いわゆる一つの Streaming DAQ の実装

五十嵐 洋一
KEK
計測システム研究会 2022.11.17

Motivation

- Hardware trigger が困難に
  - Software trigger / data reduction をしたい。
  - Trigger less DAQ をしたい。
- データ量の増大
  - 最後がシングルノードだとそこがボトルネックになる。
    - 最後をマルチノードにしたい。
    - メッシュ状の接続構成
- TCP/IP よりロバストで便利なものはないかな?
  - 通信制御を楽にしたい。
  - Point to Point だけでなく 1 to N, N to 1 も
- バッファ管理を楽かつロバストにしたい。
  - DAQ が落ちる最多原因。
    - 昔はメモリーが十分になかったので静的なメモリ確保と管理が主流だった。
  - 最近はメモリーがたくさん使える。
    → Dynamic memory allocation
    → Container class
- 少人数で運用可能であることは外せない。

Beyond the socket programming

- DAQ のほぼ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 は通信のスタンダード
  - 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

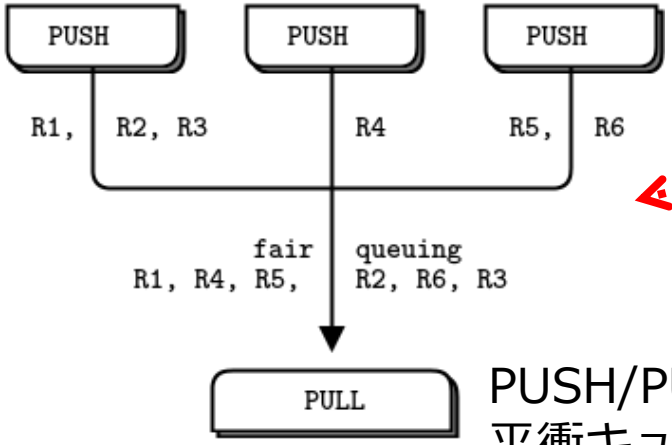
手法は確立されているが、手法を守らないとトラブる。

なにか新しくこなれたものはないかな? → ZeroMQ

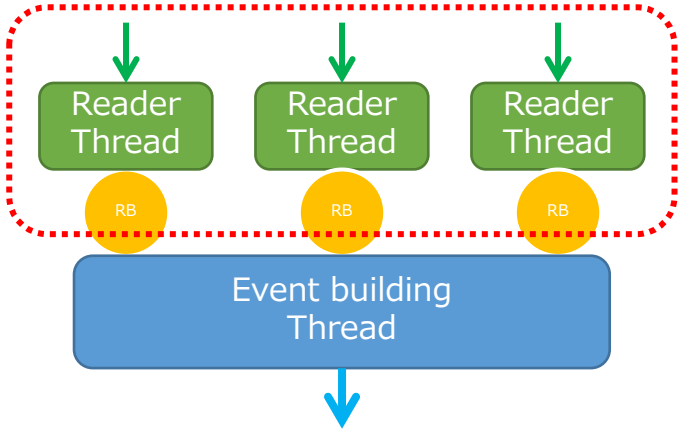
ZeroMQ

- 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?)
  - MPI よりも分散DAQに向いている気がする?

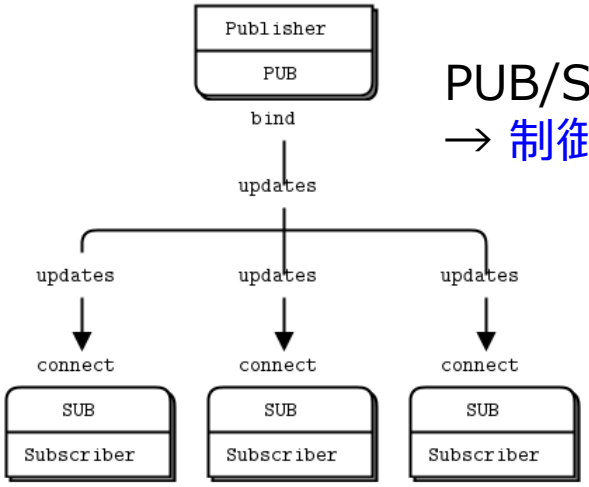
ZeroMQ 通信モデルの一部



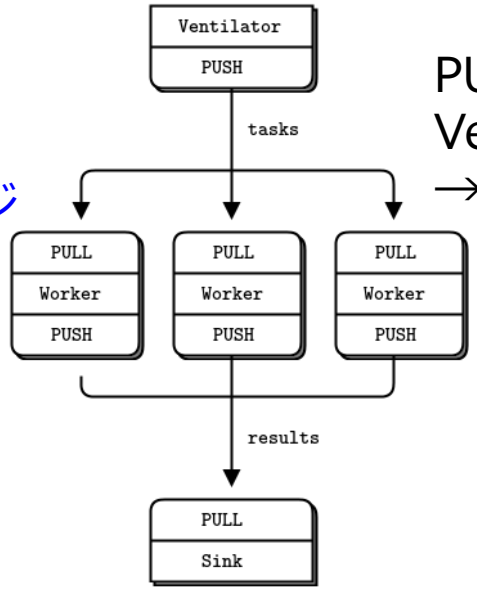
PUSH/PULL による
平衡キューイング



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

他に、もちろん Exclusive PAIR もある。

分散 DAQ を構成するには

- 通信
  - ZeroMQ
- 有限状態遷移機械
  - INITIALZE, IDLE, RUN, POSE, STOP
- 制御・被制御
- 状態把握、Watchdog

どうする?

- ハードウェアからのデータの取り込み
  - Socket programing, read(2), device driver

→ 先行研究に何かあった。→ FairMQ

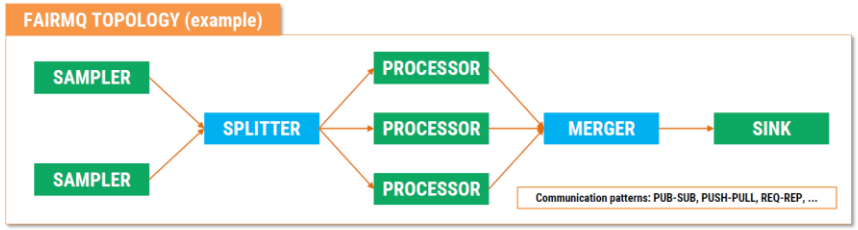
FairMQ

- GSI/FAIR のために開発されている (DAQや解析、シミュレーションを含む) 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.

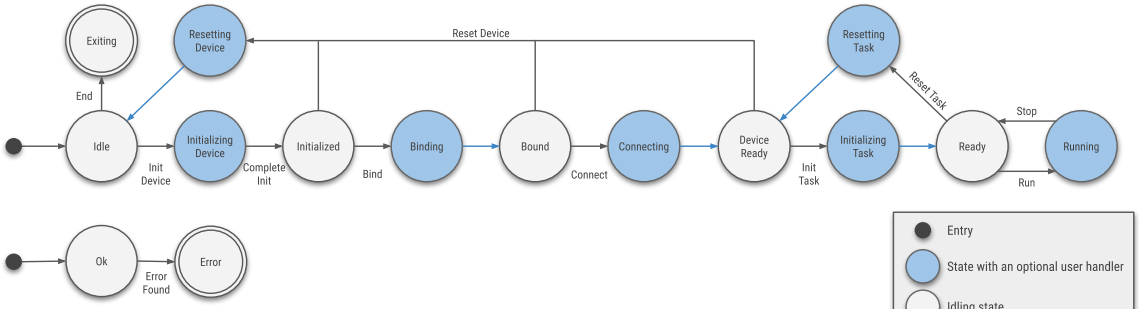
Ethernet, InfiniBand (IP-over-IB)



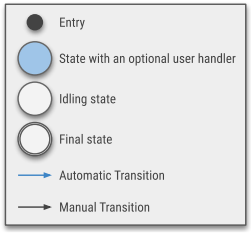
Ready to use devices are provided for typical scenarios.
User-defined devices can be implemented by inheriting from `FairMQDevice`.

Alexey Rybalchenko | ALICE Offline Week | 01.07.2015 3/21

from Alexey Rybalchenko(GSI)'s slide



FiarMQ state machine

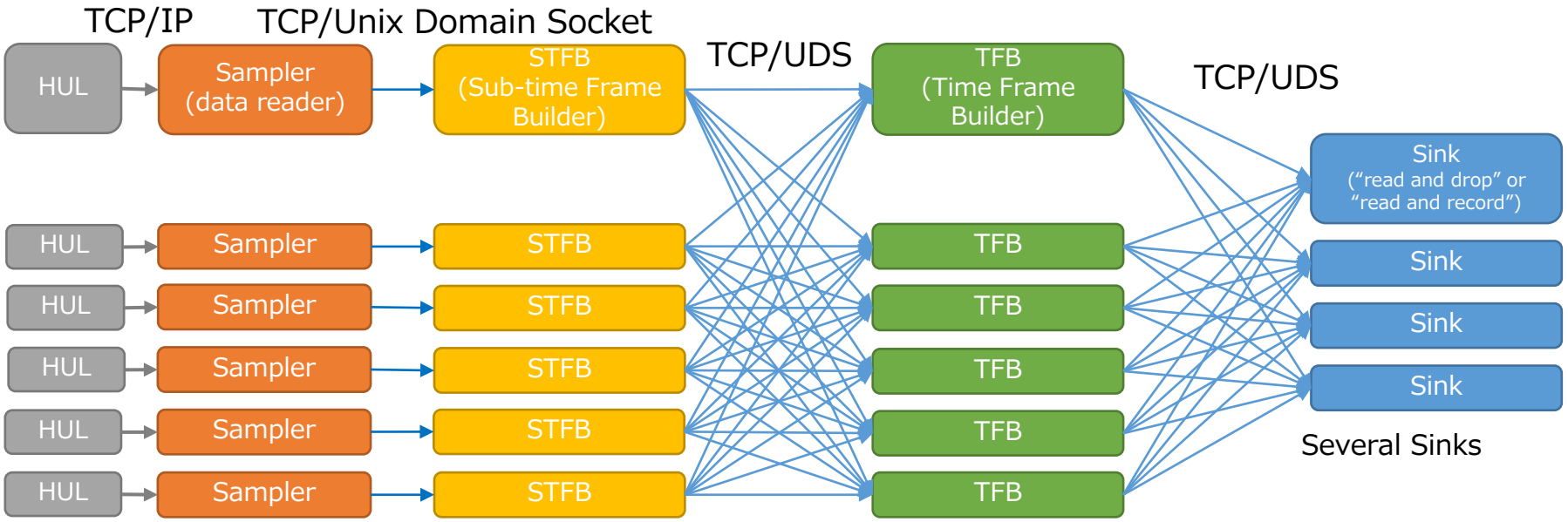


Plugin による拡張が可能

検出器Beam試験におけるFairMQの試行

- 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, PAIR” 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。

R. Honda et al. “Continuous timing measurement using a data-streaming DAQ system”, PTEP https://doi.org/10.1093/ptep/ptab128。



Full mesh connection and round robin network

J-PARC HD での Streaming DAQ

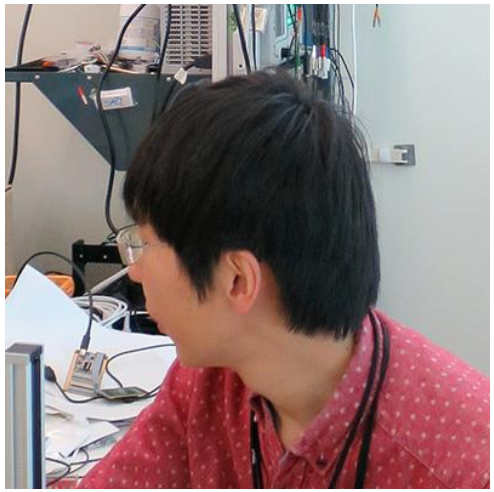
- 検出器試験でうまくいきそうな実感
  → 多数のプロセスを管理する方法があれば…
  → Key-Value 型データベース redis

→ FairMQ と redis を組み合わせたらどうだろう。

FairMQ + radis → NestDAQ (Network based Streaming DAQ)

- 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 の状態遷移機械、被制御部分を利用
  - 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

- 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



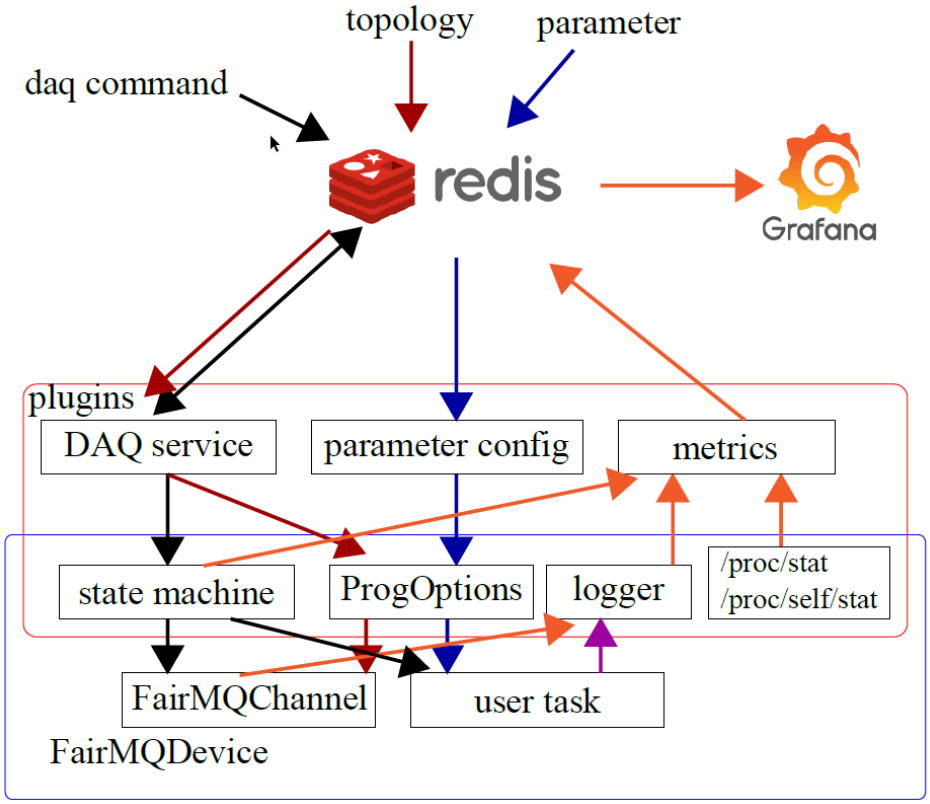
高橋智則氏（理研）が強力に推進中

たくさんのものを管理する

NoSQL/Key-Value 型データベース redis ですべてのプロセスの状態管理と制御をおこなう。

FairMQ Plugin

- DAQ Service Plugin
  - Run control
    - State machine の制御
    - Run number の設定
  - Service discovery
    - 接続設定の半自動化
- Metrics Plugin
  - Process の状態把握
- Parameter config Plugin
  - ハードウェア初期化等のパラメータをデータベースから読み込む



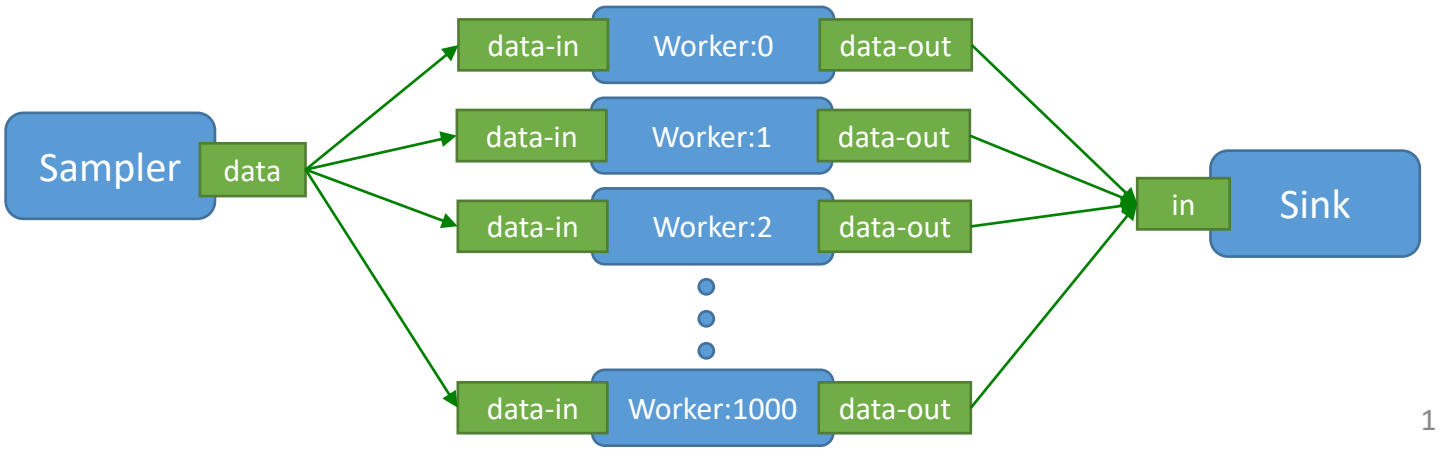
たくさんのものを構成する

DAQ Service：Service discovery

- 1000以上のテーブルは手書きは現実的でない。→ 構成の半自動化
- 機能を論理グループとして分けてヒントを与えて接続情報を自動的に構成する。
  - Interface, Service, Endpoint, Link

```
#-------------------------------------------------------------------------
#            service        channel         options
#-------------------------------------------------------------------------
endpoint     Sampler         data             type push  method bind
endpoint     Sink            in               type pull  method bind
endpoint     Worker          data-in          type pull  method connect
endpoint     Worker          data-out         type push  method connect

#-------------------------------------------------------------------------
#       service1         channel1        service2     channel2
#-------------------------------------------------------------------------
link     Sampler          data            Worker       data-in
link     Worker           data-out        Sink         in
```



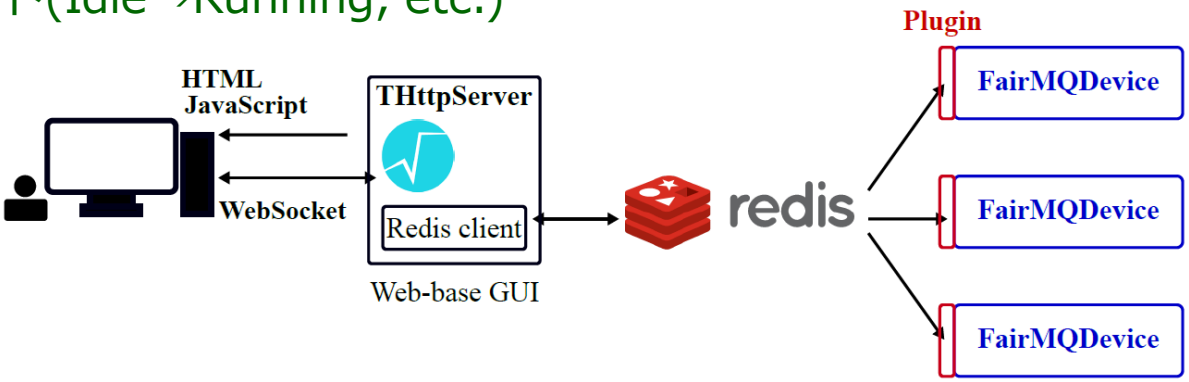
たくさんのものを制御する

RUN 制御

- DAQ Service Plugin
  - redis channel を利用
    - redis のメッセージ機能
      - Client にメッセージを送ることが出来る。
    - PUB(Publisher)/SUB(Subscriber) channel
      - 1対他の通信
  - redis interface をもつものであればコマンドを発行できる。
    - 制御 UI が分離されている。Webベース、redis-cli、python script

E16 RUN0 での制御 UI の実装

- Web-based GUI
  - HTML + JavaScript + WebSocket + boost::beast
  - RUN number の設定, 更新
  - 状態遷移コマンド(Idle→Running, etc.)



Sample UI screen

保護されていない通信 | 35.160.228.58:8080

DAQ controller

RUN number

New value: 3 Send +1 Get
Latest : 3

State transition command

Idle ▹ Running

Idle ▹ Init Device and Connection Device Ready ▹ Init Task ▹ Ready ▹ Run ▹ Running

Idle ◃ Running

Idle ◃ Reset Device ◃ Device Ready ◃ Reset Task ◃ Ready ◃ Stop ◃ Running

▹ Exit

Any state ▹ End ▹ Exiting

State Summary

Get Reset

Service	N	Undefined	Ok	Error	Idle	Init-Device	Initialized	Binding	Bound	Connecting	Device-Ready	Init-Task	Ready	Running	Reset-Task	Reset-Device	Exiting	last-update
dtdcSampler	1													1				2022-11-16 09:59:35
dtdcSink	1													1				2022-11-16 09:59:34
dtdcWorker	2													2				2022-11-16 09:59:35
fairmq-merger	1													1				2022-11-16 09:59:34

Service	Instance	state	last-update

Select command target

Choose Services: all, dtdcSampler, dtdcSink, dtdcWorker, fairmq-merger

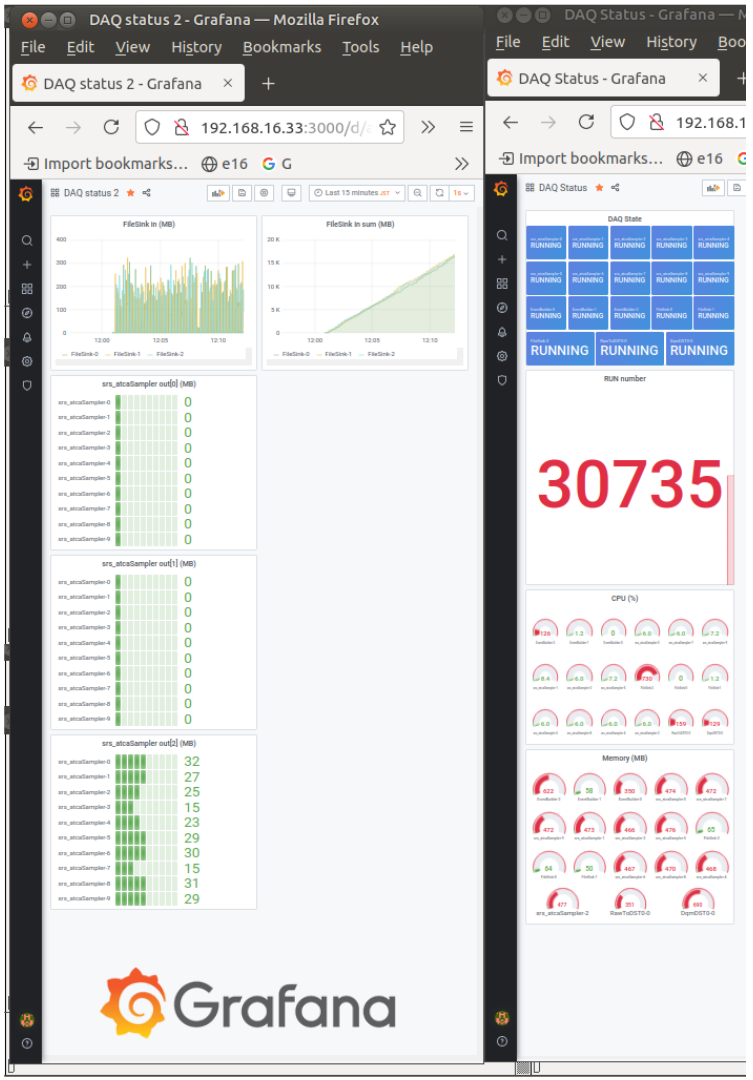
Choose Instances: all, dtdcSampler:dtdcSampler-0, dtdcSink:dtdcSink-0, dtdcWorker:dtdcWorker-0, dtdcWorker:dtdcWorker-1, fairmq-merger:fairmq-merger-0

My WebSocket Connection ID: 1 (Date: 2022-11-16 09:48:58)

WebSocket Connected ID: Date
1 : 2022-11-16 09:48:58

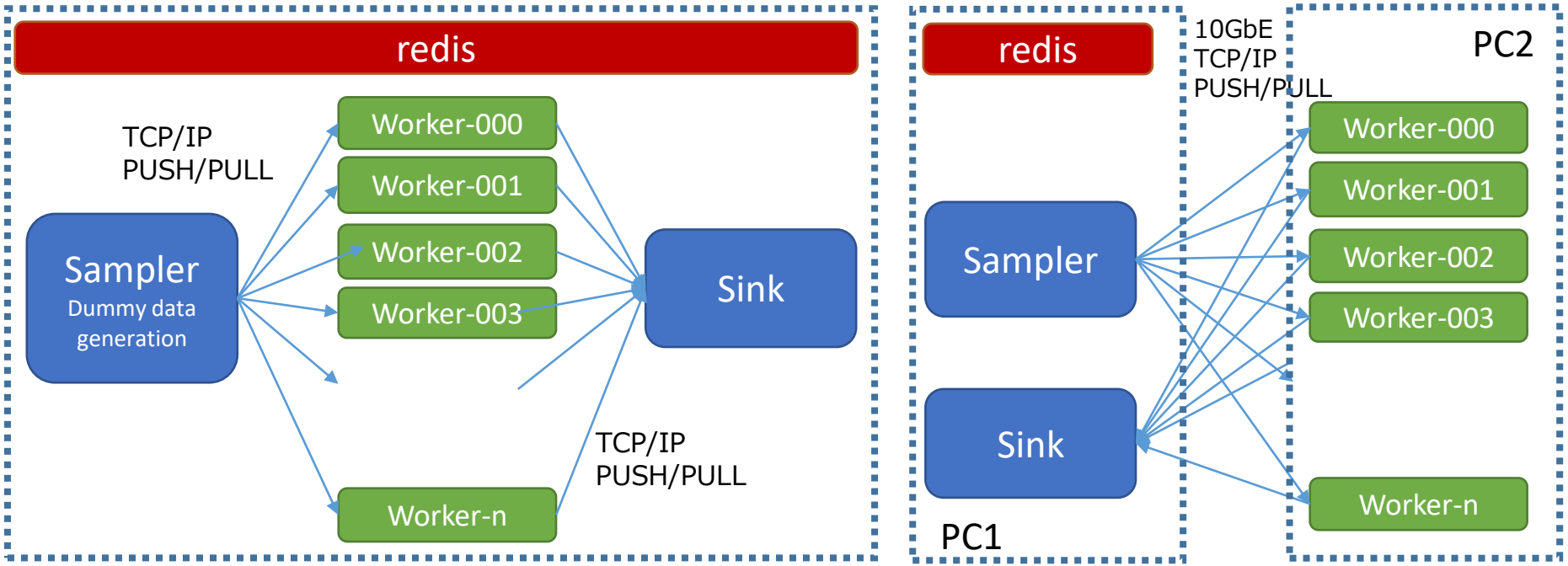
Messages

Web controller



Metrics Plugin と Grafana による
ステータス、負荷表示

FairMQ/redis がどこまで動くのか?



Software trigger farm を模した環境で試験

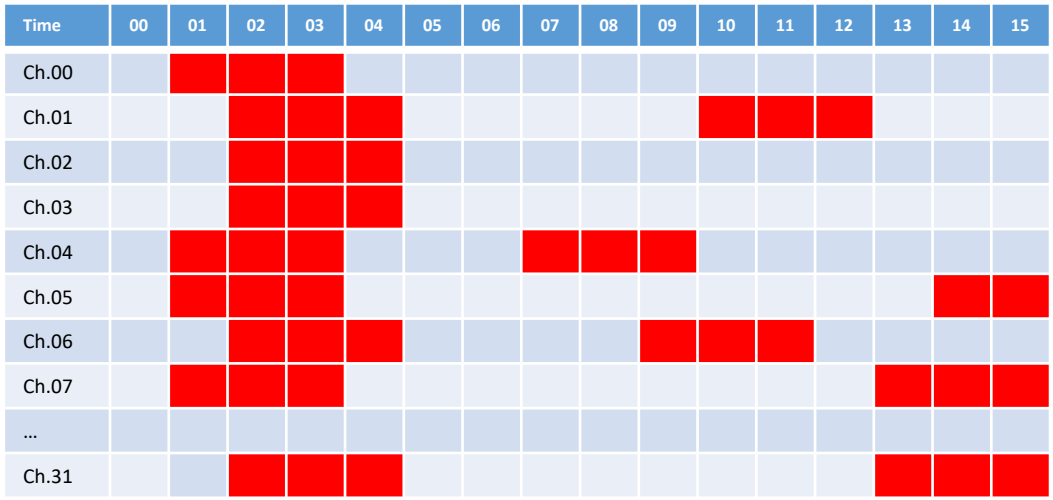
- 10 Gbps network で複数のPCがつながれている
- 十分な Worker プロセスがあれば 1150MB/s くらいで処理が可能
- 実績として 700 以上のプロセスを管理
- Round robin, Queue full だけの制御でもほどほどのロードバランスが得られる。

Coincidence trigger をやってみる

- TDC dummy data generator
  - Timeframe 20bit x unit
  - ch0 に random に Hit TDC を詰める。
  - ch1～ch31 に ch0 + random hit を詰める。
  - Coincidence trigger で読んでみる。

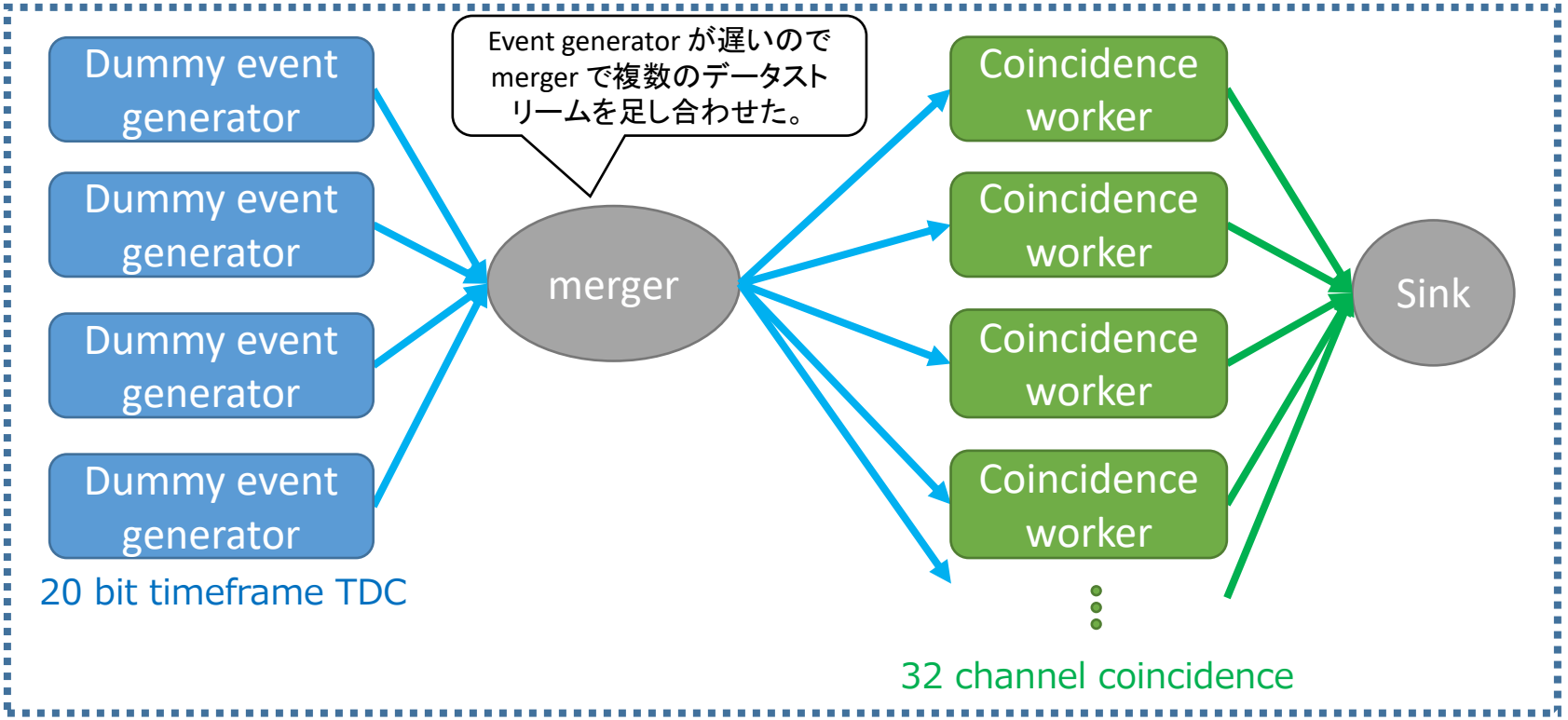
- Coincidence trigger
  - 20bit はあまり大きくないので、メモリに展開してマークを付けていく方法で試行。
  - ch0 と答え合わせ。

0xffff をスキャン



Coincidence Trigger 評価

Configuration



Coincidence 処理に 1 core あたり 2.2 ms (Xeon platinum)

→ 1ns count TDC を仮定すると 32 ch 1ms の期間の Timeframe を処理するのに 2～3 core

→ OR + AND の Trigger 処理に 数～十数 core くらいで行けそう

→ 評価は進行中

NestDAQ インストールと依存関係

- 依存
  - C++ 17

  - Redis 6.0.10
  - Redis TimeSeries 1.4.18
  - Grafana (optional)

  - Boost 1.72.0 or later
  - FairLogger 1.9.0 or later
  - FairMQ 1.4.26 or later
  - Hiredis 1.0.0
  - redis-plus-plus 1.3.3 or later

- コンパイル
  - ひたすら source からコンパイルする。
    - ディストリビューションのものを使うとはまるので避ける。
    - 試験環境：OS CentOS 7.9 + devtoolset-8
      - おそらくは C++17 以外に大きな依存性はないはず…

コミュニティ

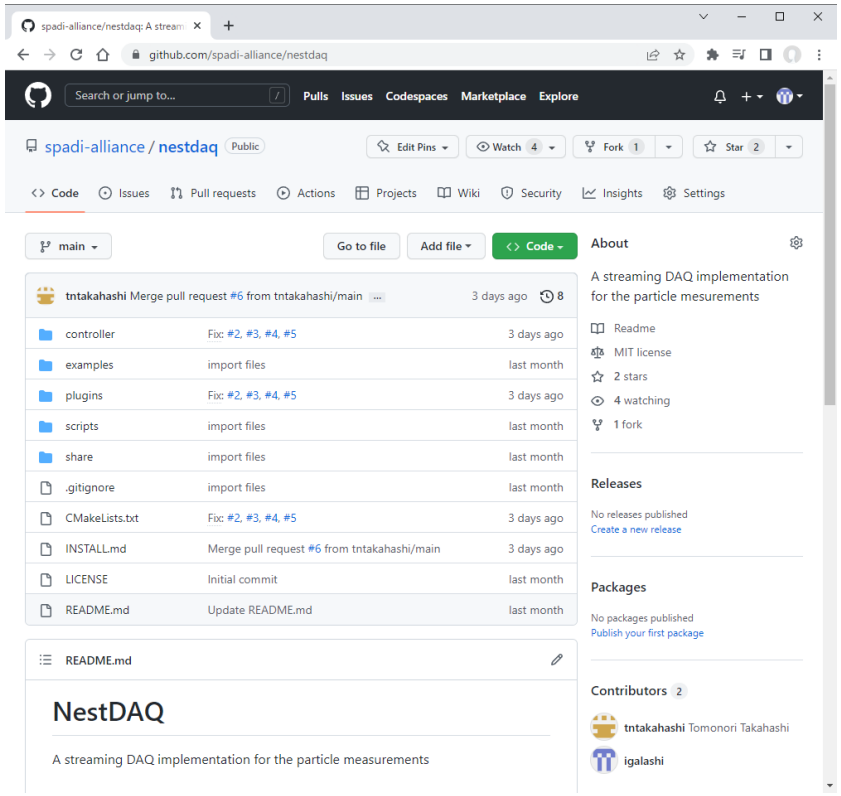
- 公開レポジトリ
  - https://github.com/spadi-alliance/nestdaq
  - まだ先行公開程度、インストールドキュメントのみ
  - ドキュメントは準備しようとしている、ホームページは検討中

SPADI alliance の活動として github に公開している。

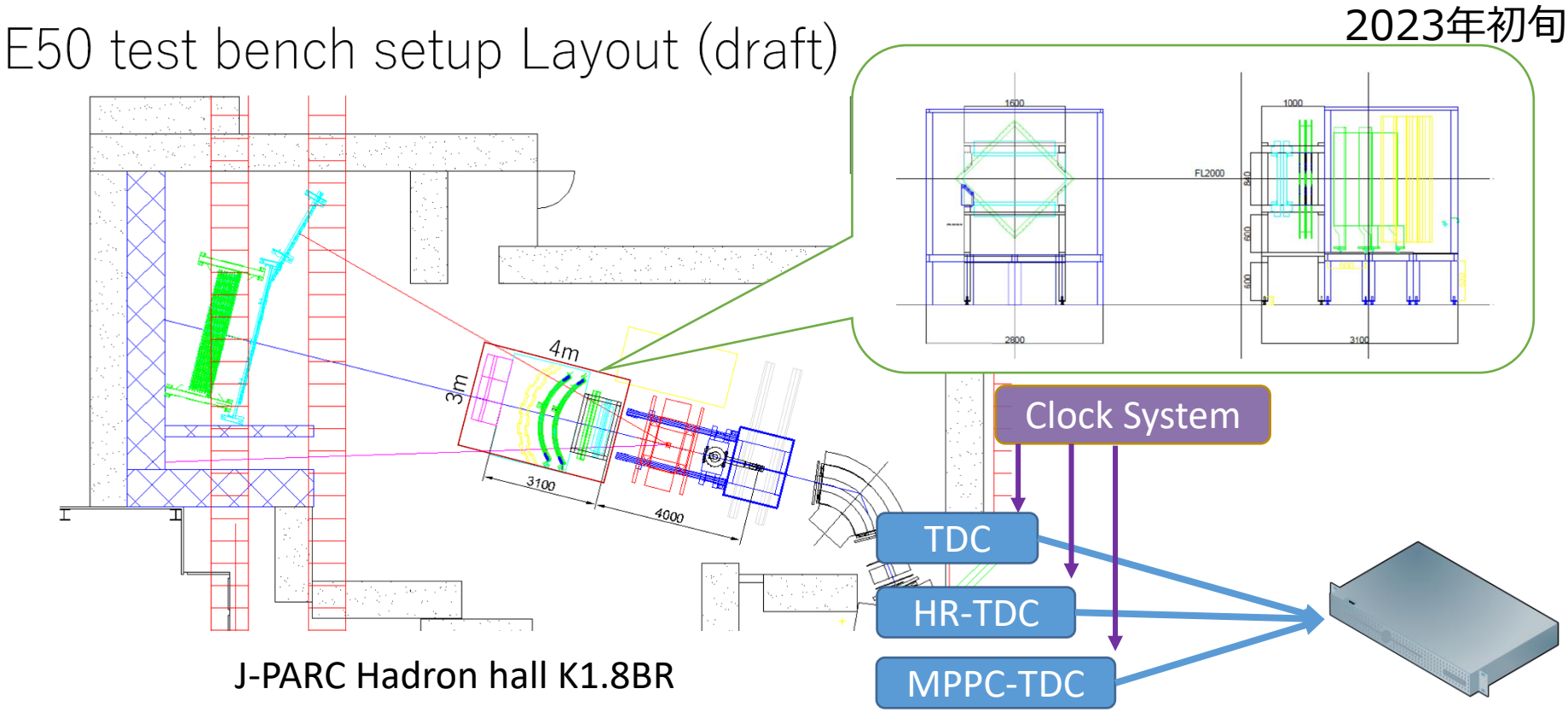
打ち合わせは、月1程度

興味がある方、人柱になっても良い方、協力しても良い方 等は　SPADI alliance に登録するか、私まで連絡ください。

次のマイルストーン
E50 検出器のためのビームテスト 2023 初旬
→ ここでいろいろ確立したい。



E50 検出器試験



Streaming DAQ の確立

- 連続 TDC (TDC, HR-TDC, MPPC)の読み出し
- Clock 配布システム (MIKUMARI)
- Software coincidence trigger

まとめ

- トリガーレスデータ収集に対応したストリーム型 DAQ software を開発中
- ZeroMQ/FairMQ + redis → NestDAQ
  - ゆっくり開発進行中
  - そろそろと動き出している。
  - Coincidence を取るだけならそこまで重くない感じ。

協力者募集中

関連

SPADI Alliance : https://www.rcnp.osaka-u.ac.jp/~spadi/

SPADI Alliance Mattermost : https://www.rcnp.osaka-u.ac.jp/mattermost/login?redirect_to=%2Fspadi-alliance%2Fchannels%2Ftown-square