Message Queue の DAQ への応用

五十嵐 洋一
KEK
計測システム研究会 2021.10.28

Motivation

- DAQ の全ての読み出し機器はバックエンドからネットワークを通して読める。
  - TCP/IP socket programing
    - Pros
      - 大方すべて制御できる。
      - 良く理解され、手法が確立している。
    - Cons
      - 複数の接続処理
      - パケット喪失、輻輳などは起こらないように運用
        - 適切なバッファリング
      - Non-blocking 読み出しの Know-How (time out や poll/select 等)
      - 接続の順番の管理 server side は先に立ち上がらないといけない。

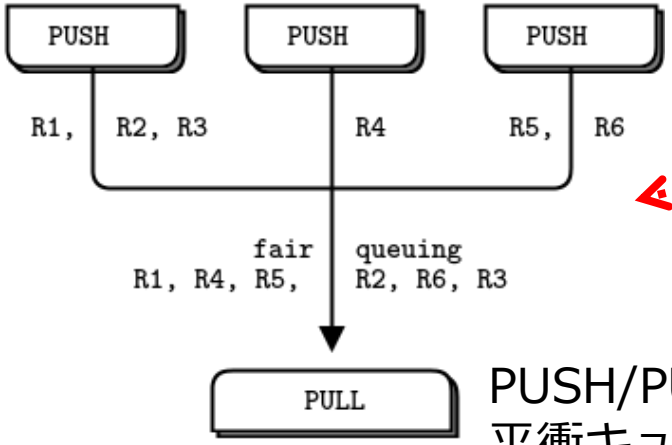
→それなりに手間がかかる

なにか新しくこなれたものはないかな?

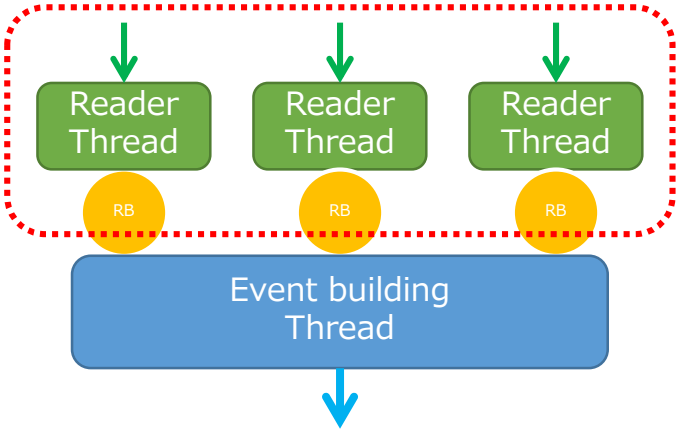
ZeroMQ

- なぜ ZeroMQ か?
  - 非同期メッセージレイヤーの一つ
    - 通信を TCP/IP, UDS の上に実現している。
    - コネクションの接続・切断を気にしなくてよい。
    - 長さとコンテンツだけの単純なフレーム
    - バッファリングを考えなくて良い。(といいなあ)
      - Memory allocation / free をよく管理する必要がある。
      - キューがいっぱいになるとブロックする。
        - Non-blocking だと思っていると止まることがある。
  - 多くの Linux distribution に含まれ、Windows を含めいろいろな OS で動いている。
  - 有用な通信モデルが構築されている。
  - スケーラビリティが高い。
    - たくさん接続しても破綻しない。(はず?)

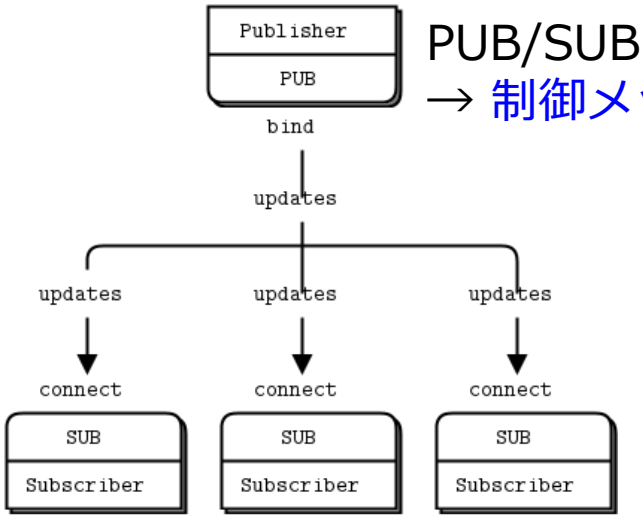
ZeroMQ 通信モデルの一部



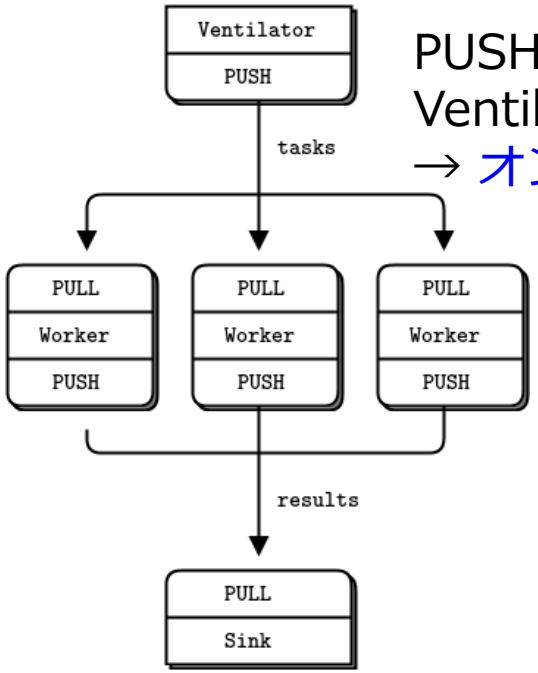
PUSH/PULL による
平衡キューイング



HDDAQ の Event-builder

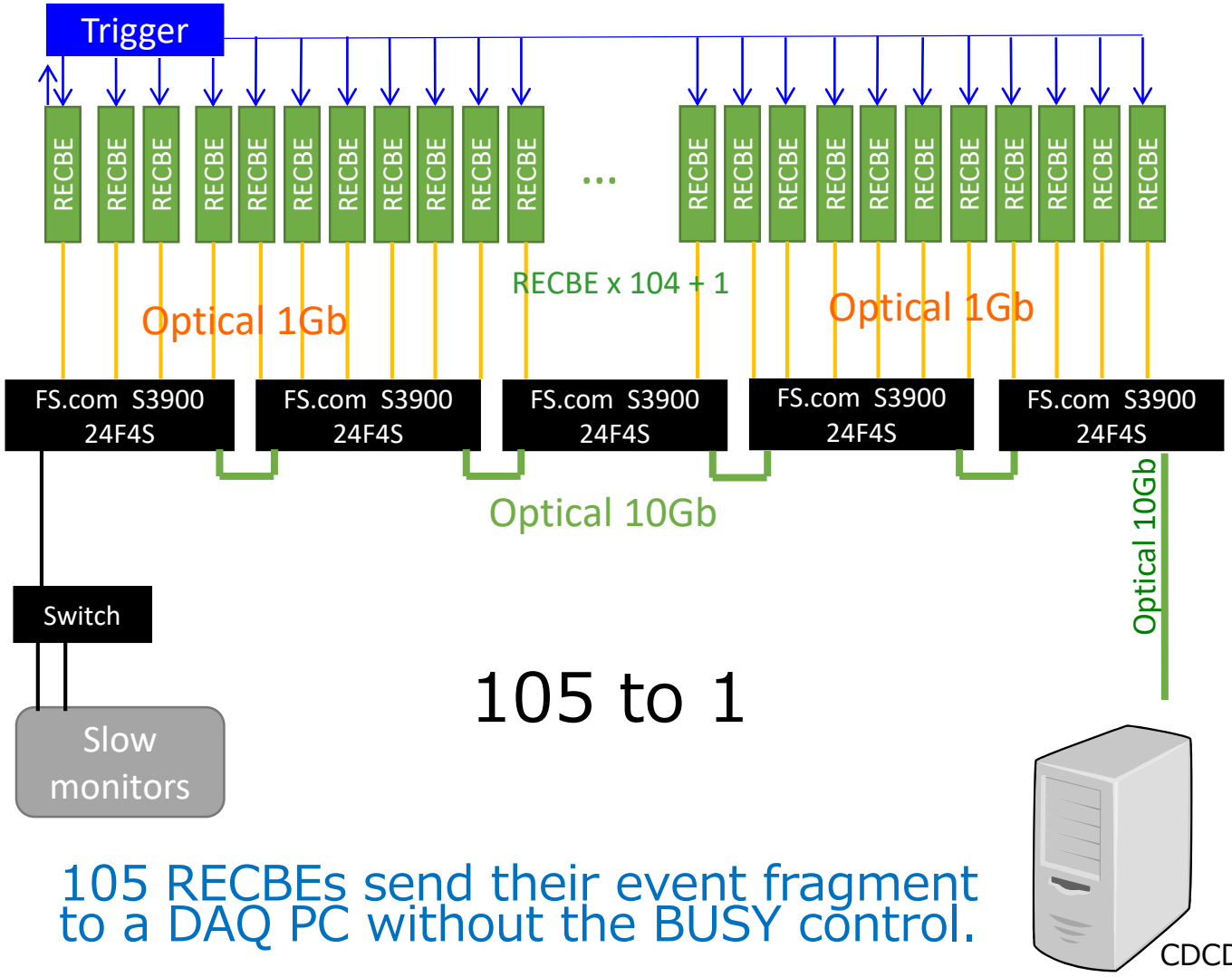


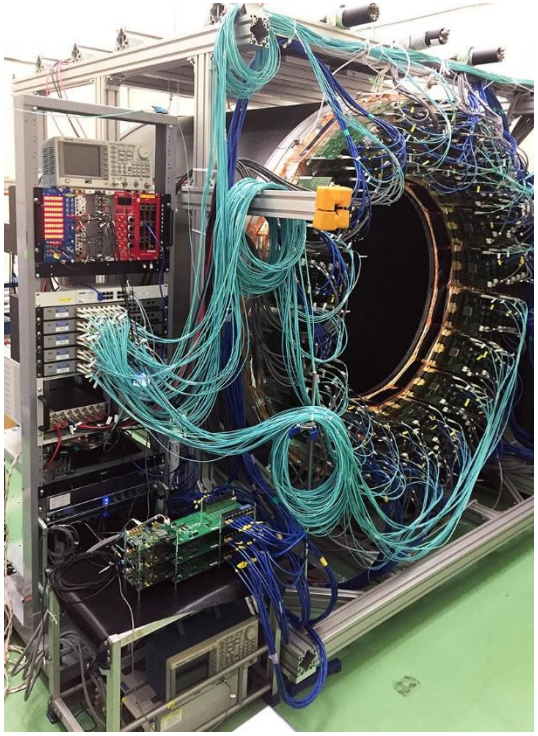
PUB/SUB
→ 制御メッセージ



PUSH/PULL による
Ventilator/Worker/Sink
→ オンライントリガー

読みたい検出器 COMET CDC



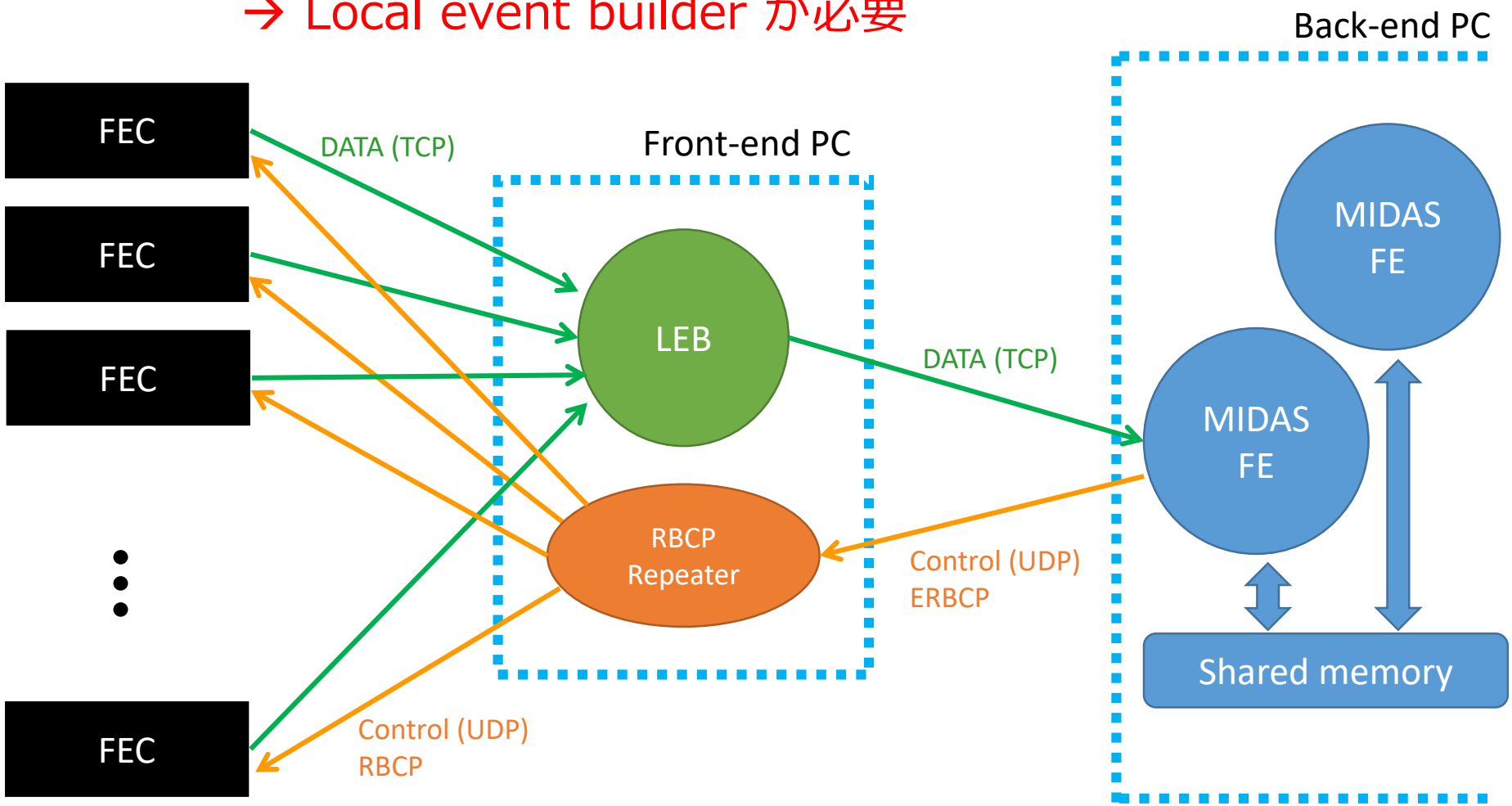


105 RECBEs send their event fragment to a DAQ PC without the BUSY control.

COMET CDC の読み出し

- 104 (+1) の SiTCP を持った Frontend cards
- 本実験時には他の検出器と統合しないといけない。

→ Local event builder が必要



Local event-builder

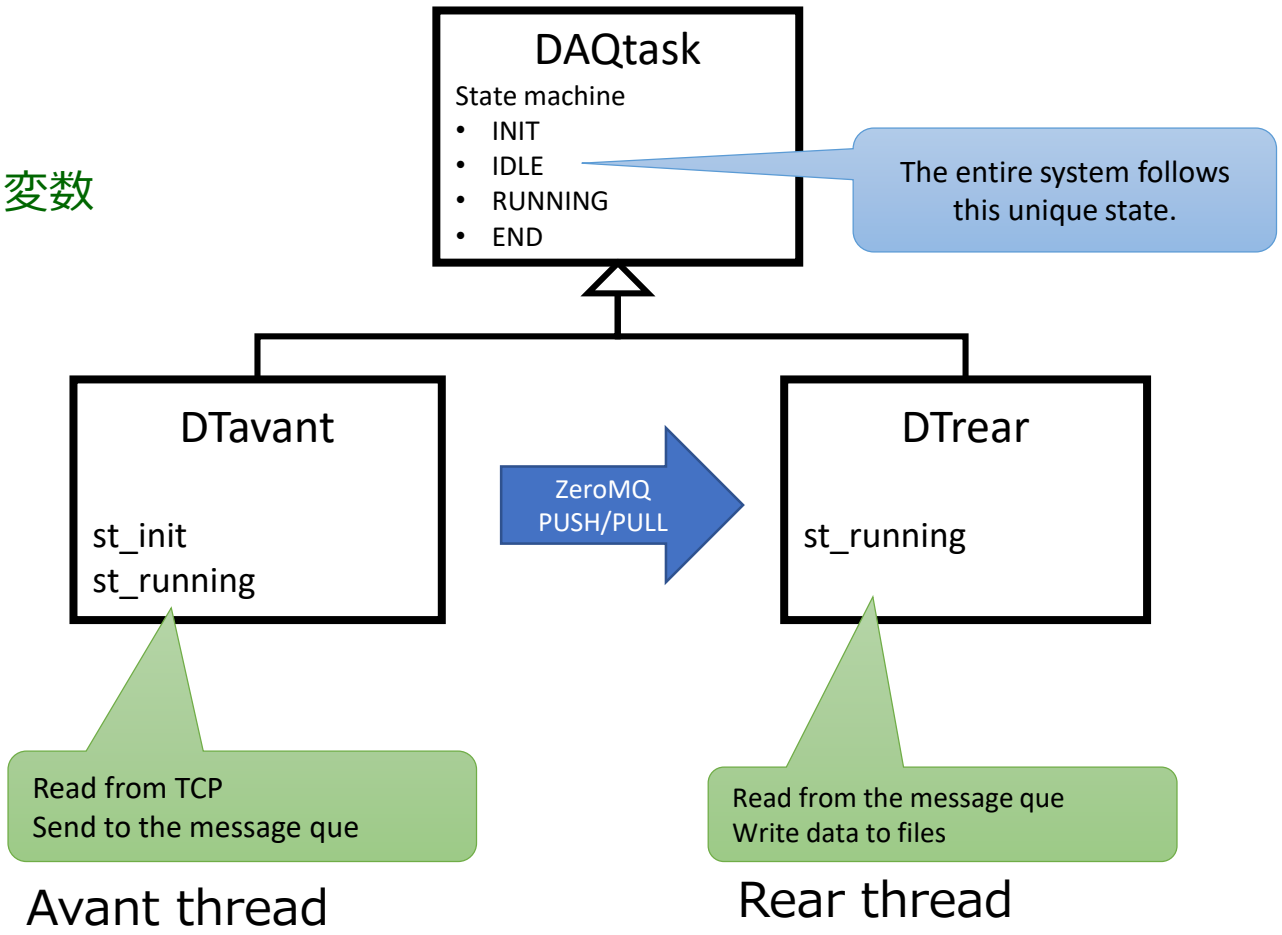
- 不完全なイベントビルドを許容する。
  - Event 番号は Trigger 番号としてイベントフラグメント中にある。この番号でイベントビルド。
  - イベントビルドが成功したら、それ以前のイベントは不完全なイベントとして後段に送る。
  - イベント番号をタグにしてデータ管理
  - Time out を検討中 (時間は記録されている。)

- Status
  - Event builder としては大体動くようになった。
  - RECBE 制御の仕組みを実験中 → RBCP repeater

Local Event-builder Class structure

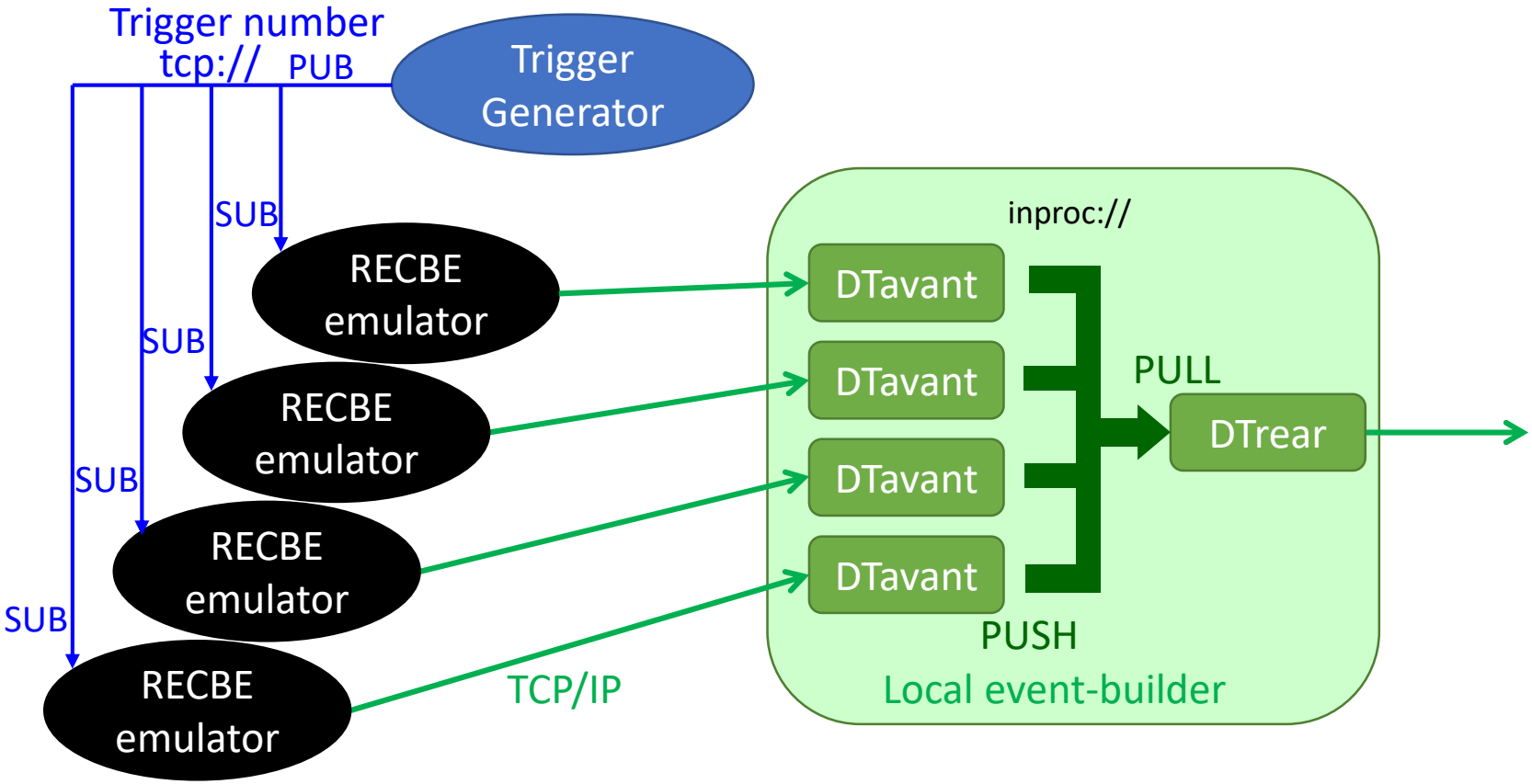
pthread から std::thread (C++11) へ

- Thread 間通信
  - 静的メンバー
    - std::mutex
    - std::atomic
    - 静的メンバー変数
  - ZeroMQ



Software debugging environment

- RECBE emulator を使って Local event-builder を開発
  - RECBE emulator を同期して動かすために、PUB/SUB model を利用
- Local event-builder では Thread 間通信に PUSH/PULL による平衡キューイングを利用
  - Thread 間では IPC ではなくて Thread 間通信である inproc 通信が出来る。



ZeroMQ なんかうまく動きそう

Trigger-less / Streaming DAQ をしたい。

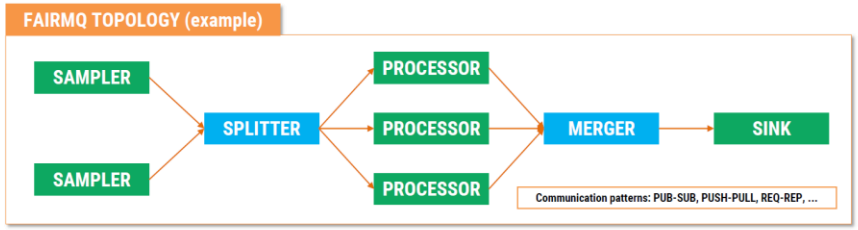
FairMQ

- GSI/FAIR のために開発されている (DAQを含む) フレームワーク
- ZeroMQ + State machine + 制御 plug-in + たくさんの周辺の統合

What is FairMQ?

Organize processing tasks in topologies, consisting of independent processes (Devices), that communicate via *asynchronous message queues* over network or inter-process.

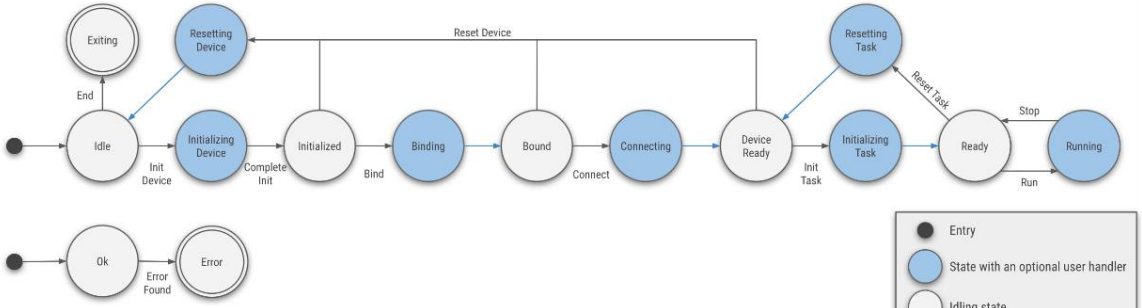
Ethernet, InfiniBand (IP-over-IB)



Ready to use devices are provided for typical scenarios.
User-defined devices can be implemented by inheriting from `FairMQDevice`.

Alexey Rybalchenko | ALICE Offline Week | 01.07.2015 3/21

from Alexey Rybalchenko(GSI)'s slide

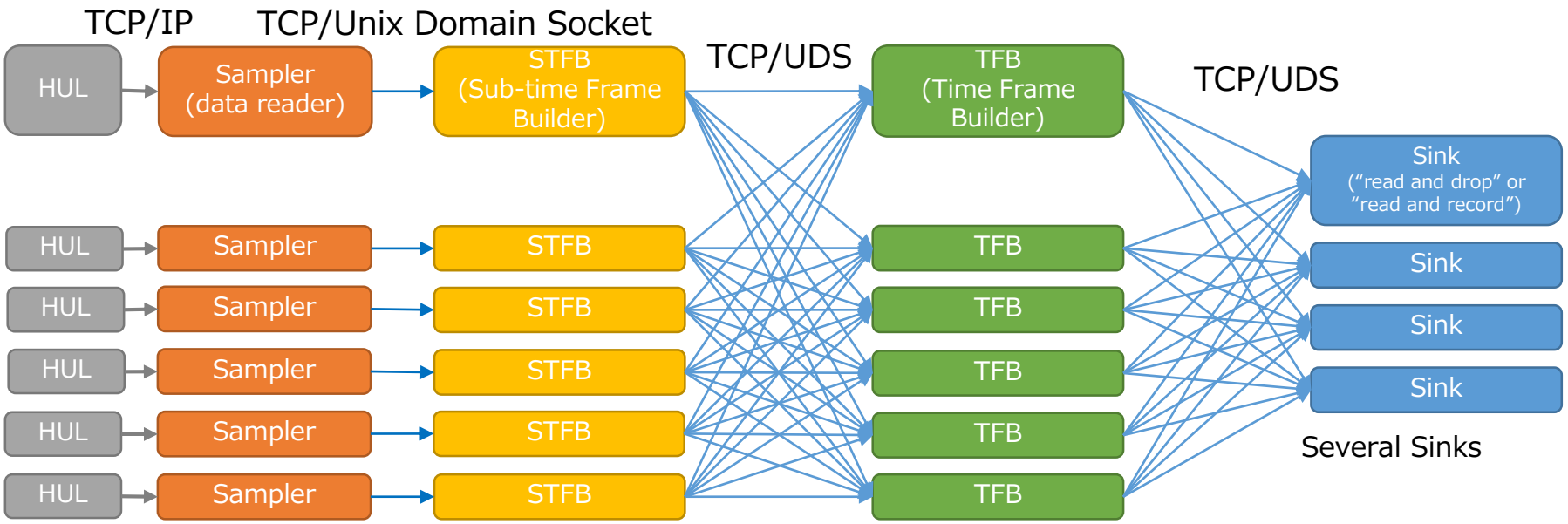


FiarMQ state machine

検出器 Beam 試験における試行

- E50 Fiber tracker と Cherenckov counter の試験を ELPH で 2018 年に行った。
- FairMQ のコア部分を使用
  - HUL TDC の連続読み出し
  - 時間分割による Time frame building
  - ソフトウェアによるコインシデンストリガー
  - 独自制御 plug-in
- “PUSH/PULL” 通信を使用
  - 一応動作
  - うまく動いているときはメッセージがなくなることはなかった。

PTEP に論文が出ます。



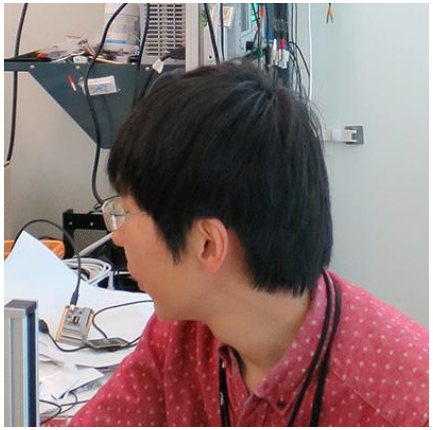
Full mesh connection and round robin network

J-PARC HD での Streaming DAQ

- 検出器試験でうまくいきそうな実感
  → 多数のプロセスを管理する方法があれば…

- E16/E50 をモデルケースにして開発が進んでいる。
  - Pipeline front-end readout (HUL/AMANEQ + …)
  - FairMQ のコア部分を利用
  - 全体管理、制御には redis を使用
    - NoSQL データベース/キーバリュー型
    - メモリ指向/高速
    - キー・スペース通知 → 制御に使える
      - Pub/Sub Channel による制御

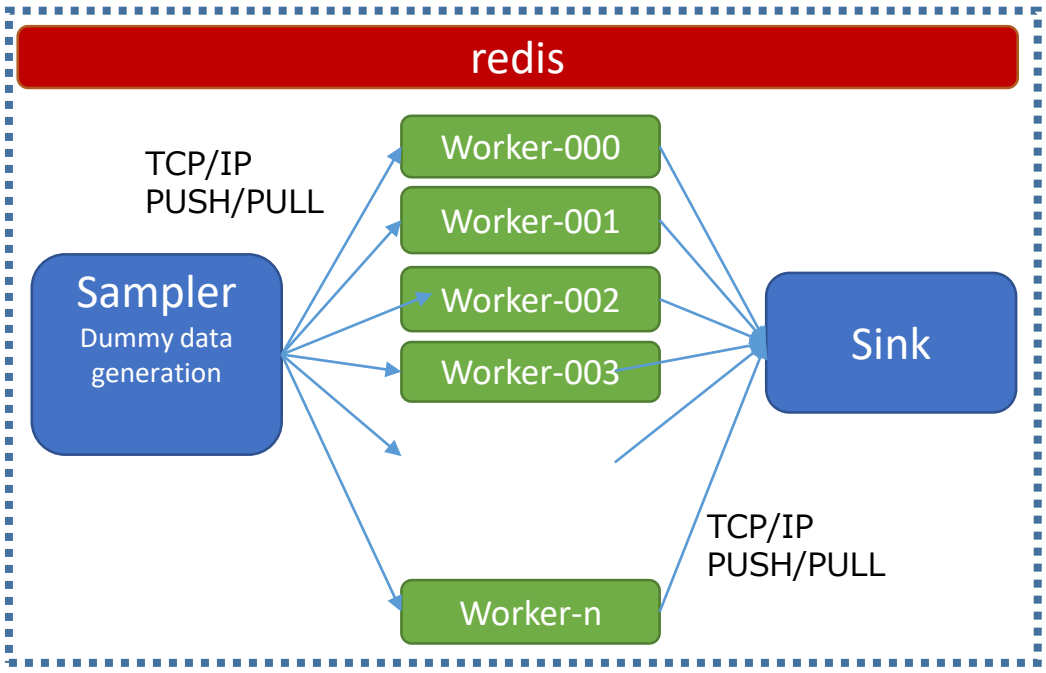
- Redis API を FairMQ 制御 Pulg-in に組み込んで redis からデータを読んだり、制御されたり。
  - Control
  - Status monitor (Metric)
  - Parameter loading



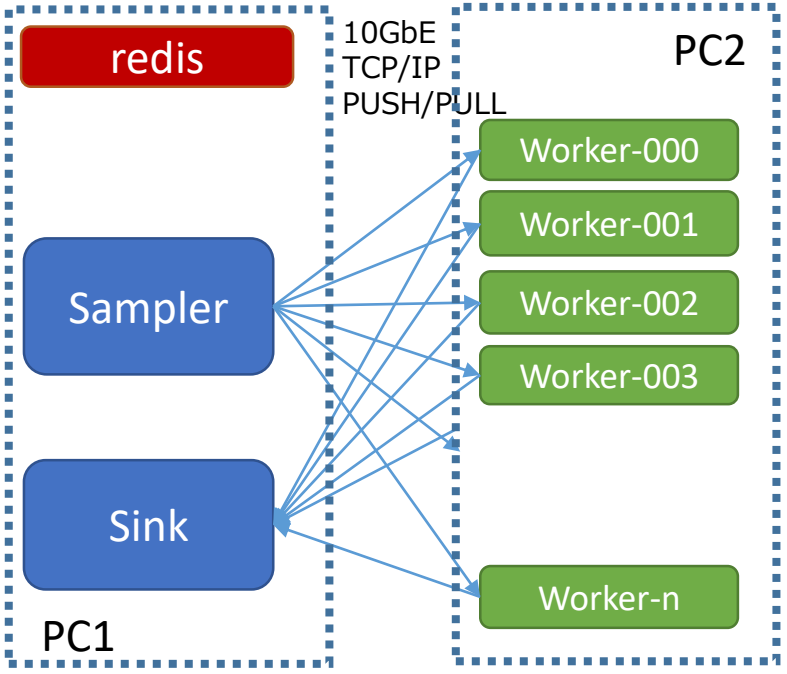
高橋智則氏（理研）が強力に推進中

FairMQ/redis がどこまで動くのか?

2通りの試験環境、2通りの Worker



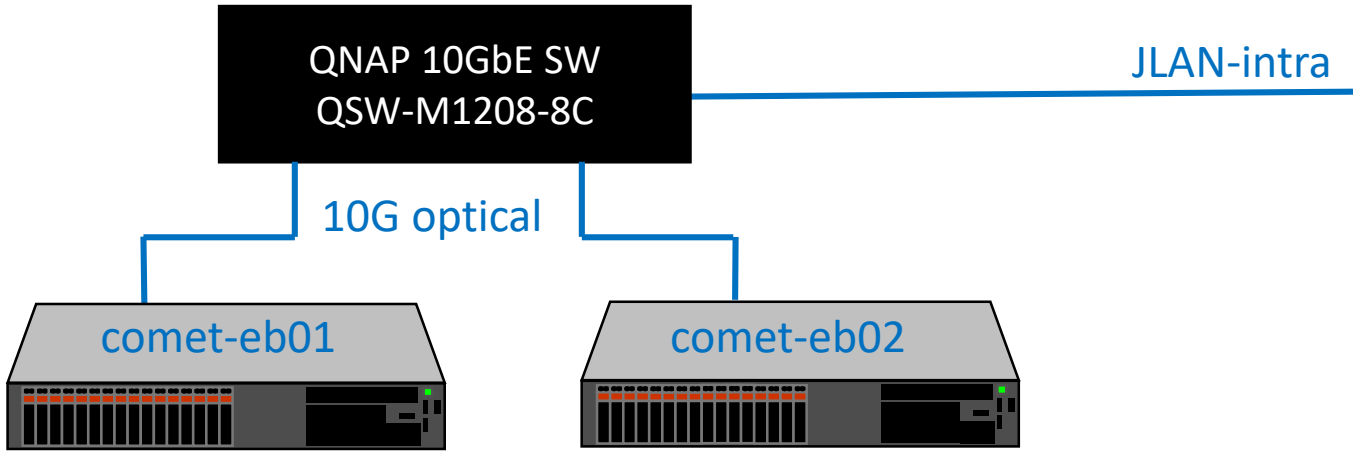
On the PC



via network

- Software trigger farm を模した環境
  - On the PC：すべてのプロセスが一台のPCで動いている。
  - via network：redis, Sampler/Sink PC1, Worker は PC2 で動いている。
- Worker
  - w/o load：何もしないでデータコピー
  - Load：event data 内でデータシャッフルを 1ms の間行い続ける。

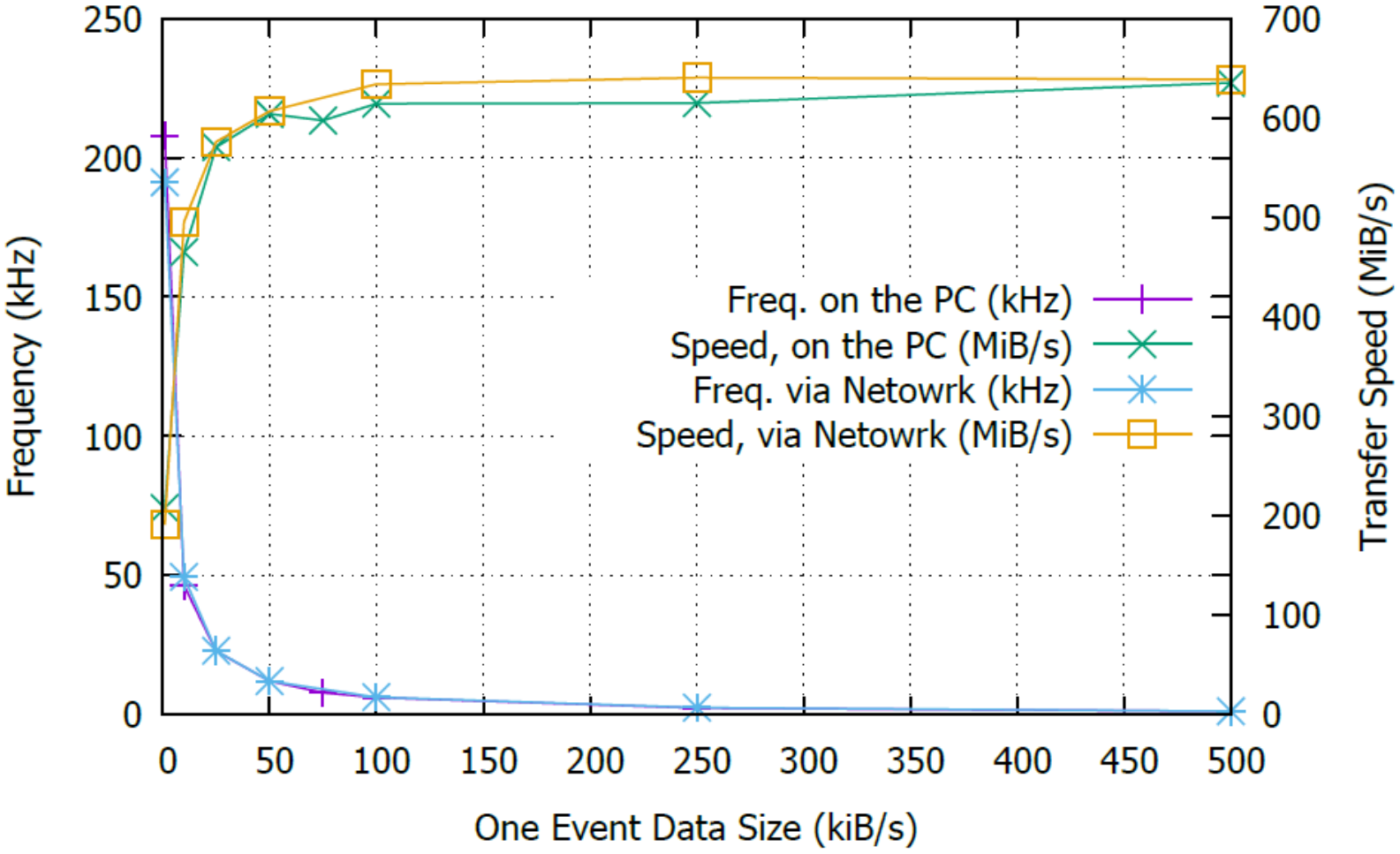
試験環境 機器



- Dell PowerEdge R740
  - Xeon Gold 6126 @ 2.6GHz
  - 24 Cores
  - Intel C620
  - Memory 64GB
  - NIC : Intel X710-DA4
- OS
  - Scientific Linux 7.9
- QNAP 10GbE network switch
  - QSW-M1208-8C

1-1-1 data transfer w/o load

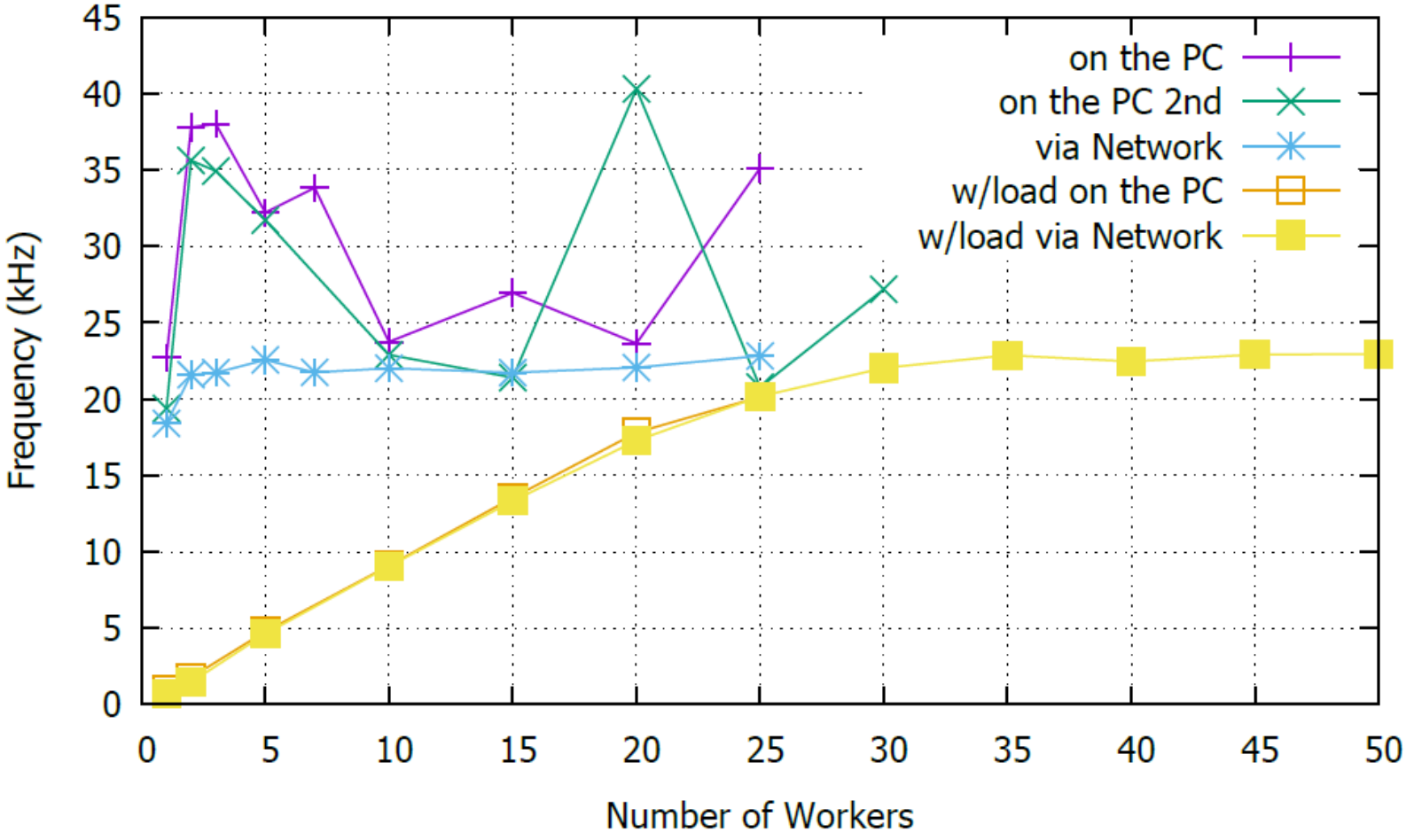
どのくらいの速度でデータの受け渡しが出来るのか?



1列の処理だと 620MiB/s くらいまで。→ それなりの over head

1-n-1 data transfer

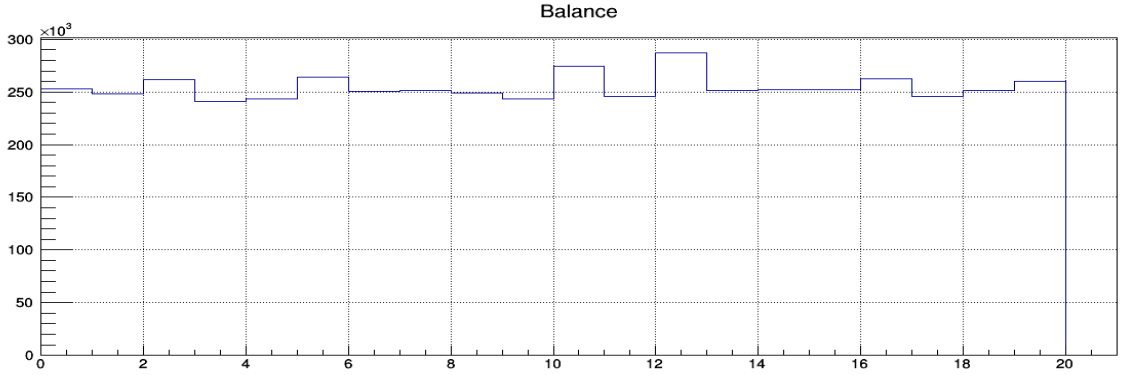
- Worker の数を変えて測定
- Event size は 50 kiB

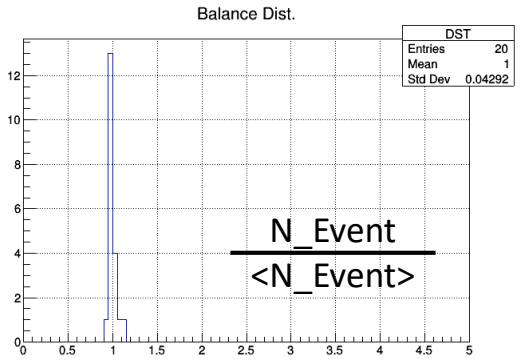


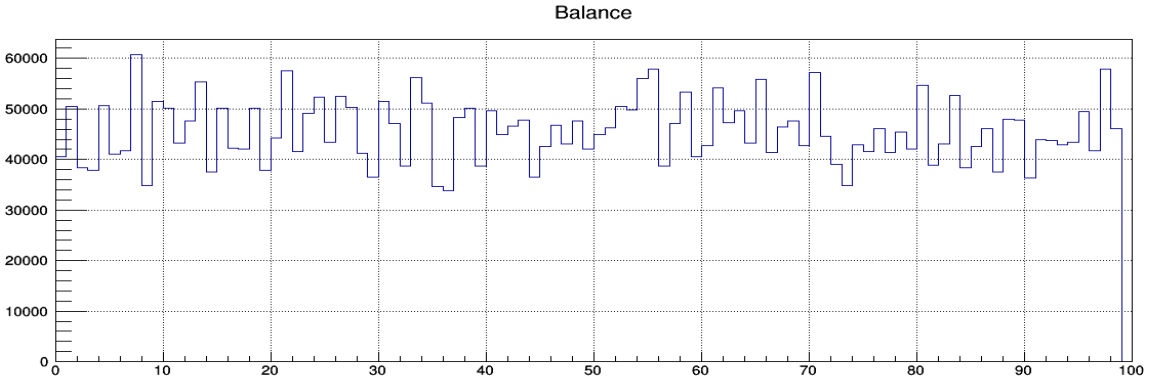
十分な数の Worker Processor があれば 10GbE でつなぐことで、一つの Sampler あたり 1150MiB/s くらいまでソフトウェアトリガー処理が可能

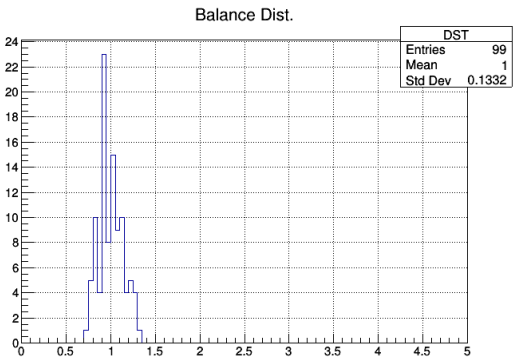
Load balancing by ZeroMQ round robin

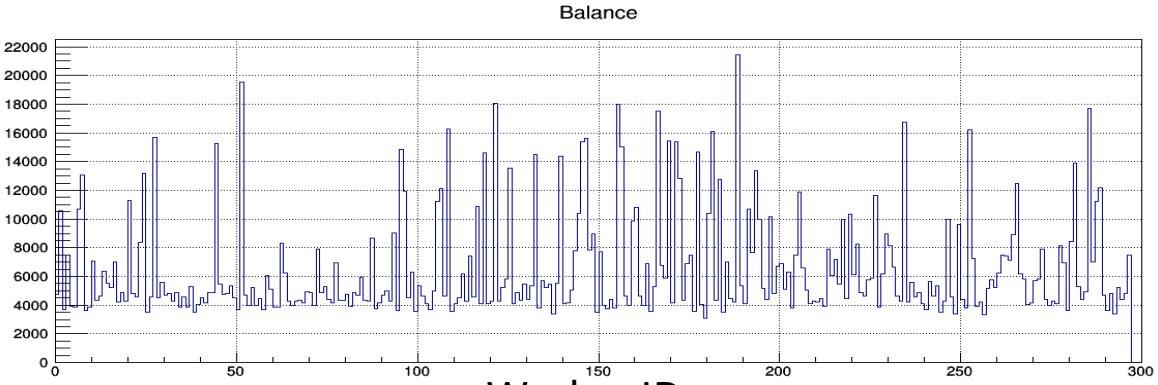
w/o load, via network

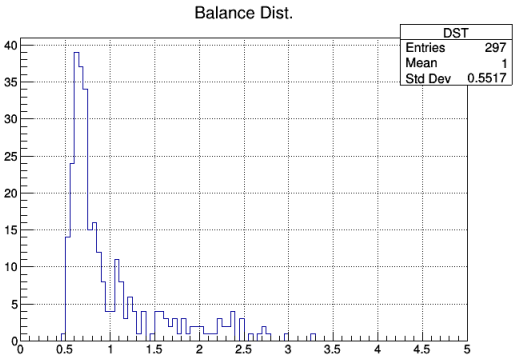










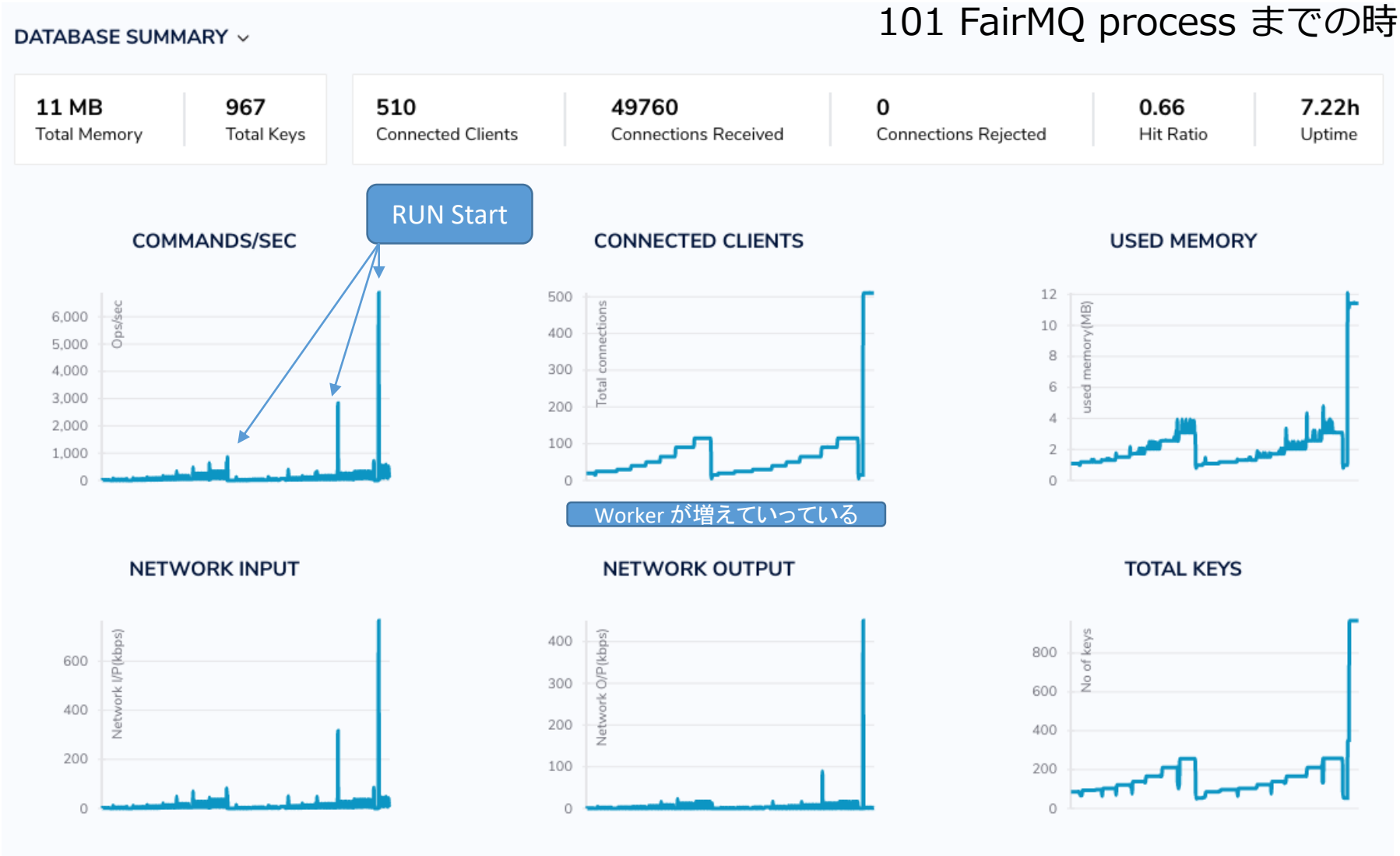


Number of the processed event

Worker ID

redis transaction

101 FairMQ process までの時



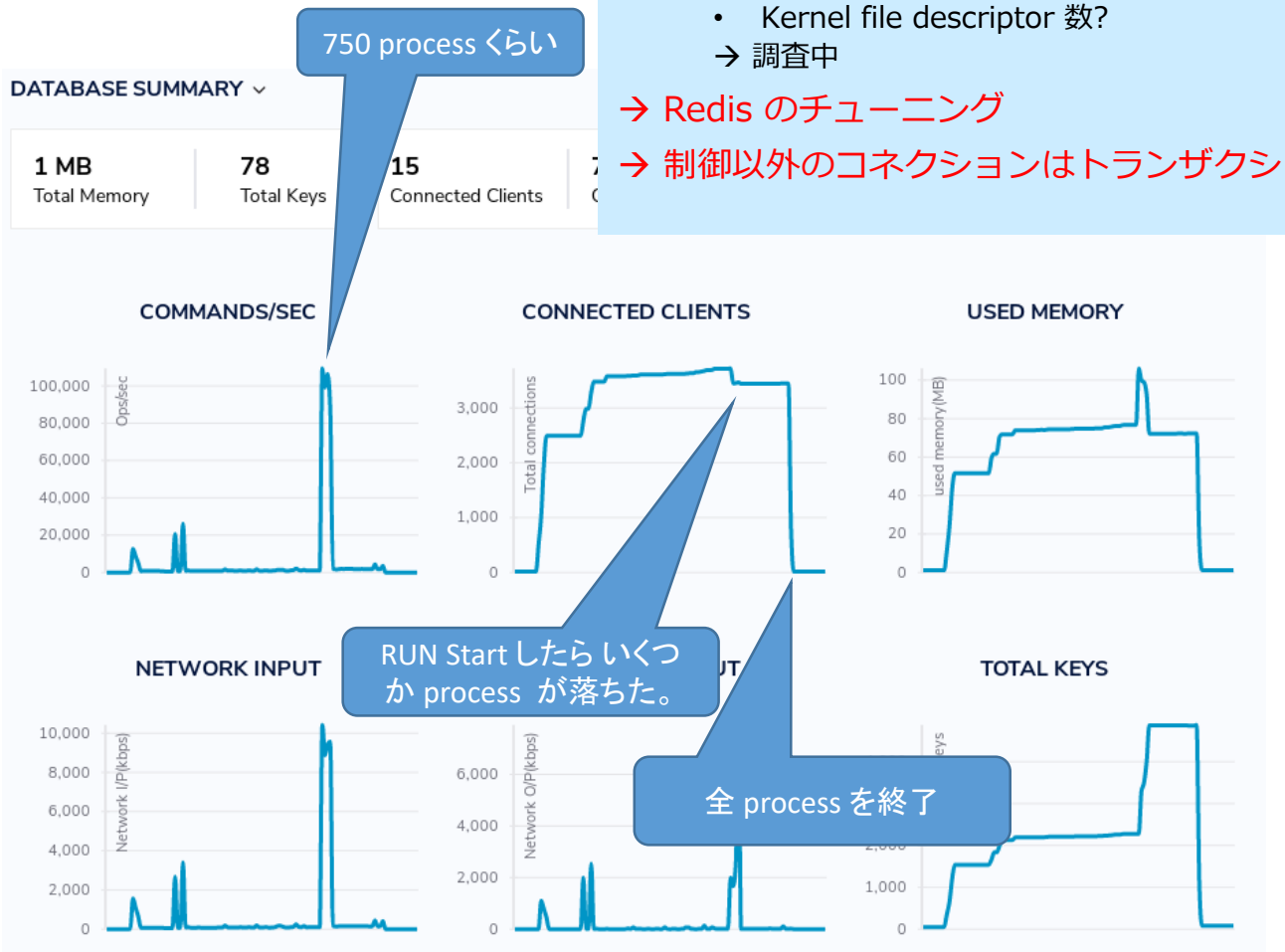
redis の default setting での限界

- 700 FairMQ devices くらいまでは動いた。
- それ以上になると Redis から接続を打ち切られてプロセスが落ちる。
- いろいろな制限
  - redis client 数?
  - redis ファイルディスクリプタ数?
  - Kernel thread 数?
  - Kernel file descriptor 数?

  → 調査中

→ Redis のチューニング

→ 制御以外のコネクションはトランザクション時以外は Close



まとめ と その他

- Network based DAQ Software は ZeroMQ を使うとだいぶ楽に信頼性の高いものが作れる。
- FairMQ/redis で Trigger-less / Streaming system はうまくいきそうだ。
  - 700 process くらいまではいい感じで動いている。
    - redis のチューニングとコネクションの工夫で 1000 以上いけそう。
    - 数百 process では思っていたよりは負荷が軽い。”top” command に現れるようなことはあまりない。
    - Load balancing もほどほどにやってくれる。

- FairMQ/redis の課題
  - バージョン依存性、ポータビリティ の解決。
  - ドキュメンテーション
  - UI
    - 暫定的なものはある
    - Grafana で Status のモニタは出来ている。
  - 問題の洗い出し。実地で経験を積む必要がある。
    - VPS で大量の計算機を短時間借りることを検討している。

  - 求む! 共同研究者