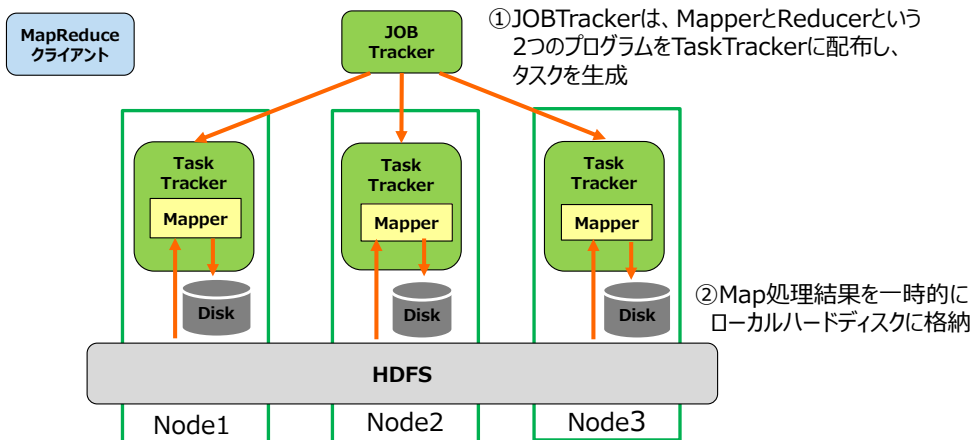
### ・説明の流れ

MapReduceの動作の流れを追ってみましょう。  
集計要求を発行したいタイミングで、MapReduceクライアントからMapperとReducerという2つのJavaプログラムをJOB Trackerに渡します。

# MapReduceのアーキテクチャ

## MapReduceのアーキテクチャ

JobTrackerはMapperとReduceプログラムをTaskTrackerに配布し、HDFSブロックと同じ数だけMapタスクを生成します。Mapタスクは、データが存在するTaskTrackerに割り当て、必要なデータ転送のみを行います。



### ・説明の流れ

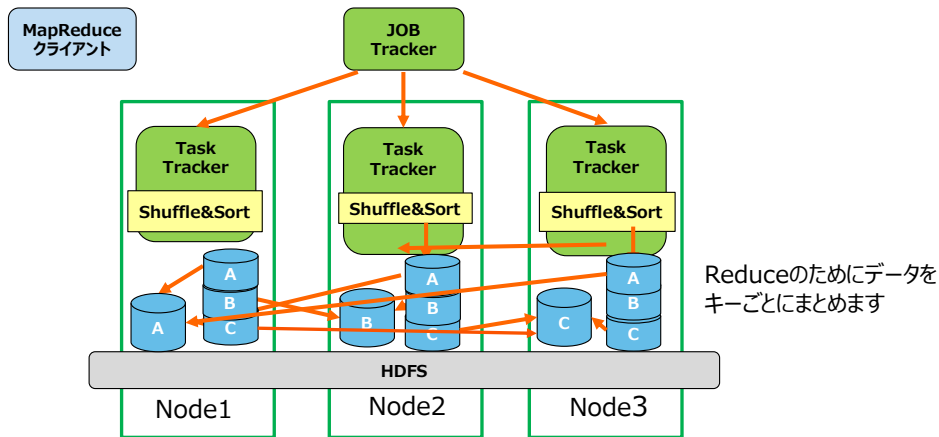
Mapperは各サーバーノードに配布され、HDFSからデータを抽出します。

抽出したデータは一時的にローカルハードディスクに格納されます。

# MapReduceのアーキテクチャ

## MapReduceのアーキテクチャ

JobTrackerは、ある程度処理が進むとShuffle&Sort処理を起動します。Mapタスクにより処理された中間データを、キーの値によって、ノードを跨っているものを含めて1箇所にグループ化します。



### ・説明の流れ

Mapで抽出したデータを各サーバーで処理すると各サーバー間で都度通信が発生してしまいます。

これを避けるため、キーごとに各ノードにデータを集め、同じキーについては、同じサーバーで処理できるように調整します。この作業をシャッフルと呼びます。

### ・ポイント（絶対に覚えてほしいこと、など）

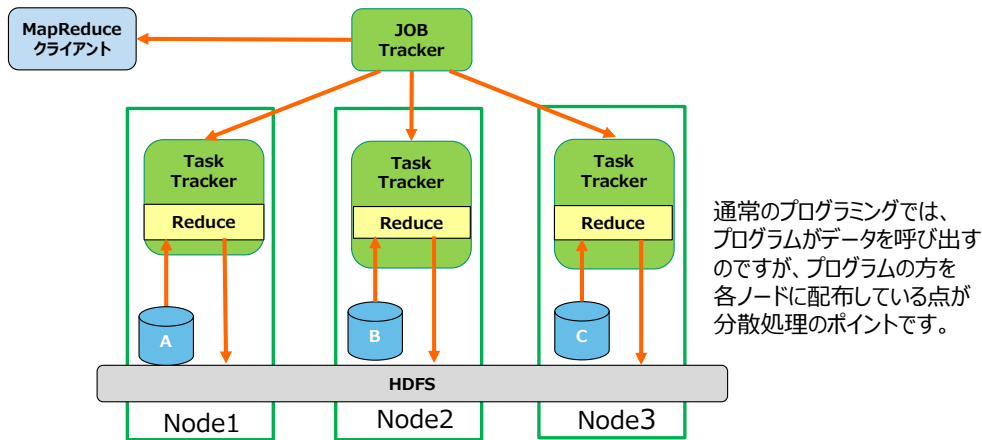
処理によってはシャッフルが不要な処理と、必ず発生するものが出てくる。

シャッフルが発生するかどうかを常に意識してプログラミングをすれば、思わぬ速度低下を回避できるだろう。

# MapReduceのアーキテクチャ

## MapReduceのアーキテクチャ

処理がさらに進むとJobTrackerはReduceタスクをTaskTrackerに割り当てます。TaskTrackerはReduceタスクを実行し、処理結果をHDFSに書き込みます。全てのタスクが終了するまで繰り返し行います。



通常のプログラミングでは、プログラムがデータ呼び出すのですが、プログラムの方を各ノードに配布している点が分散処理のポイントです。

### ・説明の流れ

次に、集計処理をReducerが行います。

一段目の集計で処理が完結しない場合は、トーナメント方式で、集約をかけ続けます。

### ・ポイント（絶対に覚えてほしいこと、など）

MapReduceの特筆すべきは、プログラムが各ノードからデータだけを収集する形ではなく、プログラムを各ノードに配布して、各ノード上で処理する点である。

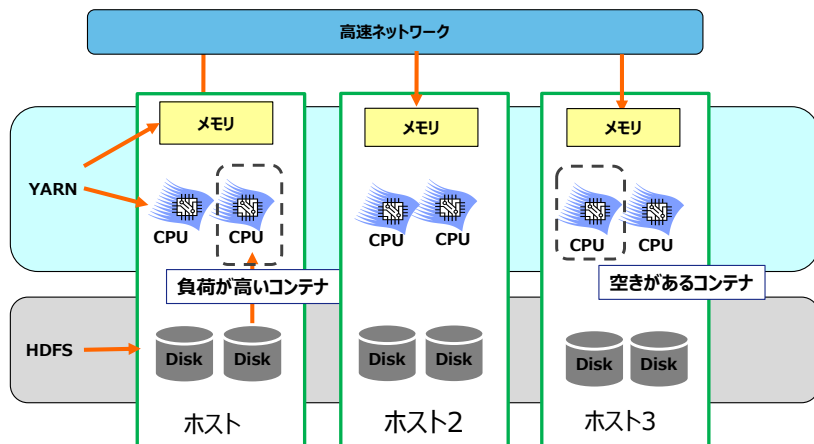
つまり、集計処理という業務ロジックまでを分散処理で行っていることになる。

## 分散ファイルシステムとリソースマネージャー

### YARN

CPUやメモリなどの計算リソースはリソースマネージャーであるYARNにより管理されます。CPUコアやメモリをコンテナと呼ばれる単位で管理します。

分散アプリケーションが実行された時に、クラスタ全体の負荷のバランスを見てコンテナが割り当てます。



### ・説明の流れ

各ノードに処理を分散させる際に、負荷がかかるノードとそうでないノードはどうしても発生してしまいます。

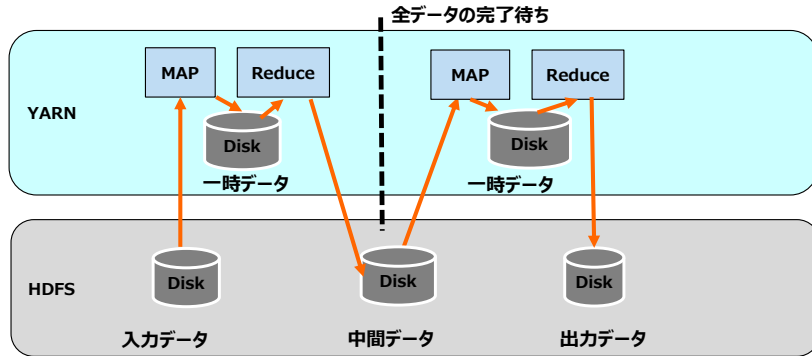
そこで、リソース管理をするYARNが各ノードの負荷状況を常に監視し、効率的に処理を割り振っていきます。

# 分散データ処理とクエリエンジン

## Hive on MapReduce

MapReduceはJavaプログラムをデータに対し実行できるため、加工には便利です。SQLに対応させる場合、専用に設計されたクエリーエンジンApache Hiveを利用します。

Hiveは応答性能が重要視される場合や少量データを扱う場面などではには不適です。



データ処理ステージが変わるタイミングで待ちが発生。  
中間データを毎回Diskに保存するため低速

### ・説明の流れ

MapReduceはJavaプログラムから呼び出しますが、ここがネックとなって使いにくいとの声もありました。

そこでSQL言語で要求を投げられるHiveという仕組みが考案されました。

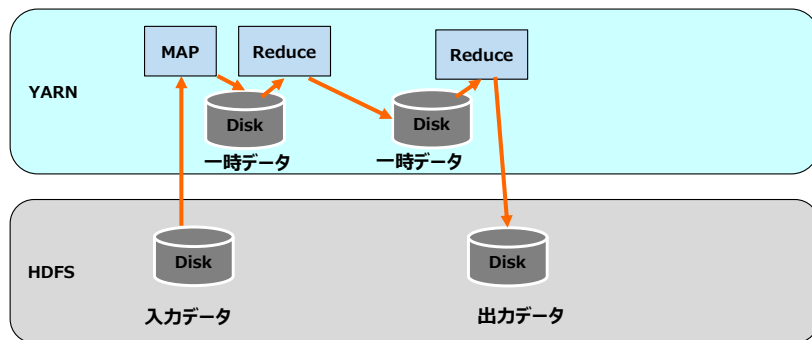


## 分散データ処理とクエリエンジン

### Hive on Tez

Apache TezはHiveの高速化するために開発されたコンポーネントです。Mapreduceの仕組みを調整し、処理段階の終わりで他のプログラムの情報を保持せずに後続の処理へと受け渡す仕組みに変更しました。

ステージの合間にあったディスクへの書き込み、待ち、Map処理を省略



現在、Hive on Sparkというものも存在します。RDDがMapreduceに代わる存在として活用されています。

### ・説明の流れ

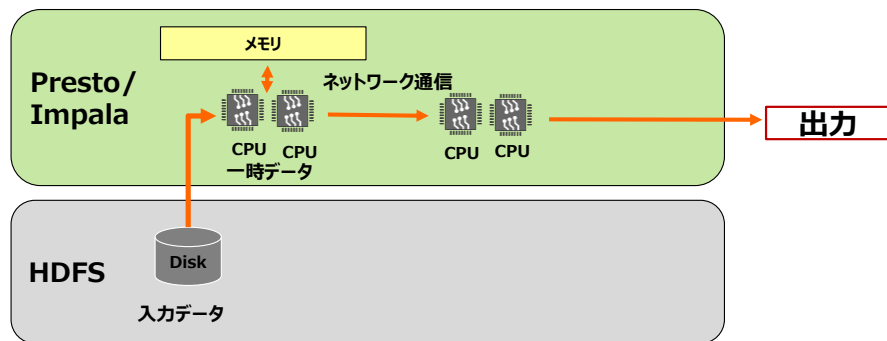
MapReduceはこれまで処理できない量のデータを処理できる点において特筆すべき能力を有していますが、処理の各段階でローカルファイルシステムにデータを書き出すため、レイテンシは下がってしまいます。

これを打破するために余分なMap処理とディスク書き込みを省略することにより、高速化を図っています。

## 分散データ処理とクエリエンジン

### Impala / Presto

ImpalaとPrestoは、Hiveの高速化ではなく、対話型クエリ実行に特化させることで、SQLに対応させた製品です。バッチ処理ではなく、最大瞬間速度を最大にするため、あらゆるリソースを有効活用するよう設計されています。



自前の分散処理機構を実装し、マルチコアを活用しながら並列化処理を実行

### ・説明の流れ

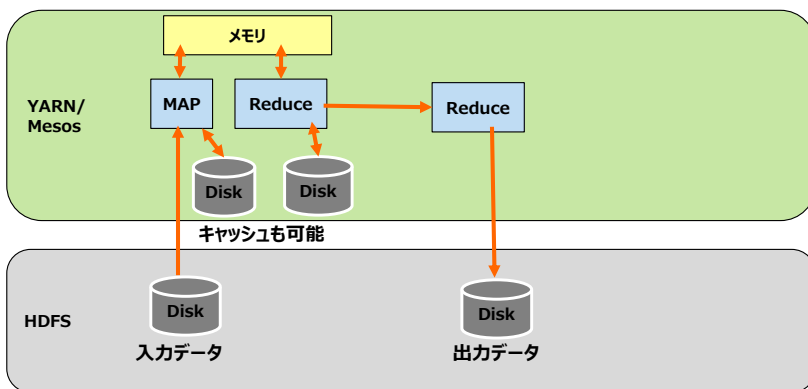
Hiveでも応答性能に对应されない場合もあったため、クエリエンジンとしての製品開発も進みました。

それらが、ImpalaとPrestoです。HDFSから抜き出したデータを一度メモリにあげ、最後までメモリ上で処理を続けます。

## 分散データ処理とSpark

### Spark

Apache SparkもMapReduceより効率良くデータ処理をするために開発されました。メモリの使用量増やすことで高速化を実現します。Sparkの実行にはJavaランタイムが必要ですが、Java、Scala、Python、Rからも呼び出すことができます。



Sparkは中間DBにデータを書き込むことなくメモリ上に保ち続けます。障害でデータが失われるともう一度最初からやり直します。

### ・説明の流れ

一方で、分散処理フレームワークに適した、データモデルも考案されました。

分散処理で扱える小さなデータ単位RDDを介して処理を行うSparkという言葉です。

SparkではMAP、Reduceという仕組みを保ちながら、処理自体をオンメモリで完結させることが可能です。

### ・ポイント（絶対に覚えてほしいこと、など）

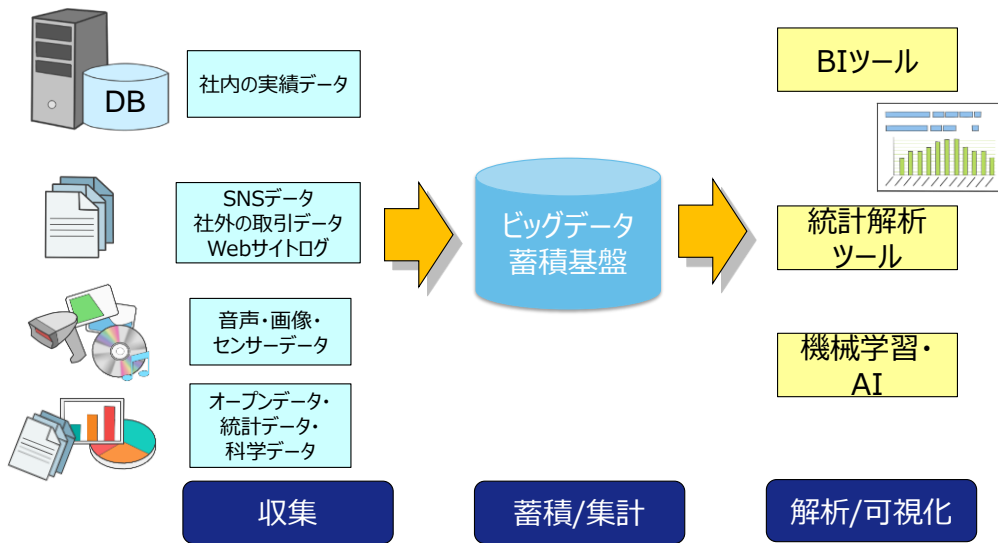
得られた結果明示的にディスクに保存する場合は、キャッシュすることも可能。

## 第5章

# ビッグデータにおける データ解析

ビッグデータで役に立つ分析手法を学ぶ

# ビッグデータにおけるデータ解析



・説明の流れ

ビッグデータにおけるデータ解析の流れを復習しましょう。

## アドホック分析ツール

データを可視化するためのソフトウェアは多種多様に存在します。

### アドホック分析ツール

行錯誤を繰り返しながら、繰り返しデータを見ていく際に利用するものです。アドホック分析環境には次のようなものがあります。

- Jupyter Notebook

Python、Ruby、Rなどのソースを書き留めながら実行することが可能。  
matplotlibライブラリを利用してグラフを作成することもできる。

- Apache Zeppelin

Spark/Hadoopなどの分散処理システムに対してコードを実行し、  
実行結果をグラフとして描画することが可能。

シェル/SQL/scala/python言語に対応している。



<https://zeppelin.apache.org/>

- 説明の流れ

アドホック（随時、自由）に集計するアドホック集計ツールなどもオープンソースで存在し、ビッグデータ処理後の可視化に利用されています。

## ダッシュボードツール

### ダッシュボードツール

傾向の把握が進むと、定期的にクエリを実行し、レポートやグラフなどを用いたダッシュボードを作成するツールです。

対話的なBI商用ツールとしてTableau、ClikViewなどが存在します。  
オープンソースのダッシュボードツールも存在します。

- Redash  
Python製のダッシュボードツール。  
SQLクエリの実行結果を直感的に可視化できる。搭載したDBに結果を記録するため、表示自体は高速だが、クエリ結果が大規模になるとエラーや遅延が生じる。
- Superset  
Python製のWebアプリケーションで対話的なダッシュボードを作成する。  
集計は外部ストレージ「Druid」で行い、リアルタイムな出力を行える。
- Kibana  
JavaScript製の対話的可視化ツール。  
Elasticsearchのフロントエンドでよく用いられる。

### ・説明の流れ

ダッシュボードツールは、分析結果を共有する対話的なBI商用ツールとして、オープンソースの他、商用製品でも存在しています。

## データ活用：一般的な統計分析手法

度数分布と  
ヒストグラム

データを範囲に区切って表したもので、データの分布を知ることができる

平均と標準偏差

状況を特定することができる

正規分布

統計的なデータを元に、可能な予測を立てることができる

標本調査  
(全体像の推測)

標本（一部のデータ）を元に、全体像を把握できる

標本平均

標本の平均を求めることで、全体の平均を推測できる

### ・説明の流れ

統計学でよく使われる手法を知ることがビッグデータ分析の助けになります。簡単に見ていきましょう。



## 度数分布とヒストグラム

### 度数分布

度数分布表を用いて表します。

※度数分布表：データをいくつかの範囲に分け、その範囲に入るデータの個数を書き表した表。

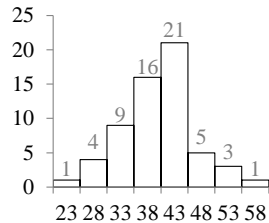
データがどのような値を中心にして、どのように広がっているのかを調べることができます。

値)	人数
21~25(23)	1
26~30(28)	4
31~35(33)	9
36~40(38)	16
41~45(43)	21
46~50(48)	5
51~55(53)	3
56~60(58)	1

### ヒストグラム

ヒストグラムとは、データの分布状態（度数分布）を表す棒グラフのことです。

度数分布の他の表現方法としては、箱ひげ図があります。



- ・説明の流れ  
度数分布とヒストグラムを紹介します。

## 平均と標準偏差

- 平均：データの代表値としてよく利用される。  
平均の代わりに、最頻値、中央値を用いることもある。

$$\text{平均値} = \frac{\text{データの値の総計}}{\text{データ数}}$$

- 分散：データがどのくらいばらついているかを表す。

$$\text{分散} = \frac{(\text{各データの値} - \text{平均})^2 \text{の総計}}{\text{データ数}} \quad \dots \text{平均との差の2乗の平均}$$

- 標準偏差：そのデータの傾向や性質を把握するために利用される。
  - 異なるデータ間でのデータのばらつきの度合いを比較できる
  - 平均値からのばらつきの幅を測定できる。

$$\text{標準偏差} = \sqrt{\frac{(\text{データの値} - \text{平均})^2 \text{の総計}}{\text{データ数}}} \quad \dots \text{分散の平方根}$$

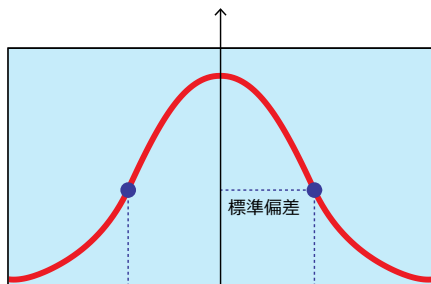
- 説明の流れ  
平均と標準偏差を紹介します。

## 正規分布

### 正規分布とは

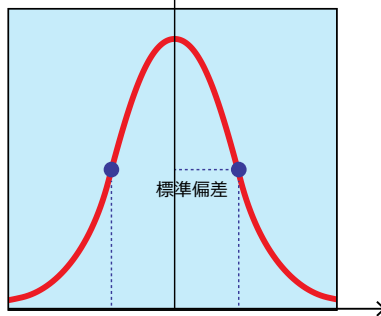
- ・ 確率分布の1つ。
- ・ 平均値（または中央値）を中心に左右対称な、釣鐘型の分布。
- ・ 平均値で最も発生確率は高くなり、平均値から離れるほど低くなる。
- ・ 正規分布をしていると想定できるデータに関しては、平均値と標準偏差さえわかれば、どのように分布が広がっているかを推定できる。

標準偏差が大きい＝ばらつきが大きい



平均値 (中心値)

標準偏差が小さい  
＝ばらつきが小さい



平均値 (中心値)

### ・ 説明の流れ

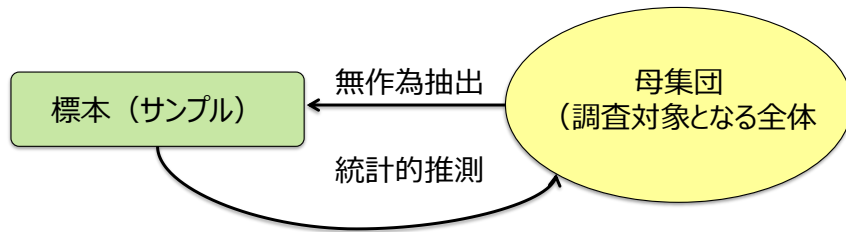
正規分布は、偏りのないバラツキと、どれくらい乖離があるかを調べる際に便利です。

例えば、適正在庫のあり方と、現状の乖離などを見たい場合に使います。

## 標本調査と標本平均

### 標本調査とは

標本、つまり大量のデータ（母集団）の一部を入手、観測した際に、その母集団を推測するために仮説検定の考え方をを用いて行う統計的推定の手法のことを言います。



### 標本平均とは

標本データの平均値を利用することで、母集団の平均を推測することができます。母集団の傾向を推測するために利用されます。

… (標本平均) = (抽出した標本データの合計) ÷ (抽出した標本データ数)

### ・説明の流れ

ビッグデータ登場以前は全数分析ができませんでした。

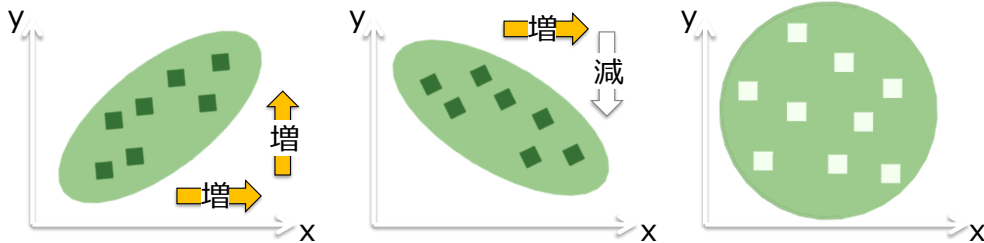
その代わりに標本を抽出して、全体を推測する手法がとられてきました。

## 相関関係

### 相関関係とは

一方の値の変化に伴って、もう一方の値が変化するという、2つの値の関係を相関関係と言います。

- ・ 正の相関：2つのデータのうち、一方の値が**増加**すると、もう一方の値も**増加**するような関係。
- ・ 負の相関：2つのデータのうち、一方の値が**増加**すると、もう一方の値は**減少**するような関係。



正の相関 (右上がり)

負の相関 (右下がり)

無相関

### ・ 説明の流れ

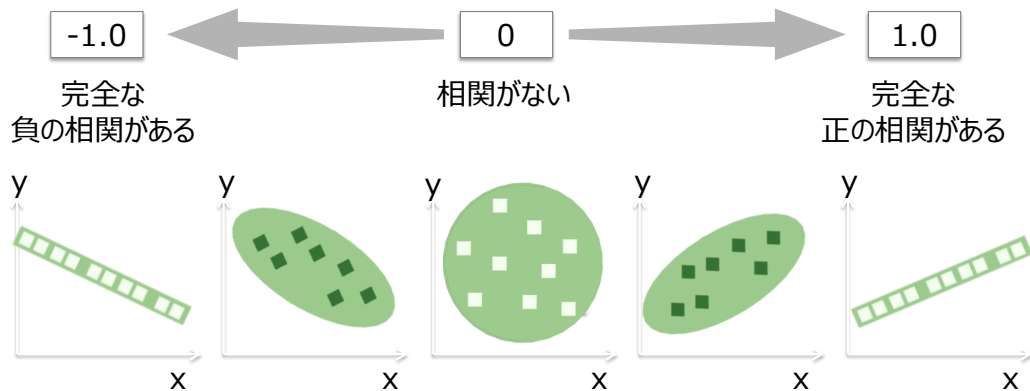
相関関係は、ある数値とある数値の関連を調べることで、一方の増減からもう一方の増減を予測することが可能になる手法です。

## 相関係数

### 相関係数とは

2つのデータ間の相関関係の強さを表す値のことで-1から1までの値を取ります。  
相関係数が0に近いほど相関は弱く、1に近いほど相関は強くなっています。

… (相関係数) = (共分散) ÷ (一方の標準偏差 × もう一方の標準偏差)



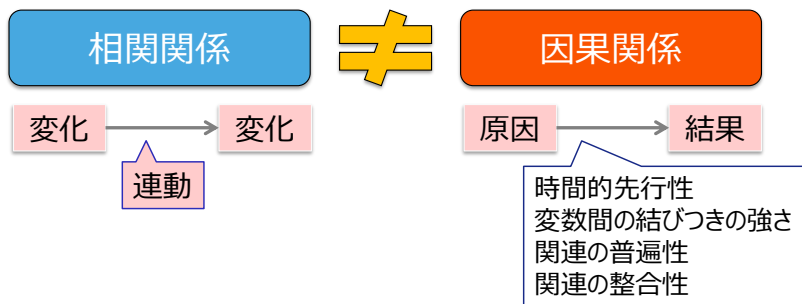
### ・説明の流れ

相関係数は相関を数値化した値で、1に近いほど正の相関があると言えます。

## 相関関係と因果関係

- ・相関関係：一方の値が変化すると、もう一方の値も変化するという2つの値の関係のこと。
- ・因果関係：2つ以上のものの間に原因と結果の関係が成り立っていること。つまり、一方によって、もう一方が引き起こされるという関係。

相関関係があっても因果関係があるとは限りません。



### ・説明の流れ

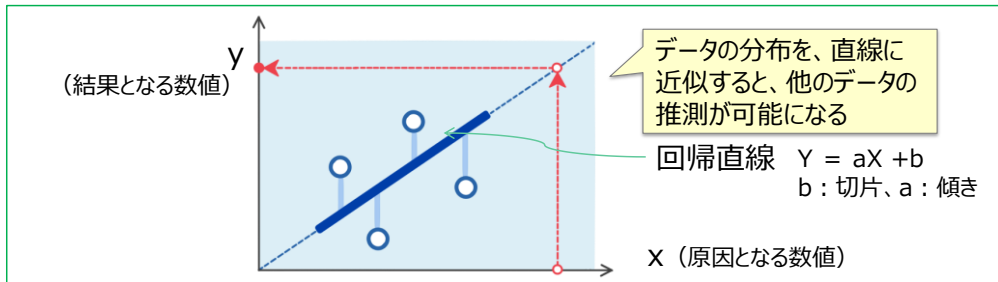
相関関係と因果関係は異なるので注意が必要です。

因果関係は原因に対して必ず結果が発生する場合を指しますが、相関関係では相関があるというだけで、必ずしももう一方の値が連動されることを保証するものではありません。

## 回帰分析と重回帰分析

### 回帰分析とは

「原因となる数値」と「結果となる数値」との関連性を、統計的手法によって分析することを言います。  
回帰分析を行うことで、一方の数値（説明変数）の変化から、もう一方の数値（被説明変数）の変化を推測することができ、仮説を立てることが可能になります。式にすると、 $Y = aX + b$  のように表されます。



### 重回帰分析とは

1つの目的変数を複数の説明変数を用いて推測する統計的手法のことを言います。式にすると、 $Y = a + b_1X + b_2X + b_3X + \dots + b_nX$  のように表されます。

### ・説明の流れ

データをグラフ化した場合に、傾向などを見たい場合があります。

数値が直線に集まっている場合は、線形回帰もしくは、一次回帰直線を引くことができます。

何らかの数式に表せる形なため、線形だけでなく2次曲線や3次曲線、4次曲線などと傾向が一致する場合があります。

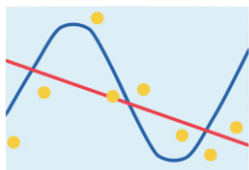
この場合は2次回帰曲線、3次回帰曲線などとなります。

また、重回帰分析では、一つの説明変数ではなく、複数の説明変数で数式を形成するものです。

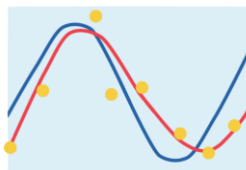


## 多項式回帰曲線のイメージ

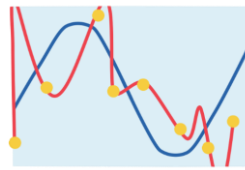
回帰曲線を用いて、データをモデル化することを考えます。



1次回帰線



3次回帰線



9次回帰線

ここでは3次回帰曲線を用いて表すのが適当だと考えられます。  
上の図からも分かるように、9次回帰線は全ての点を通っていますが、実態とはかけ離れています。従って、新たな点が現れた時にその点が9次回帰線上にある確率は却って下がってしまい、予測には向いていません。  
これを、過学習（オーバーフィッティング）といいます。

### ・説明の流れ

多項式回帰曲線の例です。

このように次数を増やせば全ての点を通す数式も得られますが、逆に予測精度が落ちてしまいます。

これをオーバーフィッティングと呼びます。

# テキストマイニング

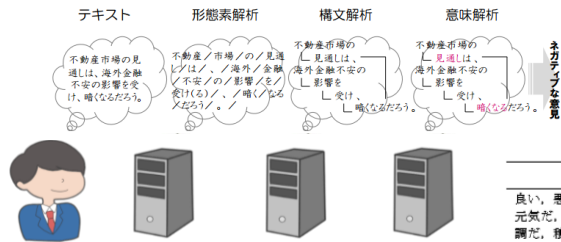
## テキストマイニングとは

大量のテキストデータから、トピックの傾向や、役立つ知識、情報などを見つけ出す分析技術のことを言います。

まず膨大に蓄積されたテキストデータを単語や文節（形態素）に分解し、それぞれの形態素の出現頻度や相関関係を解析します。

そして、係り受けなどの関係や時系列の変化といった構文解析を行うことで、結果的に客観的な意味内容を掴むことを目的とします。

図表 テキストマイニングの流れ（イメージ）



図表 評判辞書のキーワード例（抜粋）

ポジティブ単語	ネガティブ単語
良い、悪い（否定）、良好だ、活発だ、元気だ、効果的だ、最適だ、最高だ、順調だ、積極的だ、調子よく、適切だ、明るい、しっかり、安全だ、快調だ、経済的だ、上々だ、心強い、抜群だ、必要だ、便利だ、満足だ、等	悪い、良い（否定）、暗い、異常だ、割高だ、苦しい、最悪だ、最低だ、弱めだ、心配だ、不安だ、不十分だ、不調だ、危険だ、苦しい、酷い、少な目だ、心細い、辛い、大変だ、難しい、悲惨だ、不要だ、無理だ、真様だ、等

出典：「テキストマイニングによる国土政策評価手法の研究」国土交通省

## ・説明の流れ

ビッグデータで扱うデータに、通常の会話など言語データも含まれてきます。

これらを数値化するにはどのような手法があるのでしょうか。

実際には文章を単語や文節などの小さい単位に分割（形態素解析）して語句の出現頻度、語句と語句の関連性から文章内容の傾向を読み取ることが可能です。

例えば「やばい」という言葉が、ポジティブな言葉と一緒に出現するかどうかを測ったり、ある語句グループが出現したりする群をクラスタリングして、対応を打つなどの使い方があります。

## テキストマイニング

テキストマイニングをすることで、テキスト情報から何らかの有用な情報を得ることができます。

- 例 ・ 自社の評判と他社の評判を分析、比較する。  
→トピックの傾向を把握したり、新たな気づきを得たりできる。
- ・ 売上やキャンペーンデータについて、時系列で比較する。  
→マーケティングなどに有用な情報を得られる。

### ・ 説明の流れ

テキストマイニングをすることで、Twitterの発言やサイト上の口コミから、キャンペーンが効果的であったか調べることが可能になるかもしれません。

## 第6章

# ビッグデータとAI、機械学習

クラウド、IoT

## AIの発展

### AI (artificial intelligence) とは

コンピュータ上などで、人工的に人間と同様の知能を実現させようという試み、あるいはそのための一連の基礎技術のことを言います。

- ・エキスパートシステムとルールエンジン  
エキスパート（専門家）がルールをつくりプログラムに実行させ、推論機能を適用することで結論を得る。  
エキスパートシステムは大量の既知情報を処理し、関係性を導ける。
- ・事例ベース推論（CBR）  
類似した過去の事例を基準に、修正をしながら試行を行い、結果と事例を事例ベースに記憶する。
- ・ベイジアン・ネットワーク  
因果関係を確率により記述する。  
複雑な因果関係の推論を有向非巡回グラフ構造により表しながら、個々の変数の関係を条件つき確率で表す確率推論のモデルである。

### ・説明の流れ

ビッグデータ蓄積後の利用の一例として、AIや機械学習の学習データとしてビッグデータを利用する、もしくは学習結果を得るためにビッグデータを蓄積するという利用方法が広く存在します。

当初のAIは人間が知識をコンピューターに入力する方式をとっていましたが、ベイズ推定などに代表されるように推論エンジンも登場し始めました。

## 機械学習

### 機械学習と深層学習

2005年以降は機械学習と深層学習による第3次AIブームとなっています。

### 機械学習

データから反復的に学習し相関を見つけ出すことです。また、その結果を元にして将来を予測することができます。

予測分析におけるモデル構築の自動化ができるため、データサイエンティストの人材不足を補うものとして期待されています。

### 深層学習

ニューラルネットワークの多層化、1990年代の視覚野の研究や、ブルーノ・オルスホーゼンによるスパース・コーディング理論を基にしたアルゴリズムが実装されたものを指します。

情報が第1層からより深くへ伝達されると、各層で学習が繰り返され、特徴量(問題の解決に必要な本質的な変数であったり、特定の概念を特徴づける変数)が計算されます。

### ・説明の流れ

第3次AIブーム以降、機械学習が登場しました。

大量のデータを学習し相関値を元にモデルを生成し、同じ傾向の新規データの動きを予測することが可能となりました。

この仕組みはレコメンドエンジンなどで広く使われており、人が介在しないAIの道を切り開きました。

深層学習(ディープラーニング)とは、機械学習の延長でパラメーター(説明変数)を3層同時に適用できるというものです。

画像や音声データの特徴量(特徴的な変数)、例えば画像の輝度の方向などを繰り返して抽出する事で、画像の一致の判断などをかなりの精度で行う事が可能となります。

### ・ポイント(絶対に覚えてほしいこと、など)

ディープラーニングを行うためには大量の学習データ

と高速なGPUマシンなどが必要。

## 教師あり学習と教師なし学習

機械学習の手法として、教師あり学習と教師なし学習があります。

### 「教師あり学習」のイメージ 入力データに対応する出力を予測

- ① 入出力のペアで訓練データを用意し、  
入出力の関係を学習させる



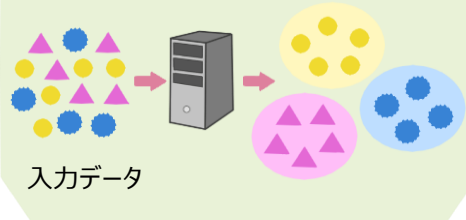
- ② 未知データを正しく判断できるように  
する



→ 迷惑メール判定、株価予測などに  
使われている

### 「教師なし学習」のイメージ 入力データの特徴を学習して分類

- ① 入力データのみを与え、データの特  
徴をつかみ、分類を行うもの



→ クラスタリングなどを用いて、同じ集合の  
持つ特性などから結果を予測する

### ・説明の流れ

機械学習には回答データを必要とする教師あり学習と、必要としない教師なし学習があります。

教師なし学習はクラスタリングのように、集合の集まりがあるかどうかを模索し、集合が取れば、どの集合に属するかで、新しいデータの特徴を類推します。



# 機械学習アルゴリズム

## 機械学習アルゴリズムの分類

- Classification/Class probability estimation  
既存のデータをクラスに分類し、未知の新規データの分類を予測する。  
新規データに対しては、属する特定のクラスを示したり、各クラスに属する期待値も計算したりする。  
例 商品の買い替え時に、もともと使っていた製品から購入を予測する
- Regression (回帰分析)  
既存データの関係を示す、数的な「関数」を推測し、定量的な予測をたてる。  
例 広告費と売り上げの相関を見つける
- Similarity matching  
新規データの、既存データへの類似性を予測する。

- 説明の流れ  
その他の機械学習アルゴリズムをご紹介します。

## 機械学習アルゴリズム

### 機械学習アルゴリズムの分類

- Clustering  
具体的な指標を示さず、既存データを自然に分類できるグループを見つける。  
分類後、各グループの特性を別のアルゴリズムで分析する。  
例 顧客をグループ化し、それぞれに別のサービスを提供する
- Co-occurrence grouping  
既存データから、同時にまたは付帯的に起こる事象を推定する。  
例 Aを買った人は、Bも買っている
- Reinforcement learning(強化学習)  
環境との相互作用を元に、データを収集し分析する。  
例 囲碁プログラムがコンピュータ同士の対局から有効な手筋を発見する

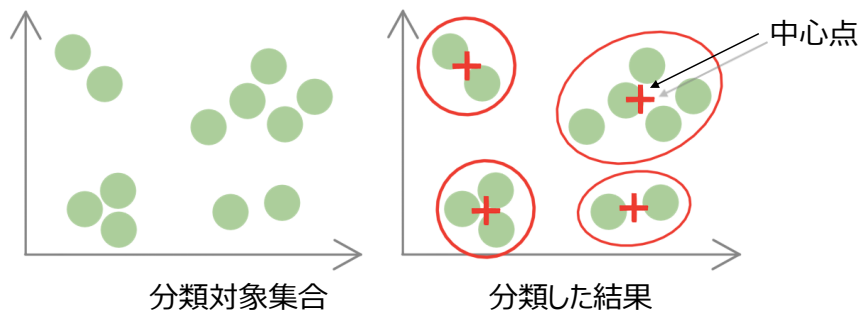
- 説明の流れ  
その他の機械学習アルゴリズムをご紹介します。

## 機械学習アルゴリズム

### クラスタリング (clustering)

データの集合を、具体的な指標を使わず性質が近い部分集合にすることです。

統計解析や多変量解析の分野ではクラスター分析 (cluster analysis) とも呼ばれ、基本的なデータ解析手法としてデータマイニングでも頻繁に利用されています。



#### ・説明の流れ

クラスタリングは、ランダムに決めた中心点で分類が可能かどうかを繰り返し実施することで、分類を試みる手法です。

分類した内容がどのような意味を持つかは、人が判断するか、別のアルゴリズムで決定します。

## 機械学習アルゴリズム

### 協調フィルタリング

商品の閲覧と購入のデータから、人同士の類似性や商品間の共起性をアソシエーション分析（相関分析）で解析し、対象者の行動履歴と関連づけることで、パーソナライズされた商品を提示する手法です。

協調フィルタリングは、商品スペックの関連性や商品閲覧の共起性だけではなく、購買データを基に人と人の類似性も重要視します。それにより、対象者に似ている集団が持つ特徴の中から、意外性のある商品を提示すること（セレンディピティ）もでき、コンバージョンレートのアップが期待できます。

### ・説明の流れ

協調フィルタリングは、行動履歴の相関値をとり、同じ行動をとるユーザーをモデルとして分類し、新しいユーザーに対して行動を予測するなど広い用途に使われています。

## 画像分析手法

画像データを解析し、顔認識や、画像の分類などを行う事が可能となります。

- SIFT  
特徴点周りの輝度から最も輝度変化が大きいベクトルを調べ分類することにより、細かい特徴を把握することができる。
- Haar-like  
形状やパーツが固定的なものに対し、平均的に示される特徴から画像を解析する。カメラの顔認識などに用いられる。
- 畳み込み深層学習 (CNN)  
画像をピースごとに比較して、位置や形状の特徴が類似する箇所を検出する。その上で、色、明度、エッジなどの多層において学習を繰り返す。顔認証などに用いられる。

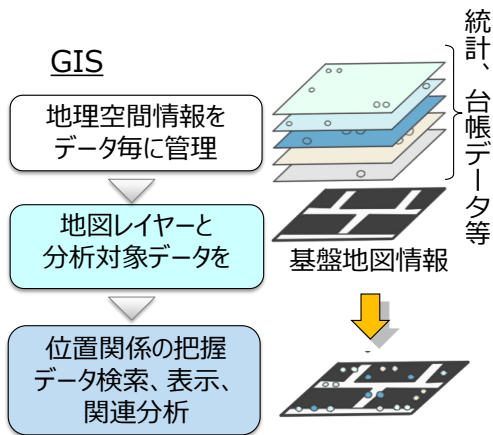
### ・説明の流れ

画像分析の手法にもいくつかの種類があります。

画像の輝度の方向などの特徴点から画像を分類するSIFTや、カメラなどに搭載されている顔のパーツの特徴から、顔を認識するHaar-like、大量の画像データを学習し、人物を見分ける畳み込み深層学習などがあります。

それぞれに得意、不得意があるため、適切なアルゴリズム選定が重要となります。

## 地図情報システムとの連動



### GISの特徴

- 地図データとビッグデータを重ね合わせ、新たな発見を得ることができます。
- 地理的な位置情報との情報の関連性を視覚的に理解することができたり、図面上で情報の集約、整理をしたりできます。

### GISの利用シーン

- 災害時の、正確で素早く効果的な行動決定などに利用されています。

### ・説明の流れ

ビッグデータを地図上にプロットし、知見を得る場合もあります。

地図情報はGISシステムから取得し、緯度経度などを持つビッグデータを地図上で表現すると、災害時の動態分析などが実現可能となります。