PostgreSQLを使用した
大規模データ解析の勘所

OSC 2026 Osaka (2026/Jan/31 r1)

滋賀大学　データサイエンス学部

特任教授　佐藤 健司

目　次

1. 自己紹介

2. 大規模データのTable 設計

3. センシング側のDBクライアント設計

4. データ登録クエリーの設計指針

5. データのレプリケーションとバックアップ

1. 自己紹介

佐藤健司

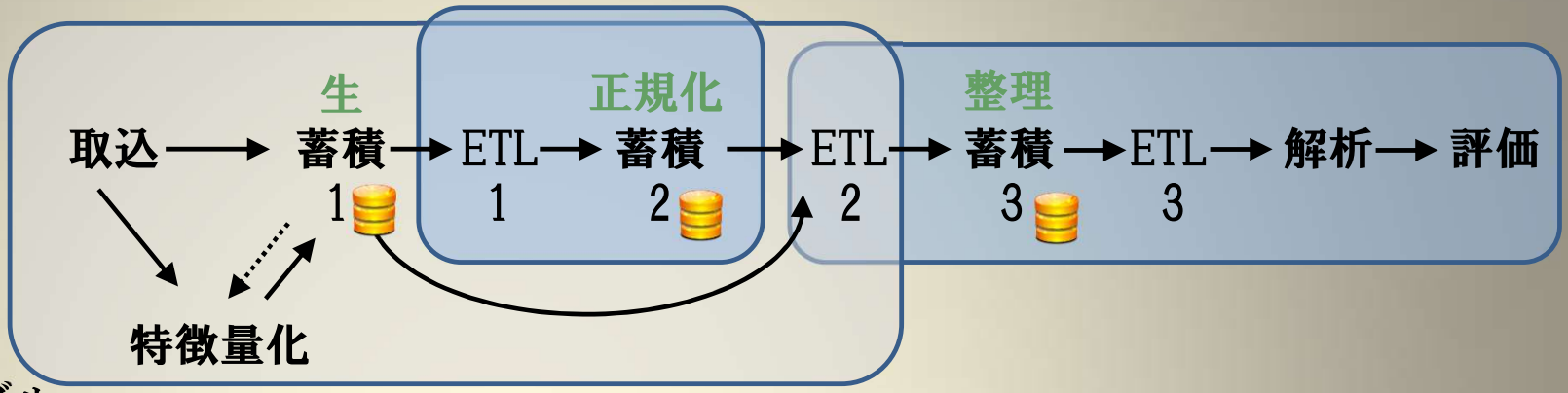
1986年住友金属鉱山(株)入社
1998年より画像処理検査装置開発を
2002年よりWeb+DBシステム開発を行い
2016年からはデータ解析システムの開発チームを立ち上げ、
IoT->DB->Web(見える化)->データ解析を、一気通貫に構築
2025年11月から滋賀大学 データサイエンス学部 特任教授

- 日本PostgreSQLユーザー会分科会理事(2022/6～) https://www.postgresql.jp/npo/director
- 製造現場におけるデータ活用セミナー2025(7/17) https://r-management.jp/MF_IoT/forum20250717/
- 製造業IoTカンファレンス2025夏(6/10) https://r-management.jp/MF_IoT/forum20250610/
- オープンセミナー2023@香川(9/9) IoT用DB設計 https://osk.connpass.com/event/292551/
- 関西DB勉強会(2023/1/21) IoT用DB設計 https://kansaidbstudy.connpass.com/event/268133/
- OSC長岡(2022/11/12) IoT用DB設計 https://ospn.connpass.com/event/261361/
- DXのゴールはUXなのか？(2022/3/8) https://www.youtube.com/watch?v=ePrndUmu5_4
- データ解析チームのスタートアップとスケールアウト
  製造業IoTカンファレンス2021冬(11/10) https://r-management.jp/MF_IoT/forum20211110/
- AI・IoT時代に製造業が取り組むべき分析組織の作り方 update版
  RapidMiner Conference 2021(4/8) https://www.rapidminer.jp/conference2021.html
- AI・IoT時代に製造業が取り組むべき分析組織の作り方
  第3回 AI人工知能EXPO(2019/4/5)
- SMMにおけるIoTとデータマイニングの一例
  RapidMiner東京ロードショー2017(2/16)

目　次

2-1. データ解析の為に必要なDB技術

- ETL (3)
  解析用の抽出
- 蓄積 (整理)
  解析目的別の
  巨大な単一テーブル
- ETL (2)
  解析用の抽出、異常値処理、欠損値補間、移動平均
- 蓄積 (正規化)
  通常のWebシステムと同様
- ETL (1)
  ロット紐付け
- 蓄積 (生)
  月や年単位でのテーブル・パーティショニング



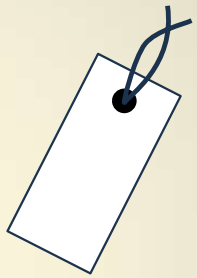
データ前処理の
多くをDB内演算で
済ませている

解析ソフトの
負荷が低くなる

2-2. 大規模データのTable 設計1(基本)

10,000行/60s x 3600s/h x 24h/day x 300day/year
= 4.32G行/year/工場

id, 測定時刻, 測定値, 測定属性の4列だと、
約48byte/行 程度なので、
=207GB/year/工場



- 測定の前提(例)
  - Tag点数 = 10,000Tag/工場
  - 間隔 = 60s
  - 期間 = 24h, 300day

測定値テーブルの構造と内容(例)

id mno (SERIAL)	測定時刻 mtime (TIMESTAMP)	測定値 mval (DOUBLE)	測定属性 (matt) (INTEGER)
5698	2026/1/31 13:00:00	95.6	1654
5699	2026/1/31 13:00:03	0.452	1655

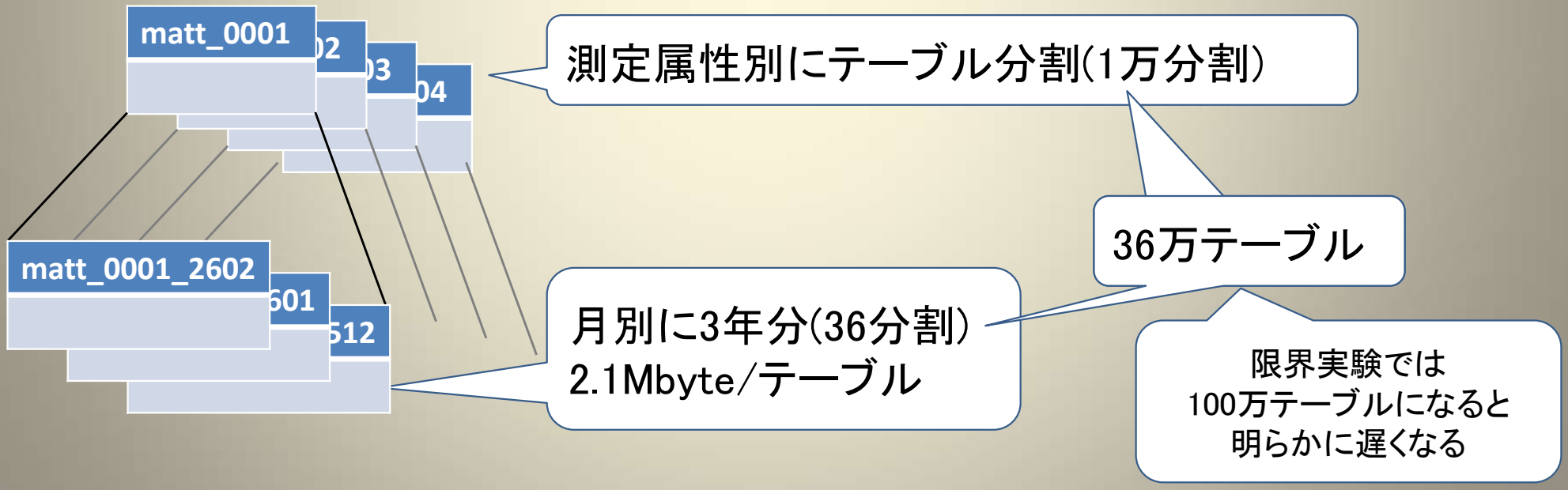
どの場所の、
どの種類(温度/圧力/流量)の
どんな単位(℃/Pa/kPa/etc)なのか？は
別テーブルでマスター化しておく

2-3. 大規模データのTable 設計2(パーティショニング)

データが溜まり、テーブルサイズが肥大化すると、
必ず遅くなる

パーティショニングで分割しておき、
所定のサイズより大きくならないように管理する

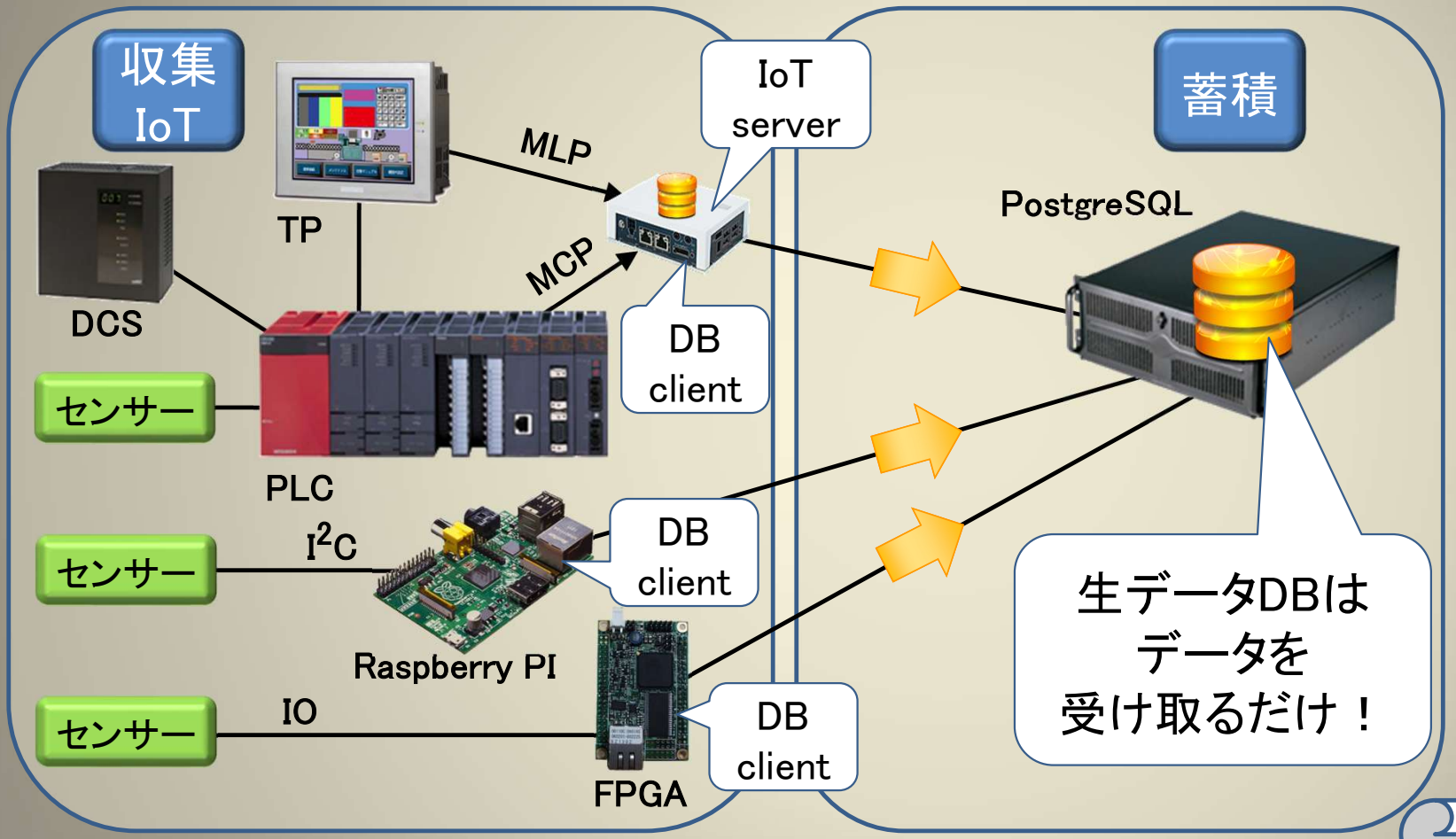
1行/60s x 3600s/h x 24h/day
x 30day/month/Tag
= 43.2k行/month/Tag
≒ 2.1Mbyte/Tag



2-4. IoTからデータ解析までのDB構成

目　次

3-1. センシング側からDBへプッシュ



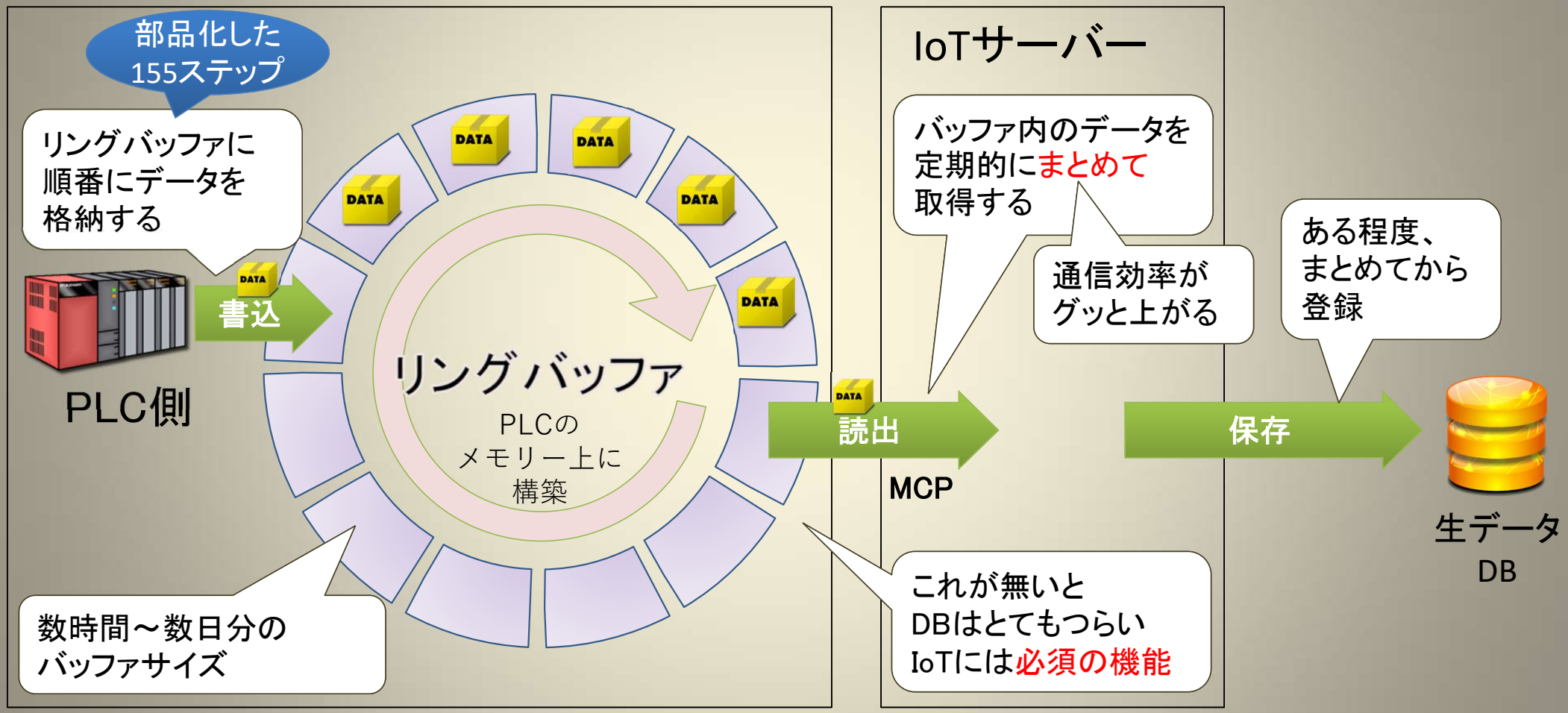
4章で詳述

毎分1万件のデータを個別に登録しようとすると、167tpsのInsert intoを実行することになる

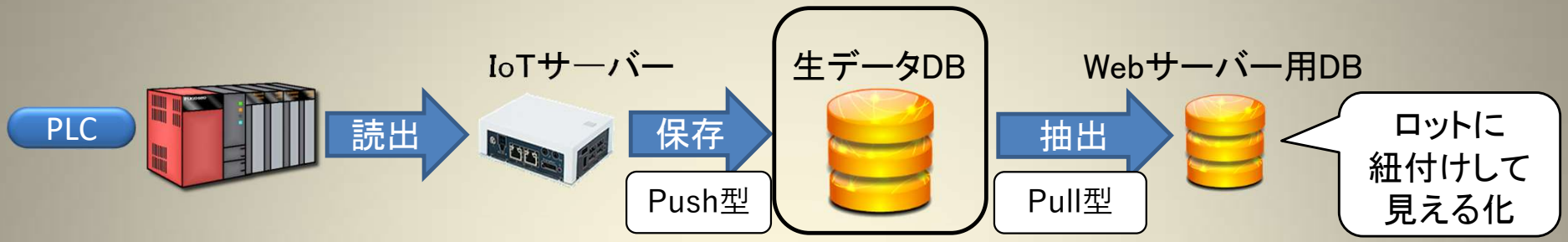
※1. MLP : Memory Link Protocol

※2. MCP : MELSEC Communication Protocol

3-2. PLCに追加したリングバッファ機能



3-3. 生データDBの設計指針

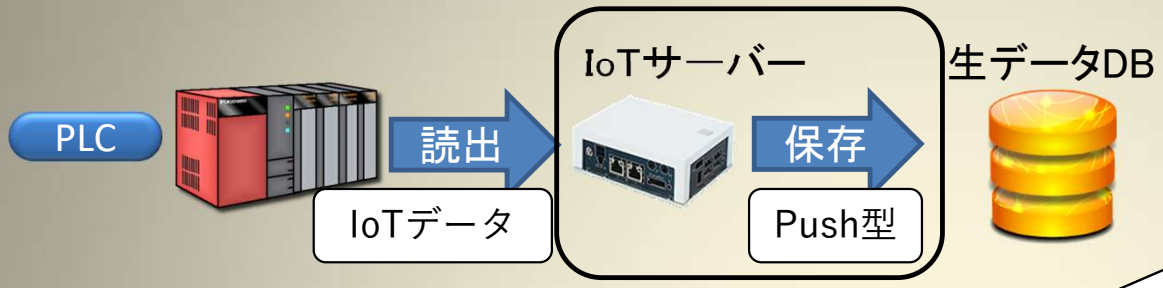


1. 休みなくデータ保存と読み出しが続くので、(偶であっても)アクセスが遅くなる要因は厳禁
2. 従ってVacuumはしない
3. CPU負荷を平準化するために、データ収集プログラムはDBサーバーでは動かさずにIoTサーバーなどの周辺機器で動かし、データをPushする
4. データの利用側からPullする
   (生データDBには、データ収集やデータ**配信プログラム**を置かない)
5. 測定箇所(tag)毎にテーブルを作ると共に、月別でテーブルパーティショニングする
   (例えば、1万tag, 3年分で36万テーブルになるが、動作は可能)
6. 生データDBにはデータのinsertと読出しの為のselect以外の何もさせない
   (削れる負荷は削ぎ落とす)

2章で詳述済

目　次

4. データ登録クエリーの設計指針



5～10分程度がお勧め

1. IoTデータの時間軸をできるだけまとめて送信する方が、
   生データDBの負荷が軽くなるが、まとめすぎると、
   見える化のレスポンスが悪化するので、バランスを考える
2. 沢山の測定箇所をまとめて送信する方が、生データDBの負荷が軽くなるが、
   クエリーが長くなるので、限界を見極め、程々の長さで

いろいろ、まとめる事で、トランザクション・コストを低減させている
（長いクエリーでTPSを減らす）

関係するプログラムのバッファサイズに依存(数kbyte?)

目　次

1. 自己紹介

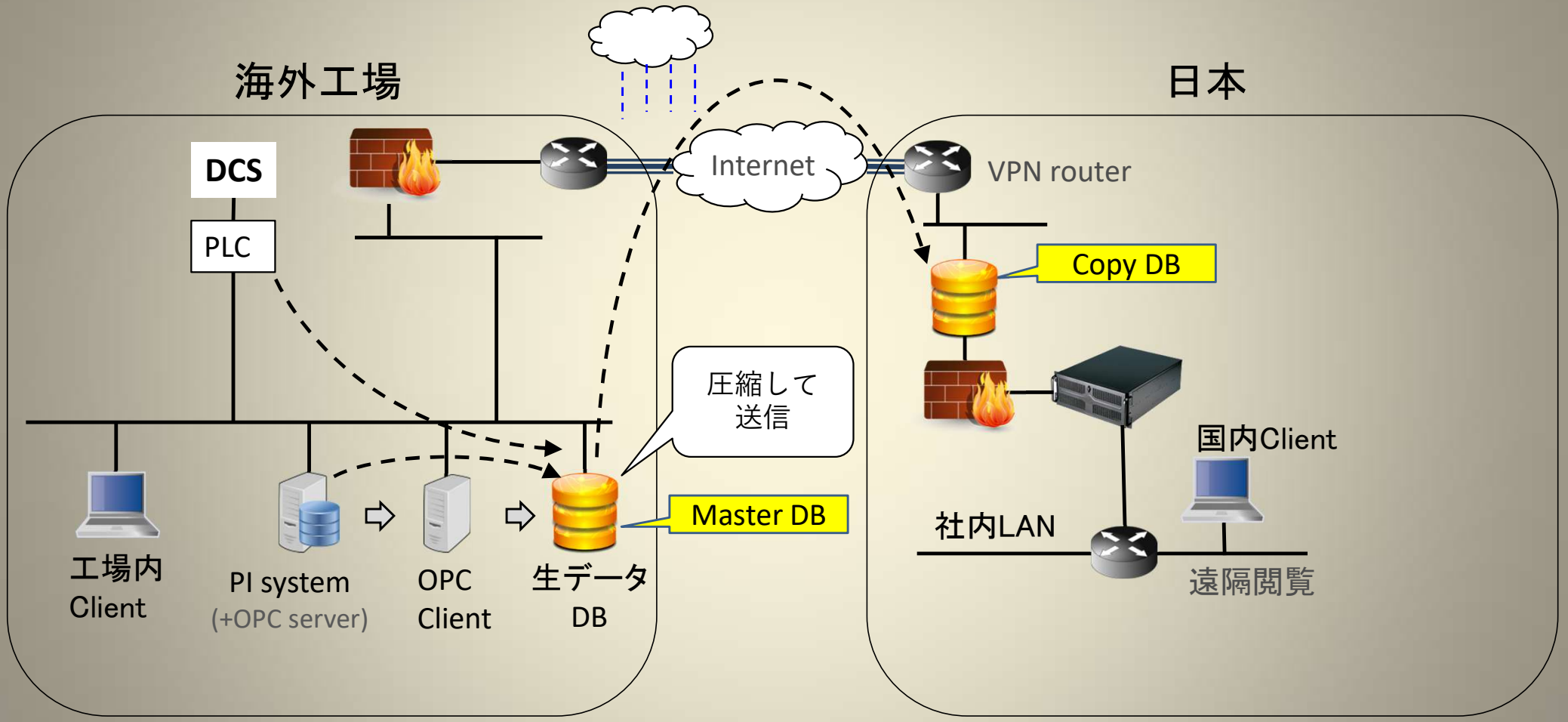
2. 大規模データのTable 設計

3. センシング側のDBクライアント設計

4. データ登録クエリーの設計指針

5. データのレプリケーションとバックアップ

5-1. データのレプリケーションとバックアップ



5-2. データ転送の考え方

- 回線の帯域が十分ではなく、安定度が低い場合は、
  データ転送に工夫が必要になる
- 全てを転送できない場合は、優先度の高いデータだけを
  internet経由で転送する
- ある程度、まとめたデータを圧縮し、シリアルNo.を割り振ってからftp転送する
- 転送失敗が度々あったので、シリアルNo.を基に再送要求した
- 再送要求は、1回行って失敗したら、暫く待ってから行う(例えば15分)
  失敗回数が多い再送要求は、より時間を開けて行わないと、
  再送要求がパイルアップしてしまうので注意が必要
- 優先度の低いデータはFedExや帰国者を使って転送できる
  100GB/5日 = 0.23Mbps　1000GB/5日 = 2.3Mbps

5-3. バックアップの考え方

- バックアップは一次データのみで十分 （後段のデータは再作成できるから）
- バックアップの保存期間は、(おそらく)3年程度で十分
  - (大抵)工場内の生産方法が毎年更新される
  - 古いデータは古い工程のデータであり、今の解析に役立たない （場合が多くなる）
  - 解析が洗練されてくると、僅かな生産方法の変化に敏感になってくる
    （工程が違う昔のデータの価値は薄れる）

ま　と　め

2. 大規模データのTable 設計
   大規模なパーティショニングでテーブルサイズを小さく

3. センシング側のDBクライアント設計
   DBサーバーはDB管理だけをさせ、
   それ以外は周囲のサーバーが対応する

4. データ登録クエリーの設計指針
   できるだけ纏め、長いクエリーで

5. データのレプリケーションとバックアップ
   転送失敗を見込み、再送は程々のゆっくりで