データセンターの過去情報のためのサブシステム(PS DataStore)の設計と実装

2016年2月18日

株式会社ネットワーク応用通信研究所　　橋本　将、原　悠

NaCl

NaCl

自己紹介

■**橋本 将**
- ネットワーク応用通信研究所 研究員
- アプリ、検索処理(SL)を担当

■**原 悠**
- ネットワーク応用通信研究所 研究員
- 蓄積処理、検索処理(BL)を担当

NaCl

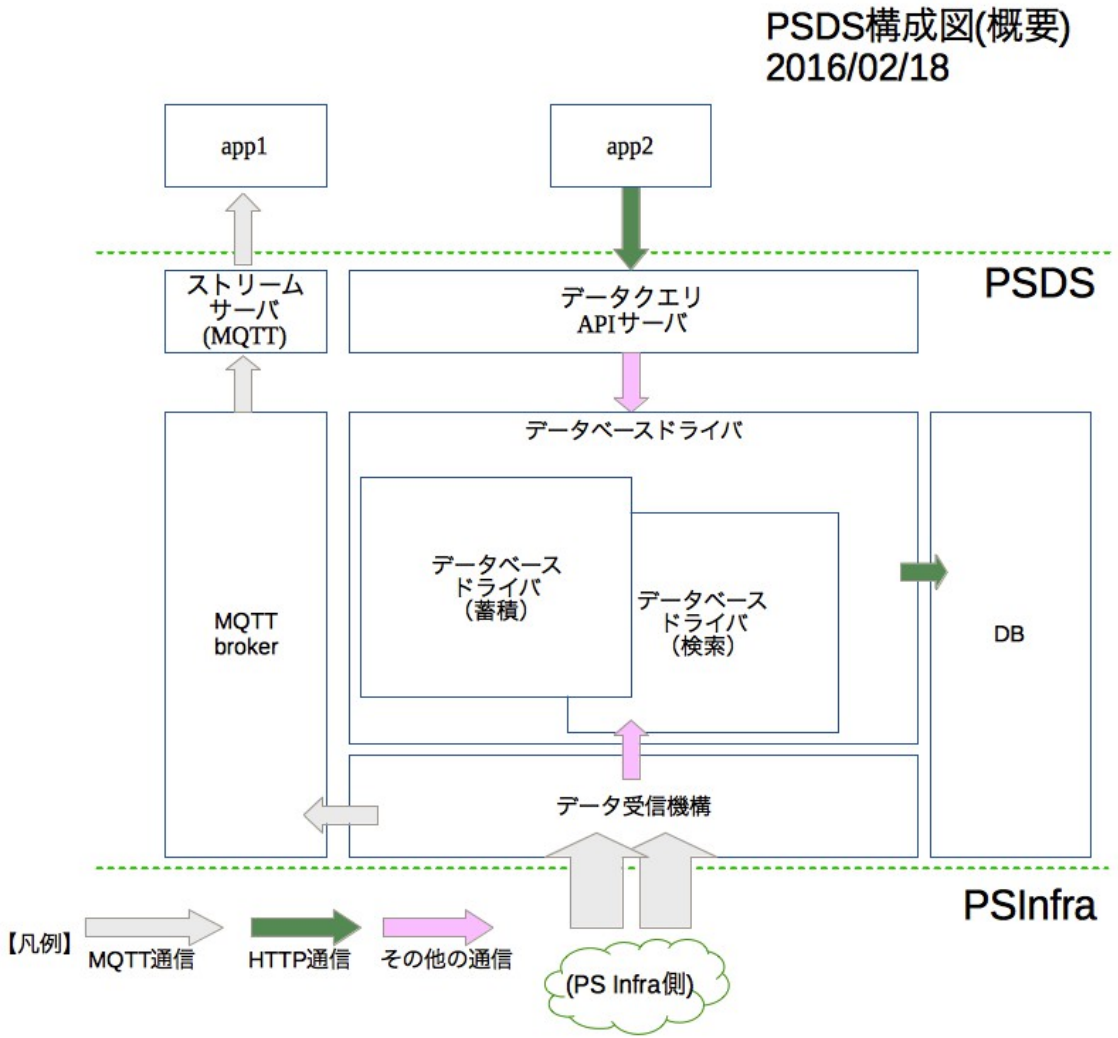
PS/PSDSとは

■**PS(PrairieStack)**
- DCをプログラムで制御
  - 可用性の向上
  - エネルギー効率の向上
  - 運用性の向上の実現

■**PSDS(PrairieStackDataStore)**
- PSのサブシステムの1つ
- DCのセンサ情報(過去情報)の蓄積/提供
- センサ情報の収集はPSInfra

NaCl

PS/PSDS/PSInfraの構成



NaCl

本発表の流れ

■**PS Dashboardの紹介**
- 前述の構成図のapp部分
- DCのセンサ情報(過去情報)を取得、表示

■**PSDSの一部を紹介**
- PS Dashboardから見たPSDSの設計意図
  - PSDSを用いてappを簡単に開発できるようにする

NaCl

PS Dashboardの紹介(デモ)

■機能

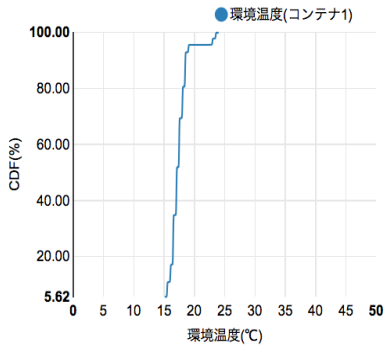
- センサ情報を元にヒートマップ/累積分布を描画
  - ヒートマップ: センサーの種類別、全ラックの情報を描画
  - 累積分布: センサーの種類別に描画
- 目的

画面	目的
ヒートマップ	画面を元に空調制御
累積分布	センサーの絞り込み (40℃以上のCPUの特定など)

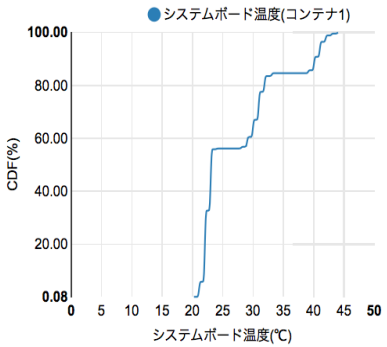
PS Dashboardはデモ用だが、こういったものを簡単に作りたい

画面の例1

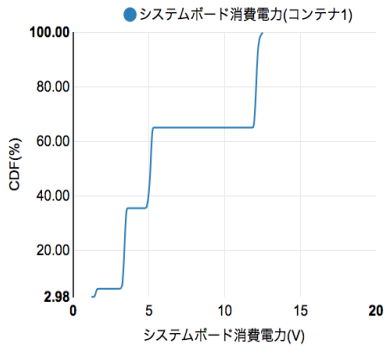
累積分布一覧



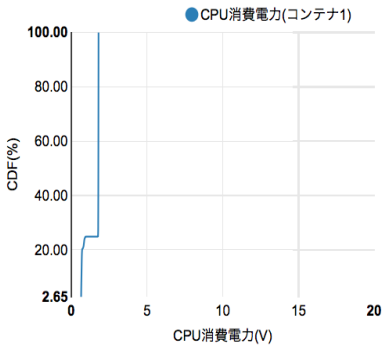
詳細表示



詳細表示



詳細表示

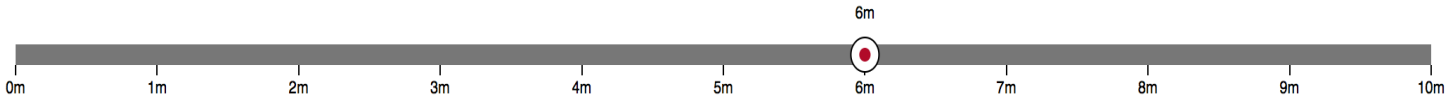


詳細表示

画面の例2

ヒートマップ(IT Server各種センサ)

過去の表示(現在時刻から指定時間前)



ラックの状態

	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
41																		22	17	23	4.9		3100	
40	19.5	23	30	22	3.4	1.8	5760	92	19	23	32	22	3.4	1.79	6120	92	19	24	31	23	3.44	1.79	6120	88
39	20.5	24	32	22	12	1.78	6480	92	18	24	34	22	3.4	1.78	6000	92	19	25	34	23	5.1	1.79	6000	92
38	19.5	24	35	23	12	1.79	5640	92	20.5	23	31	22	5.13	1.79	6120	92	19.5	24	32	23	3.44	1.78	6120	88
37	18.5	24	35	22	3.4	1.79	6000	96	20.5	24	32	22.5	5.13	1.79	6000	92	20.5	24	34	23	12.06	1.79	6360	92
36	20	24	33	22	12	1.79	5760	92	19.5	24	33	22	3.42	1.78	6480	92	19	24.5	35	23	3.4	1.78	5940	94
35	20	24	34	22	12.06	1.78	5760	92	18	24	33.5	24	5.13	1.79	6120	104	20	24	33	23	5.13	1.79	6120	92
34	18.5	24	32	22	5.13	1.79	6000	92	19	25	35	23	3.42	1.78	6120	92	19.5	25	35	24	3.42	1.77	6120	108
33	18.5	24	35	23	5.1	1.79	6000	96	19.5	24	37	23	3.4	1.78	5880	92	21	25	35	24	12.06	1.79	6120	100
32	20	24	36	22	5.13	1.78	6120	92	19	24	33	23	5.16	1.78	5880	96	20	25	36.5	26.5	3.4	1.79	6780	138
31	19	24	32	22	12	1.79	6360	92	18.5	24	34	22	12	1.79	5880	92	19.5	24	32	23	12	1.8	6120	104
30	19.5	23	31	22	5.13	1.78	5880	92	18.5	24	34	22	5.13	1.79	6000	92	19.5	24	33	23	12	1.79	6240	92
29	18.5	24	31	22	3.42	1.78	6120	92	17	23	35	21	3.4	1.79	6000	104	20.5	25	34	23	12	1.79	6000	96
28	18	24	33	22	5.1	1.79	6120	92	20	24	32	23	5.13	1.79	6240	104	19	25	37	25	5.13	1.8	6120	100

PSDSの設計/実装上の工夫について

NaCl

- このようなアプリを作りやすくするために

  A)過去情報をどのようなAPIで提供するか

  B)情報をどのように整理して蓄積するか

過去情報をどのようなAPIで提供するか

NaCl

■**PSDSでは同期/非同期API+ストリームを提供**

- 同期/非同期API
  - Query/QueryIntoTable: SQLを非同期的に実行
  - QueryAndWait: SQLを同期的に実行
  - それぞれでバックエンドが指定可能(以下)
    - SpeedLayer: InfluxDB
      - 一定期間(3日など)より前のデータは揮発
      - その代わりにレイテンシが小さく、高速に値を返す
    - BatchLayer: BigQuery/Impala
      - 長期間センサ情報を保持
- ストリーミング
  - MQTTで最新の情報のみを受け取り続ける
  - topicに指定したIDのセンサ情報が流れてくる

NaCl

SpeedLayer/BatchLayer

- 保存期間とレイテンシはトレードオフ関係
- 現在に近い情報ほど許容されるレイテンシが低い
  - データ取得のユースケースによる分類

	利用方法	データ保持期間	レイテンシ
Streaming	MQTT	なし(現在値のみ)	極小 (～0.5秒)
Speed Layer	HTTP (同期、非同期)	短い (設定で3日など)	小 (～5秒)
Batch Layer	HTTP (同期、非同期)	長い	大 (1分以上)

NaCl

想定したユースケース

1.制御アプリ
- ストリーミングを利用

2.PS Dashboard(前述)
- Speed Layerを利用

3.中長期的なトレンド分析等を行うアプリ
- Batch Layerを利用

NaCl

制御アプリ

■ 想定
- 障害に通じる情報→迅速に対応
- 例）サーバの電源オフ

■ 必要とした性質
- 情報をレイテンシ極小で受け取る事
  - 過去情報は重要ではない

NaCl

PS Dashboard

- 想定
  - ヒートマップなどを日々のチェックに利用
    - 広範囲の情報
  - 問題がある場合により詳細なチェック
    - 狭い範囲の情報

- 必要とした性質
  - 近期間の情報を繰り返しストレスなく取得できる事
  - 繰り返しの回数と金銭的な負担が比例関係にない事

中長期的なトレンド分析等を行うアプリ

NaCl

- 想定
  - 故障予測などの中長期間の過去情報に対して解析処理

- 必要とした性質
  - 長期間にわたるセンサ情報を処理できる事
    - 直近の情報をレイテンシ小で取得できる必要はない
    - 繰り返しの回数はすくなく、金銭的な負担と比例関係があっても問題ない

今回のAPI設計における夢と現実解

NaCl

■**夢(本当に欲しいもの)**

- 遅延なし、秒単位でいかなるデータも取得できるAPI
- お金がかかる
  - DCの規模に比例する台数のサーバが必要
  - →データ量の妥協などにつながり、特定の目的にしか利用できなくなりがち

■**現実解**

- ユースケースと金銭的な負担を元にストリーミング/SL/BL
- 利用頻度が高い直近の情報はストリーミング/SLで
  - インスタンス分程度のお金しかかからない
- 利用頻度が低く長期間の情報はBLで
  - 日々のクエリで少々お金がかかっても問題ない

PS Dashboardをはじめとする、DC向けのアプリを現実的な負担で簡単に作れる

PSDSがデータをどのように蓄積しているか

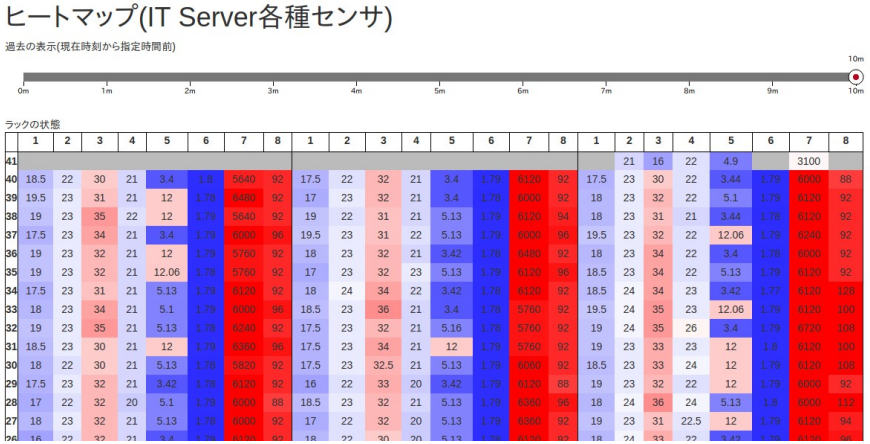
NaCl

- DC構成要素へのID付けについて
- 収集の流れと収集リクエストについて
- PSDSが受け取るデータについて

NaCl

DC構成要素へのID付け(1)

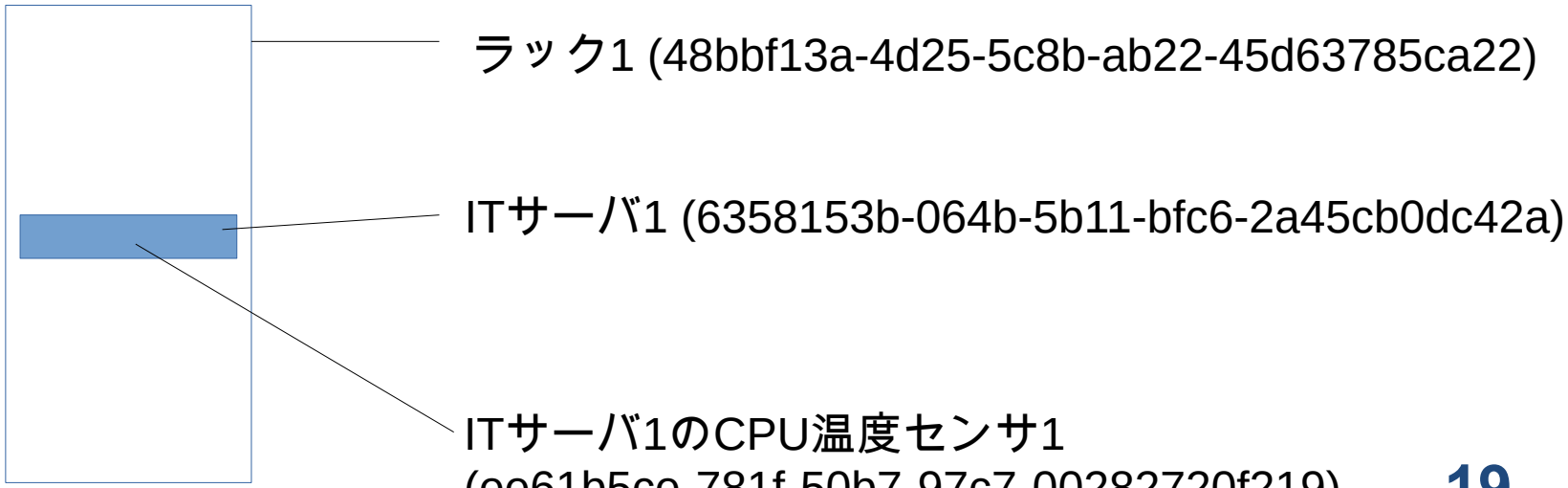
- DC内のある機器のあるセンサの値が、ある時刻にどのような値だったかを知りたい
- 「ある機器」や「あるセンサ」をポイントする方法が必要
- 方法1: IDを発行するサーバを用意し、DC内の機器を1つずつ登録していく
  - ⇒中央管理が必要
- ⇒ UUIDを生成するルールを共有



NaCl

DC構成要素へのID付け(2)

- ラック、機器、センサにそれぞれUUIDを振る
  - 将来的にはコンテナにも(現在はコンテナ1台だけなのでコンテナにはUUIDが付いていない)
- PSDS, PSInfraや、PSDSを利用するアプリケーションは、いずれもこのUUIDでDC構成要素をポイントする



NaCl

DC構成要素へのID付け(3)

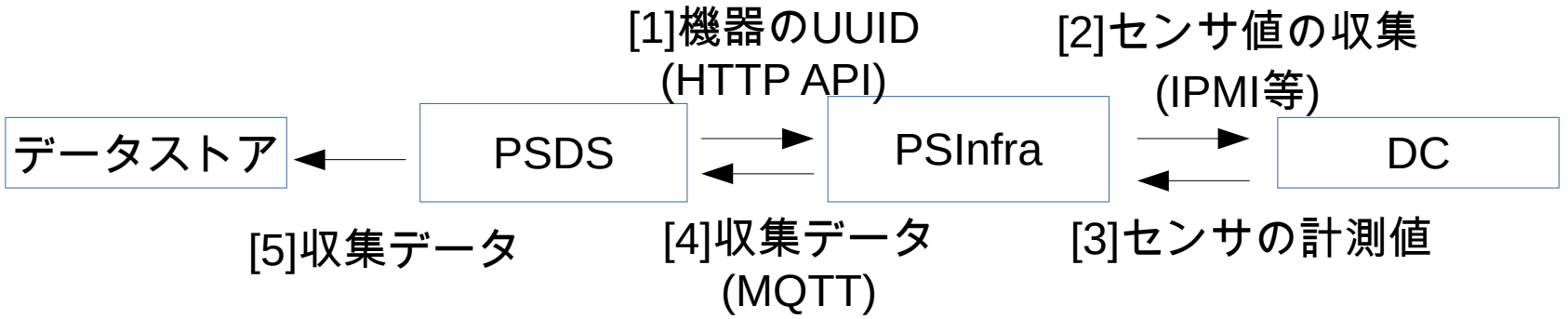
- **構造からUUIDを生成**
  - UUID v5を使用
  - 例：ITサーバなら、IPアドレスから生成
  - メリット
    - 中央サーバが不要
    - 例：ITサーバなら、IPアドレスを知っていればIDがわかる
  - デメリット(今後の課題)
    - 構造が変化した場合にどう対応するか
    - 例：ITサーバを別のコンテナに移動して、IPアドレスが変化した

NaCl

収集から蓄積までの流れ(1)

■ **収集の流れ**

- 「どの機器のどのセンサの情報を蓄積するか」はPSDS側に設定がある
- [1]PSDSは起動時にPSInfraに収集リクエストを送る[1]
- [2]PSInfraは指定された機器に対し一定間隔で収集を実行し、[3]取得した計測値を[4]MQTTで流す
- [5]PSDSは受け取ったデータをデータストアに蓄積する



NaCl

収集から蓄積までの流れ(2)

■**収集リクエスト**

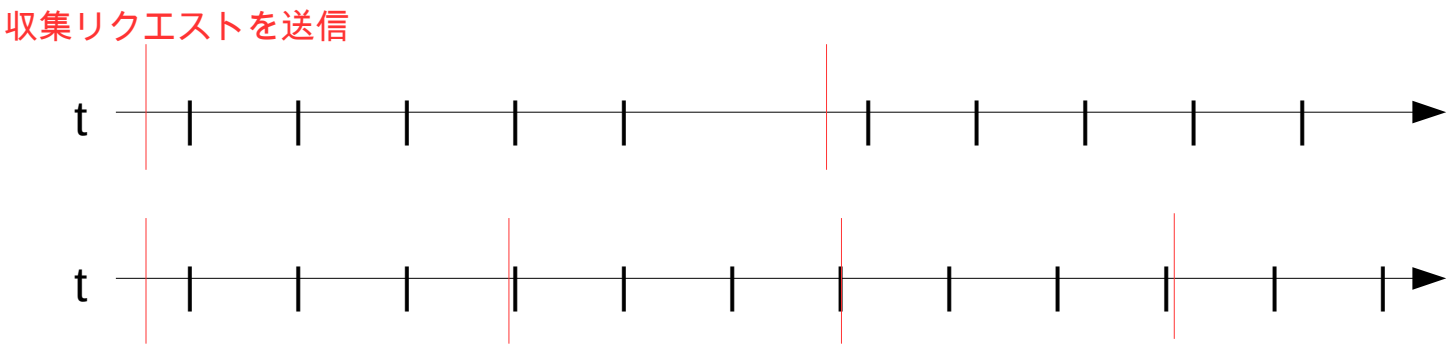
- PSDSからPSInfraに収集リクエストを送ると、MQTTでデータが流れてくる
  - ただし永遠にではない(一定期間で終了する)
    - バグがあったとき、リソースを食い続けるよりは死んでしまう方が安全なので
  - PSDSはデータが一定時間来ていないことをチェックし、収集リクエストを再送する

NaCl

収集から蓄積までの流れ(3)

■**収集リクエストの延長**

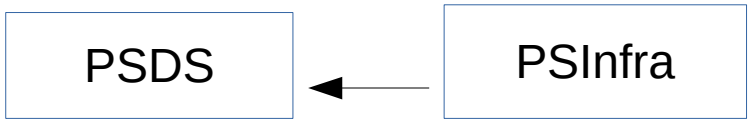
- 「タイムアウトした」ことを検知してから収集リクエストを再送するのだと、蓄積されるデータの間隔がそこだけ間延びしてしまう(上図)
  - タイムアウト時間は、期待される送信間隔の2倍なり1.5倍なりに設定するため
- ⇛収集リクエストの仕様を拡張し、送信終了予定時刻の8割くらいで「延長」できるようにして対処(下図)



NaCl

PSDSが受け取るデータ(1)

- **MQTT message payload**
  - PSInfra - PSDS間はMQTTで通信
    - メッセージは独自フォーマット(固定長ヘッダ + MessagePack)
    - ボディ部分は現時点では2種類のフォーマットがある (**schema 0**, **schema 1**)

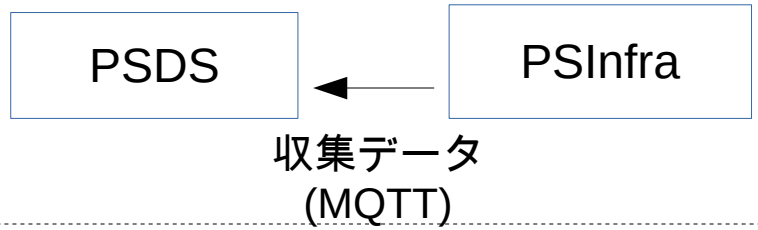


schema 0
(時刻, (センサID, 値), (センサID, 値), ...)
schema 1
(時刻,
(センサID, 集計期間, サンプル数, 最大値, 最小値, 平均値, 値),
(センサID, ...))

NaCl

PSDSが受け取るデータ(2)

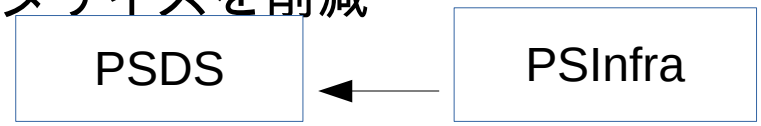
- **schema 0**
  - schema 0, 1とも、MQTTの1メッセージに複数のセンサの収集値が入っている
    - 負荷軽減(単位時間あたりのメッセージ数を減らす)ため
  - PSInfraは現状、機器内のセンサに対して一括して収集を行う
    - 収集した値を1メッセージにまとめたもの = **schema 0**



schema 0
(時刻, (センサID, 値), (センサID, 値), ...)
schema 1
(時刻,
(センサID, 集計期間, サンプル数, 最大値, 最小値, 平均値, 値),
(センサID, ...))

PSDSが受け取るデータ(3)

- **schema 1**
  - センサによってはとても細かい間隔で収集できるものがある
    - 例：PLCのセンサは秒間10回以上収集可能
    - 全ての収集値をデータストアに蓄積するとデータサイズが膨大になるが、そこまでの精度は必要とされていない
    - 一定期間の集計値のみを送信することで、情報量を落とさずにデータサイズを削減



収集データ
(MQTT)

schema 0
(時刻, (センサID, 値), (センサID, 値), ...)
schema 1
(時刻,
(センサID, 集計期間(例：1秒), サンプル数, 最大値, 最小値, 平均値, 値),
(センサID, ...))

NaCl

まとめ

■**PS Dashboardの紹介**

- 前述の構成図のapp部分
- DCのセンサ情報(過去情報)を取得、表示

■**PSDSの一部を紹介**

- PS Dashboardから見たPSDSの設計意図
  - PSDSを用いてappを簡単に開発できるようにする