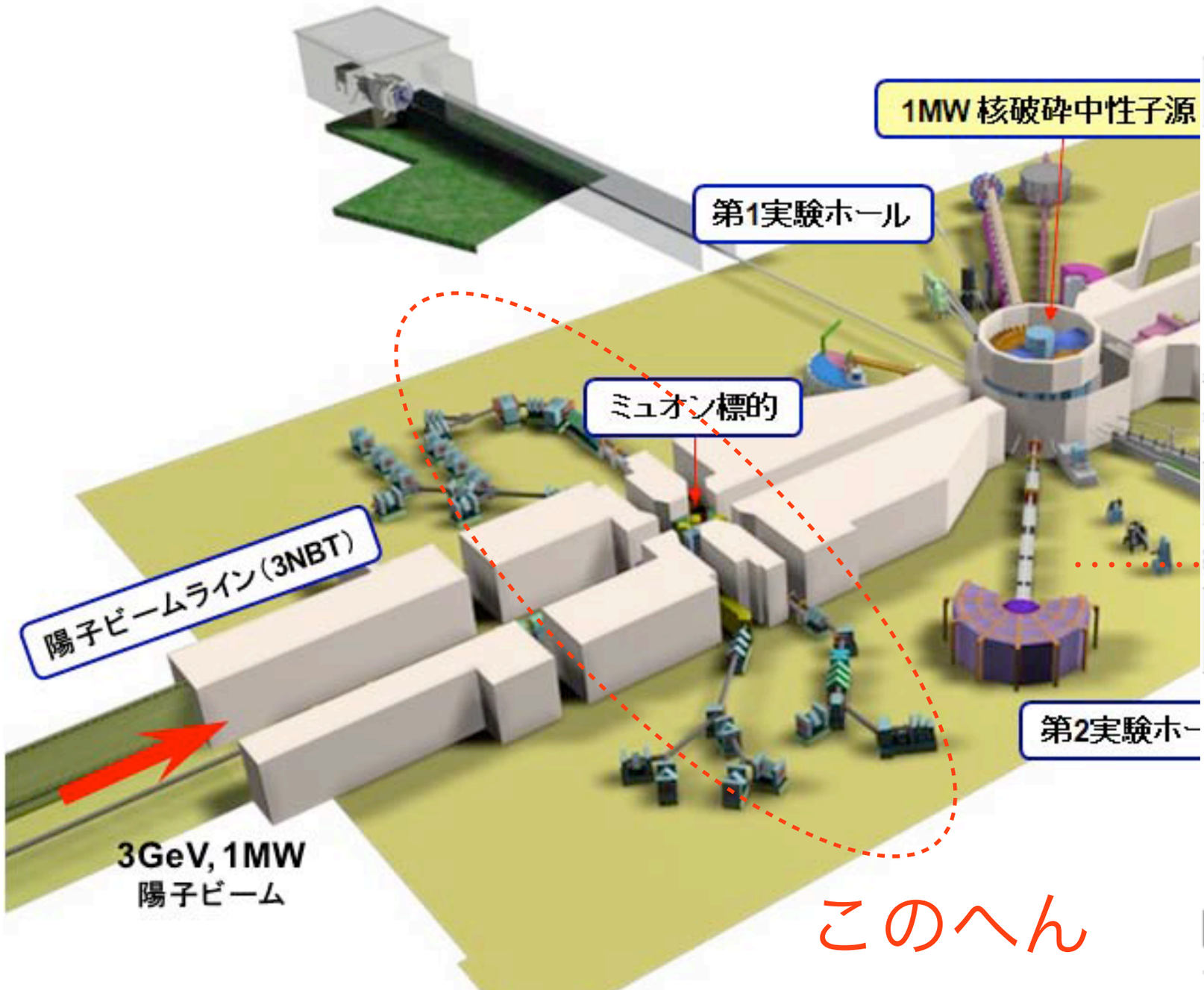
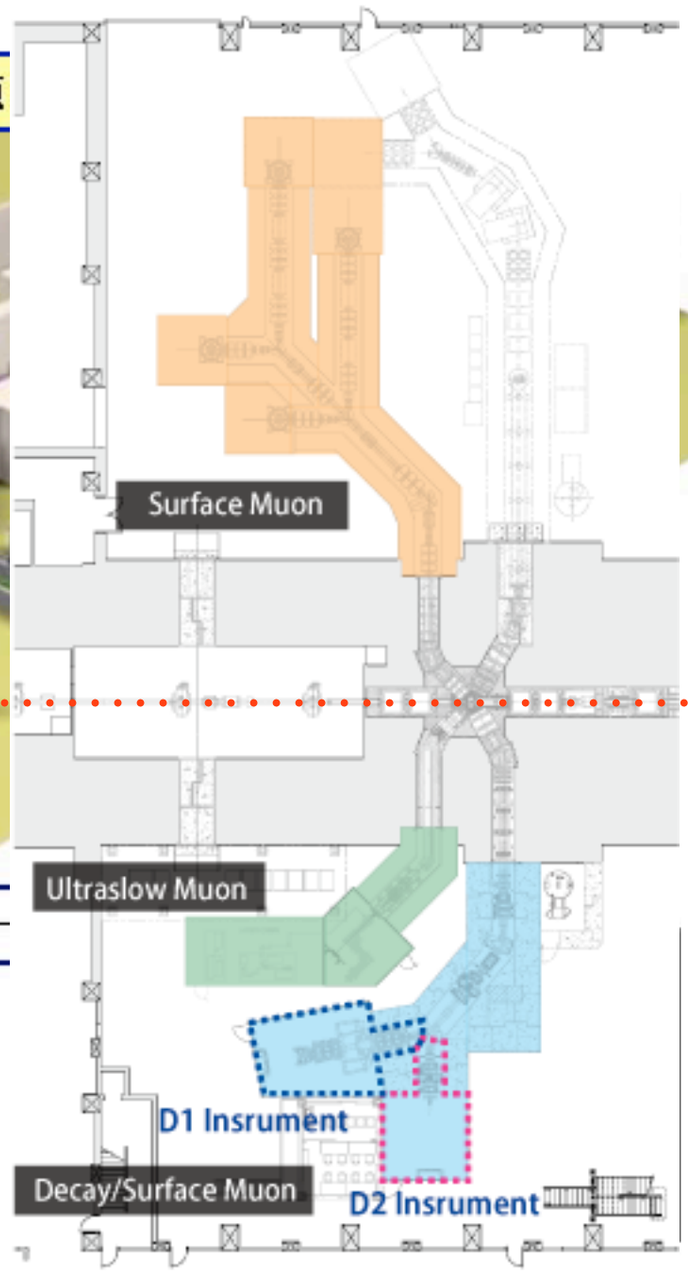
MUSEのMuSR用DAQ
ソフトウェアと今後の課題

J-PARC MUSEのみなさん と
鈴木聡 (KEK 計算科学センター)

MUSEの場所





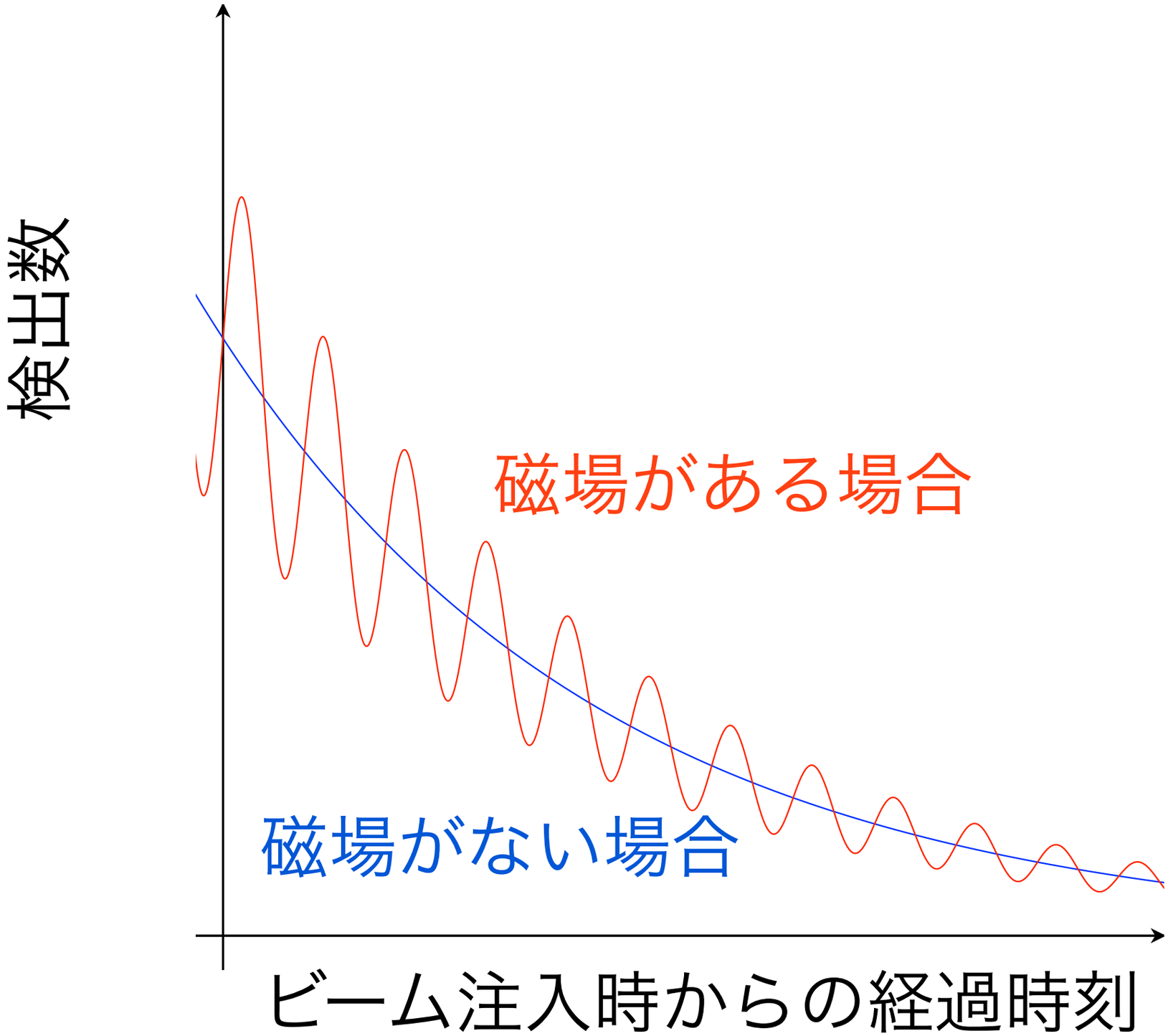
このへん

KEKならば物構研のジャンルの建物

MuonSpinRotation実験

- 試料にミュオンビームを当てると物質中でμが止まる(一部)
- 平均寿命2.2μ秒でμはeに崩壊するが、スピンの方向にeが出やすい
- μ^+をあてればe^+が出るのでシンチレータで検出する
- 検出されたe^+の偏りから逆にスピンの偏りが検出できる
- スピンが磁場を感じて歳差運動すると、e^+の偏り方向も歳差運動する
  - http://msl.kek.jp/msr/muonintro/index_j.html
  - http://legacy.kek.jp/ja/news/press/2008/J-PARCMLF4.html
- 磁場をかけつつスピンの偏ったビームを注入すると磁場がどのくらいまで侵入しているかわかる
  - 偏らなくなったら磁場が届いてないと考えられる
- 超伝導だと磁場が侵入できなくなるので、超伝導になっているかどうかがわかる。周期から磁場の強さもわかる。
  - 別に超伝導専用の実験手法というわけではございません

特定方向のe^+検出数の変動



いろんな環境パラメータで同じ物質を測定する

- どこまで進入しているかはビームと試料の位置関係を変えないと測定でいない
  - 圧力・温度についてもスキャンしたい
- 環境パラメータのスキャンをやるので、1RUNが短く出来れば嬉しい
- ほとんど一定の条件で統計をためまくる素粒子実験とは性格が異なる

中性子とも異なる

- 中性子はシャッターを開けっ放しで写真を撮る感じ
  - ただし、色(中性子の運動量)の分解能が必要
  - Pulse Hight Analyzerのような動作で十分
- MUSEは連射
  - コマ独立でなくてはいけない
  - さっきのコマと今のコマが相関していない
    - 部分的に欠けたコマを集めても嬉しくない
  - 色(ミュオンがいつ崩壊したか)の分解能ももちろん必要

拾い食いだけではムリ!

- 測定の芸風は中性子に近いが、DAQは高エネに近い
- すでにMUSEは世界最高強度のパルスμビームなので、よその劣化コピー + スケールアップではだめ
  - よそ: Triumf, PSI => DCビーム, RAL ISIS => パルスビーム
- 開発のためだけにビームを出せるわけでもない
  - 身内の実験にいわゆる寄生ビームタイムでとりつく
  - スタッフには供用実験のための運転業務もあるので激しく高負荷

DAQの出来事

2007 2008 2009 2010 2011 2012 2013 2014

RAL beam test

MUSE

AMT-3 FINESSE

FPGA-TDC FINESSE

MPPC test

COPPER

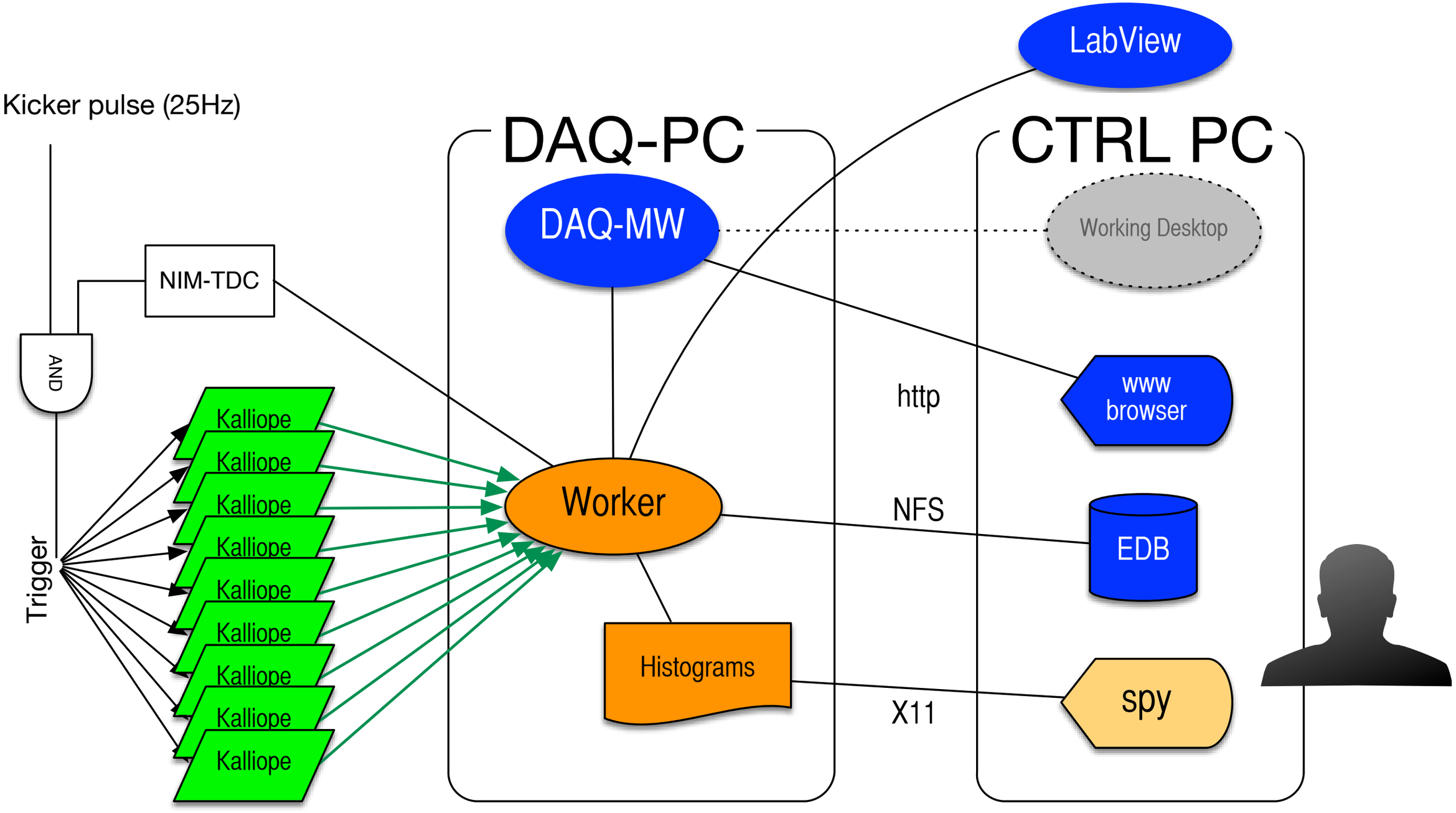
COPPER-Lite

Kalliope

Niki modules

毎年なにか変わる

DAQシステムの構造



ここは明日

DAQ-Middleware

ホーム | 初めての方 | DAQ-Middlewareユーザの方

DAQミドルウエアホームページ

Contents

- DAQミドルウエアとは
- 概要
- ソフトウエアパッケージ
- マニュアル
- トレーニングコース(講習会)
- DAQ-Middlewareユーザ紹介
- FAQ
- サポート
- 発表・論文
- 研究会
- イベント
- メンバー

DAQ-Middleware

ネットワーク分散環境でデータ収集用ソフトウェアを容易に構築するためのソフトウェア・フレームワーク DAQ-Middlewareのサイトです。

DAQ-Middlewareパッケージの最新版

DAQ-Middleware 1.3.0を利用できます。

What's new

[2014-08-04] 2014年度DAQ-Middlewareトレーニングコースを2014年9月3日～5日の日程で開催します。
[2013-11-30] DAQ-Middleware 1.3.0をリリースしました。
[2013-09-05] 産総研オープンラボのRT-Middleware関連展示でDAQ-Middlewareの展示を行います。
[2013-08-29] 2013年9月3日、RT-Middleware関連講演会でDAQ-Middlewareの話をします。
[2013-08-28] 2013年度DAQ-Middlewareトレーニングコース(@広島工業大学)を2013年9月10日(火)～11日の日程で開催します。

MLFの中性子の標準DAQフレームワーク

IROHAは試料の条件(位置・温度・圧力など)

を変更してRUN開始・停止をDAQMWに指示

DAQMW文化との結合

- MuSRはLabViewと結合したいが、その時点ではIROHAが未完成
  - 完成してもLabViewとの結合は自作必須
  - IROHAではどのくらいビームを浴びせたかで切り替えは可能だが、どのくらいヒットが貯まったら切り替えろ、が出来ない等
- が、DAQMWはIROHA以外からのステート変更は非公認
  - STOPといってないのにSTOPしているとはどういうことかね?プンプン
  - 本来ならばIROHA互換の命令を発行するような素敵プログラムを開発するべきだが、時間がない
  - DAQMWとDAQプログラムを分離し、疎結合にする
- サンプルコンポーネント"Skeleton"を利用してほぼ何もしないDAQMWのコンポーネント"Alone"を用意
  - GathererやLoggerなどのコンポーネントは一切存在しない
- DAQMWのStart/Stop/Pause/Resumeのハンドラは指令を"start RUN番号"などの単純な文字列としてWorkerにTCPで投げつける
  - DAQMWのサンプルWWWインターフェースもstart/stopの失敗などは全く確認していないし...
  - ヒットの溜まり具合は別途テキストに記録する

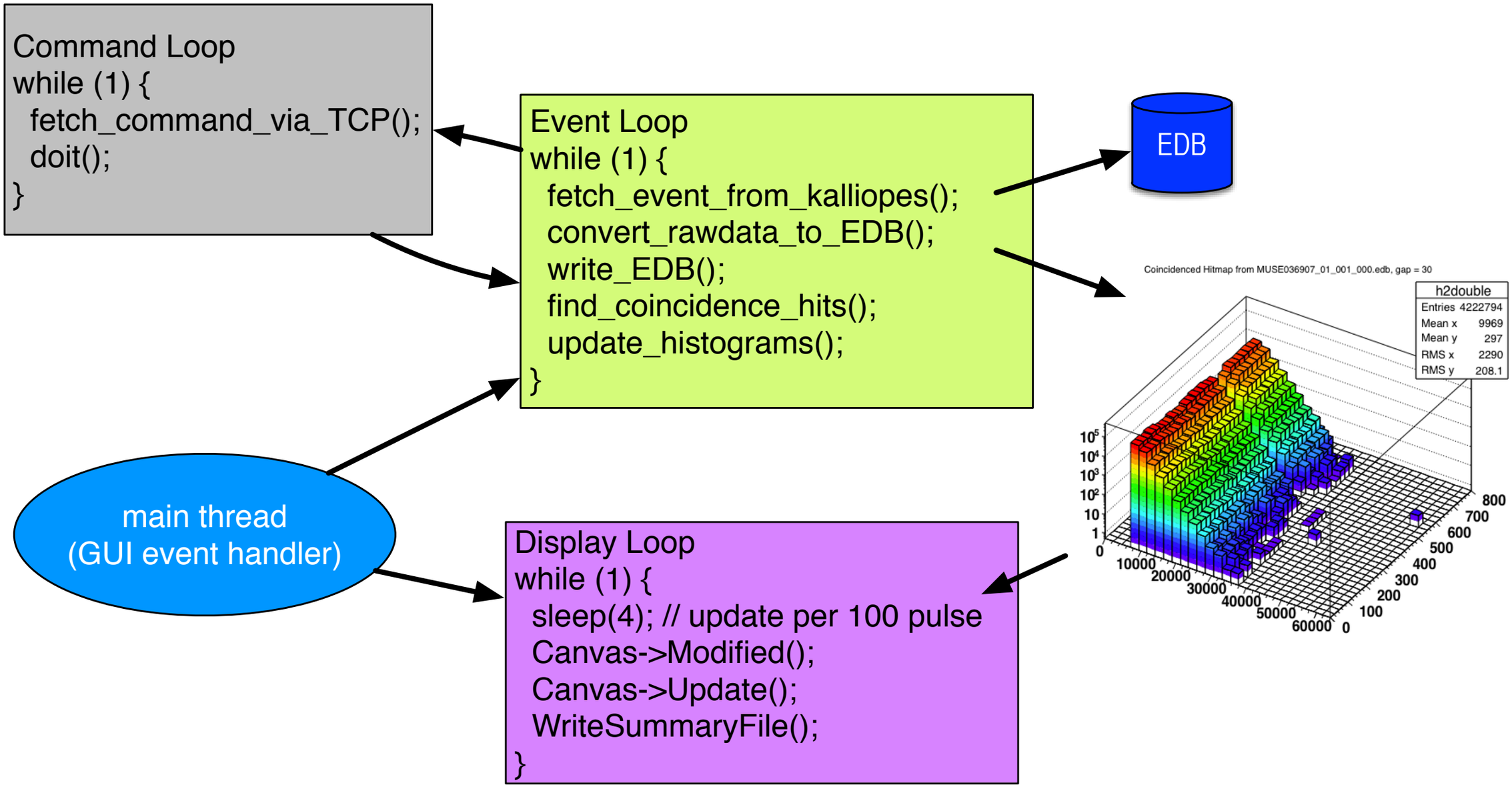
Main Threadの取り合い

- DAQMWもROOTもpthread必須
- DAQMWはmain()を自分が最後に掴んでイベントディスパッチの無限ループに入りたい。

- ROOTでGUIを構成するとmain()を自分が最後に掴んでイベントディスパッチの無限ループに入りたい。

- どっちも使いたいとムリなので
  - GUIをブラウザから操作するのはあきらめて、
  - ROOTの方に近づくことにした。
  - T2KのオンラインモニタもROOTのGUIで作ったので、なるたけ同じ風にしたかった

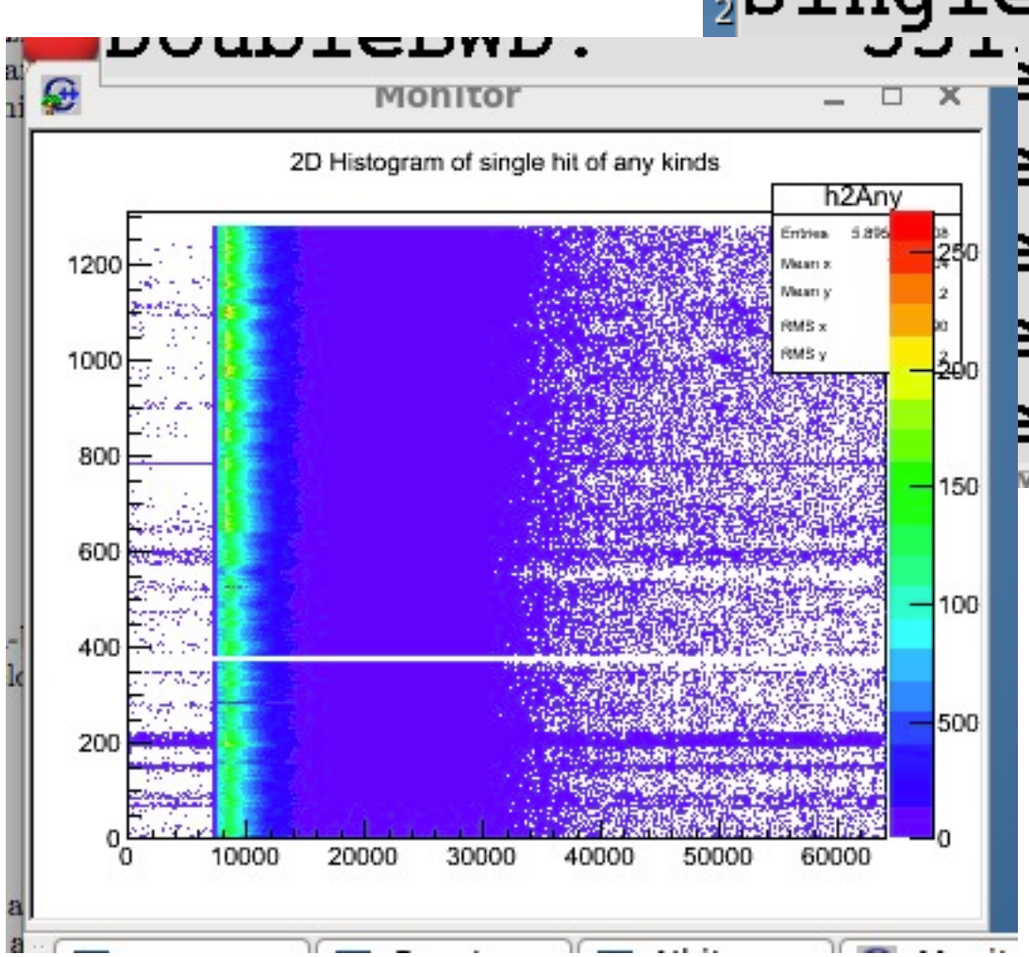
内部構造

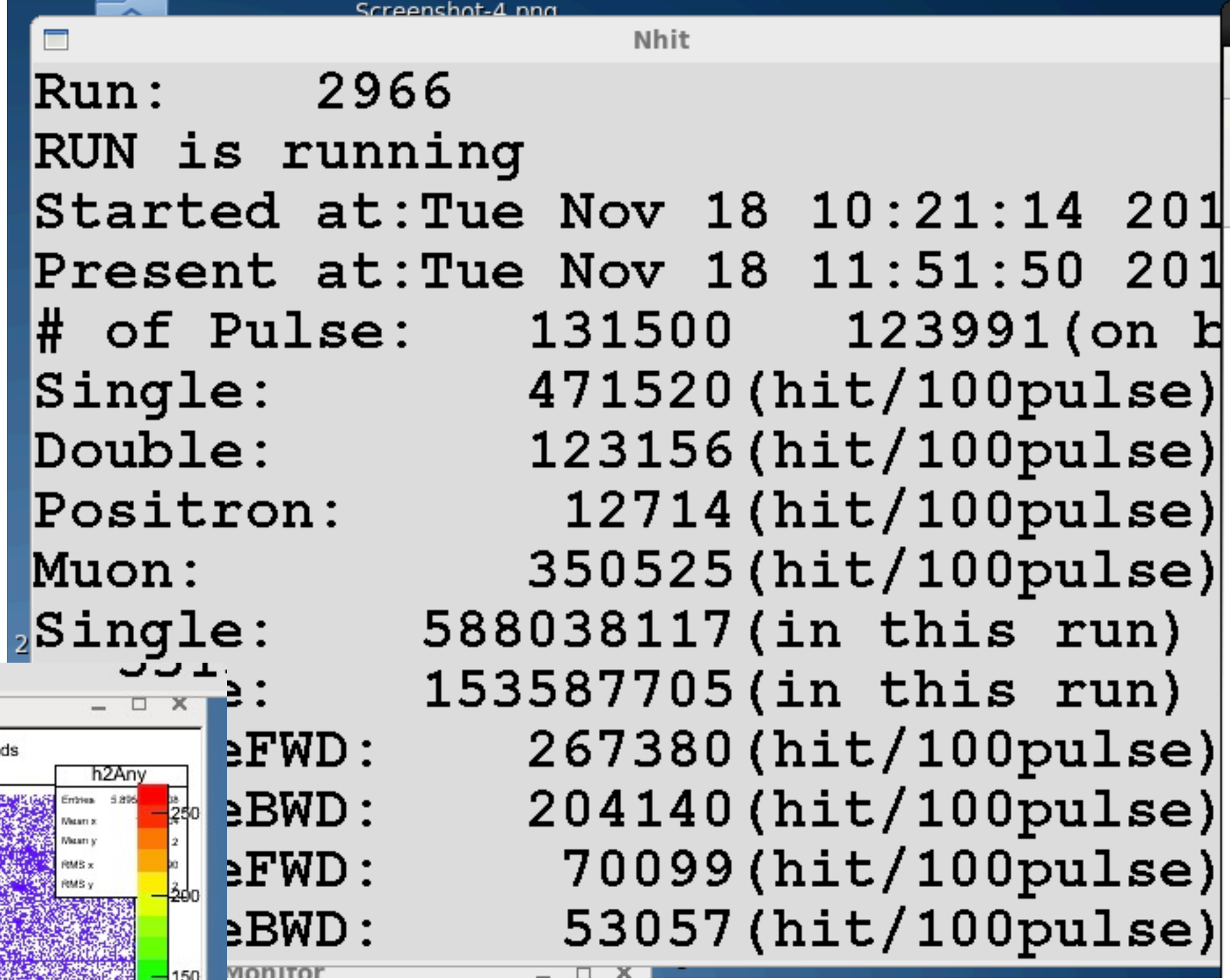


スレッドの動的生成・破壊は基本的に行わない

チャンネル・ヒット

時刻分布





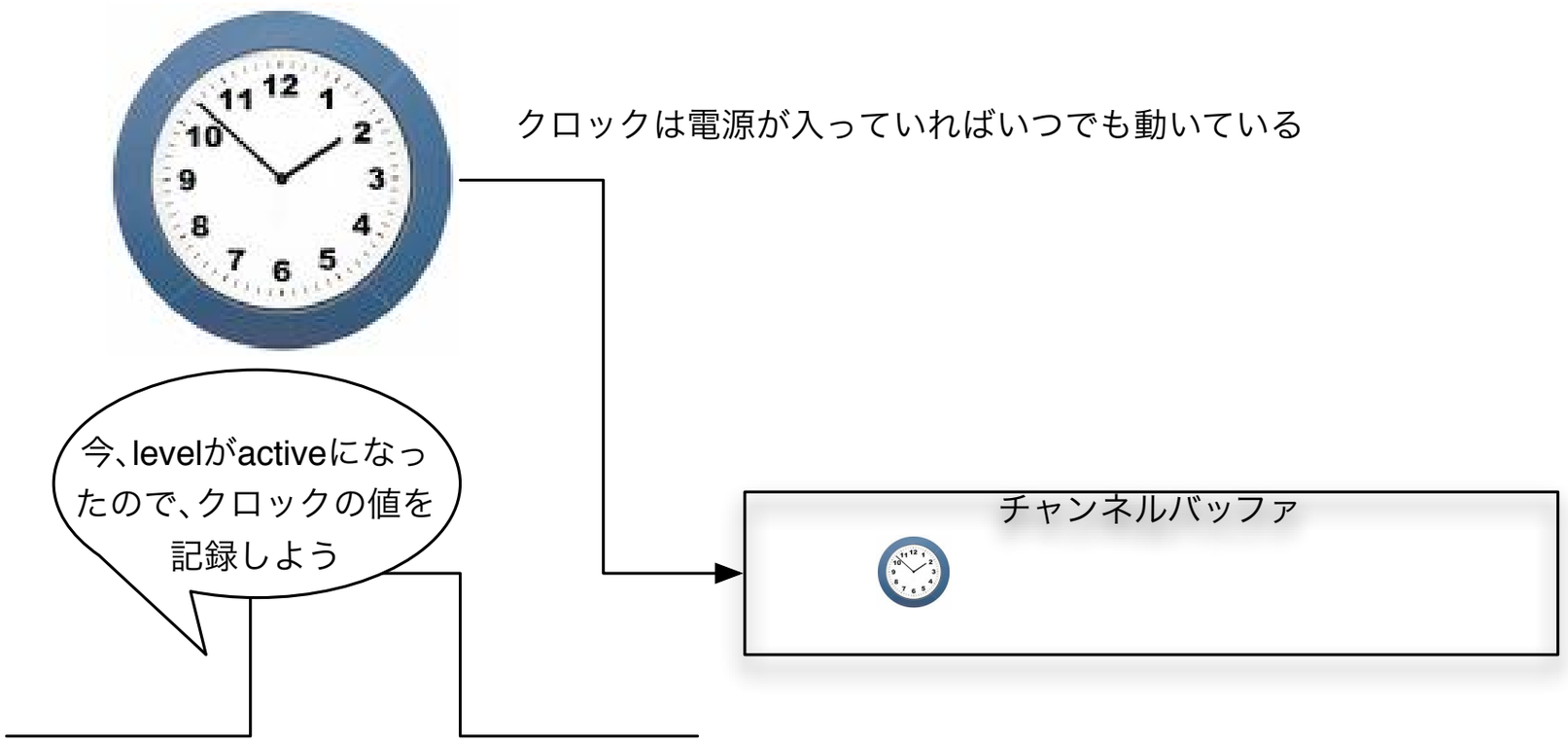
ヒット数表示

EDB

- 元々はDAQMWのLoggerが書き出すファイル形式
- 元々DAQMWはGathererが受信したデータをそのままファイルに書き出していた
  - 固定長ワード(64bit ただしiBIXは16bit)の羅列
  - 基本的に隣のワードによって意味は変わらない
- FPGA-TDCからの生データは場所依存
  - 時刻の基準値が各トリガーの最後に出現する
  - チャンネル番号がDAQシステム全体のユニーク番号になってない
  - 1ワードでヒット情報がわかるようにしてから書こう
- 1ヒットあたり64ビットで内部的に記憶しておき、イベント毎にまとめて吐き出す
  - ヒストグラム処理はその羅列を見て行う
  - オフライン処理とまったく同じ処理ですむ

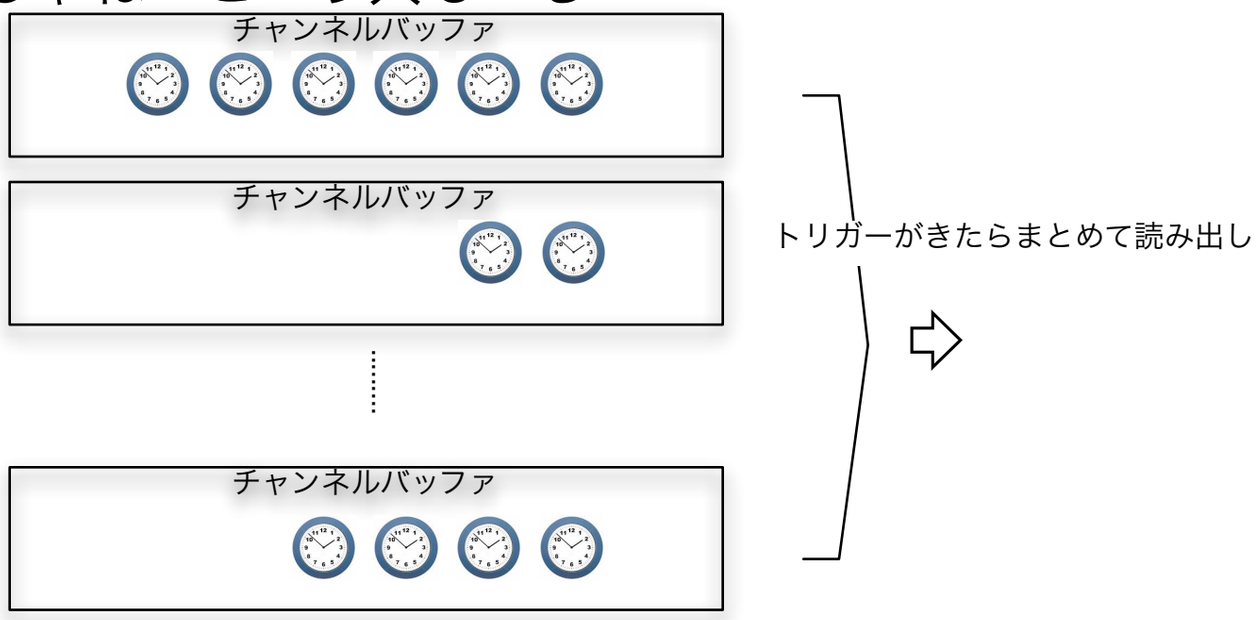
FPGA-TDCの仕掛け

- TDCはクロックカウンタとチャンネルバッファで出来ている
- クロックカウンタ(時刻)は1ns単位でいつも更新されている
- FPGAは入力信号の立ち上がりを見たら時刻をバッファに記録
  - バッファはチャンネル毎にある。(32本+トリガ用1本)



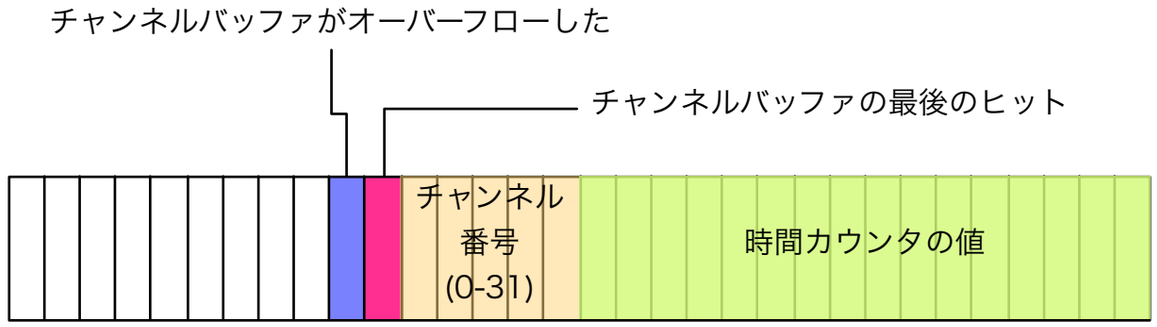
トリガーが来ると

- トリガーが来ると全部のチャンネルバッファから時刻情報を全部読み出す
- トリガーはコモンストップとして扱われ、指定したタイムウインドウだけ引いたカウンタをスタート時刻とする
- 各時刻からスタート時刻を引き算すれば時間(差)がわかる
  - スタート信号は16本の検出器からの信号と別にNIMで入力する
  - 「時刻」を記録しているだけで、「時間(差)」を計測していない
    - こんなのTDCじゃねーという人もいる

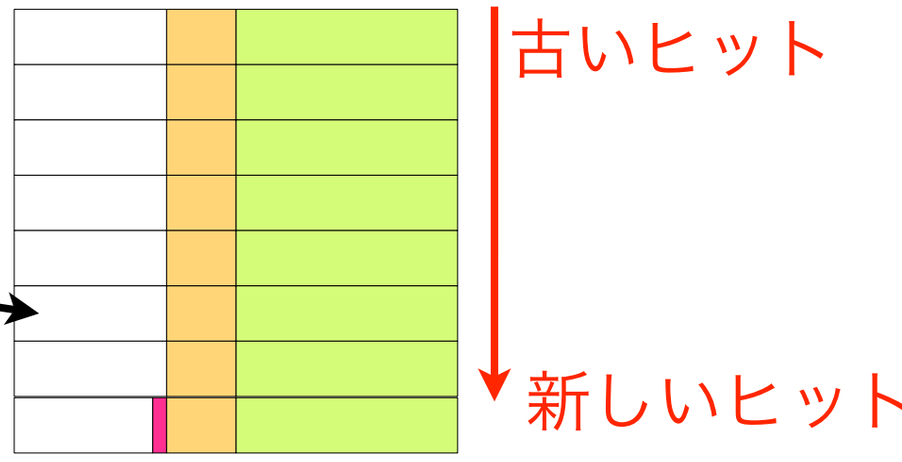


TDCのデータ構造

- 各チャンネルのバッファに入るデータ
  - ヒット情報のビットパターン



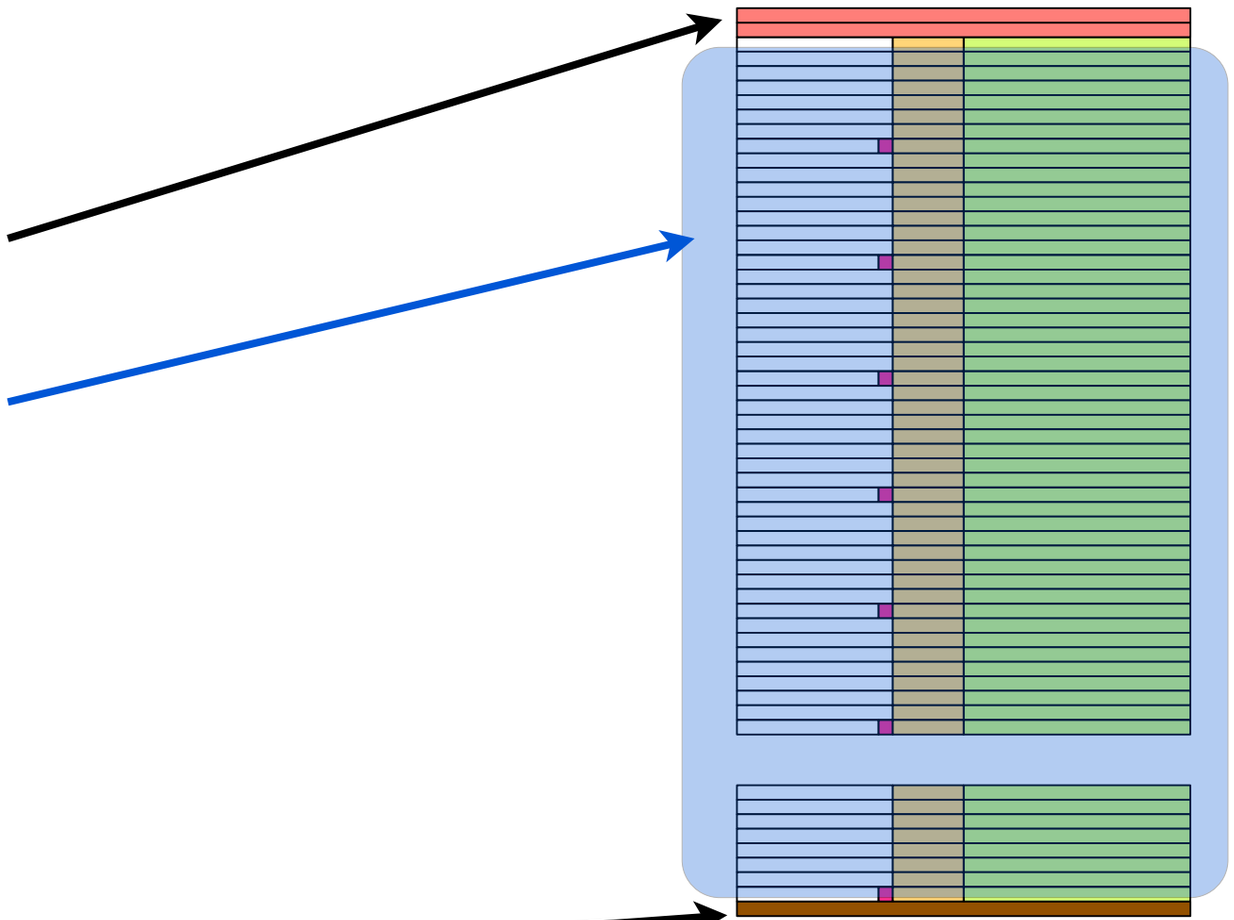
  - これがヒット数だけ貯まる
  - バッファから読み出すときはつながって出てくる
  - バッファの最後のヒットにはマークがついている
  - 1本のバッファを読み出すとこうなる



  - これが32本分集まって1つのKalliope分

Kalliopeからのデータ構造

- ヘッダ
  - マジックワード2種
  - 長さ、カウンタ等
- 各チャンネルのデータ
  - 0-31(0x1F)まである
- フッタ
  - スタートタイミング
  - マジックワード

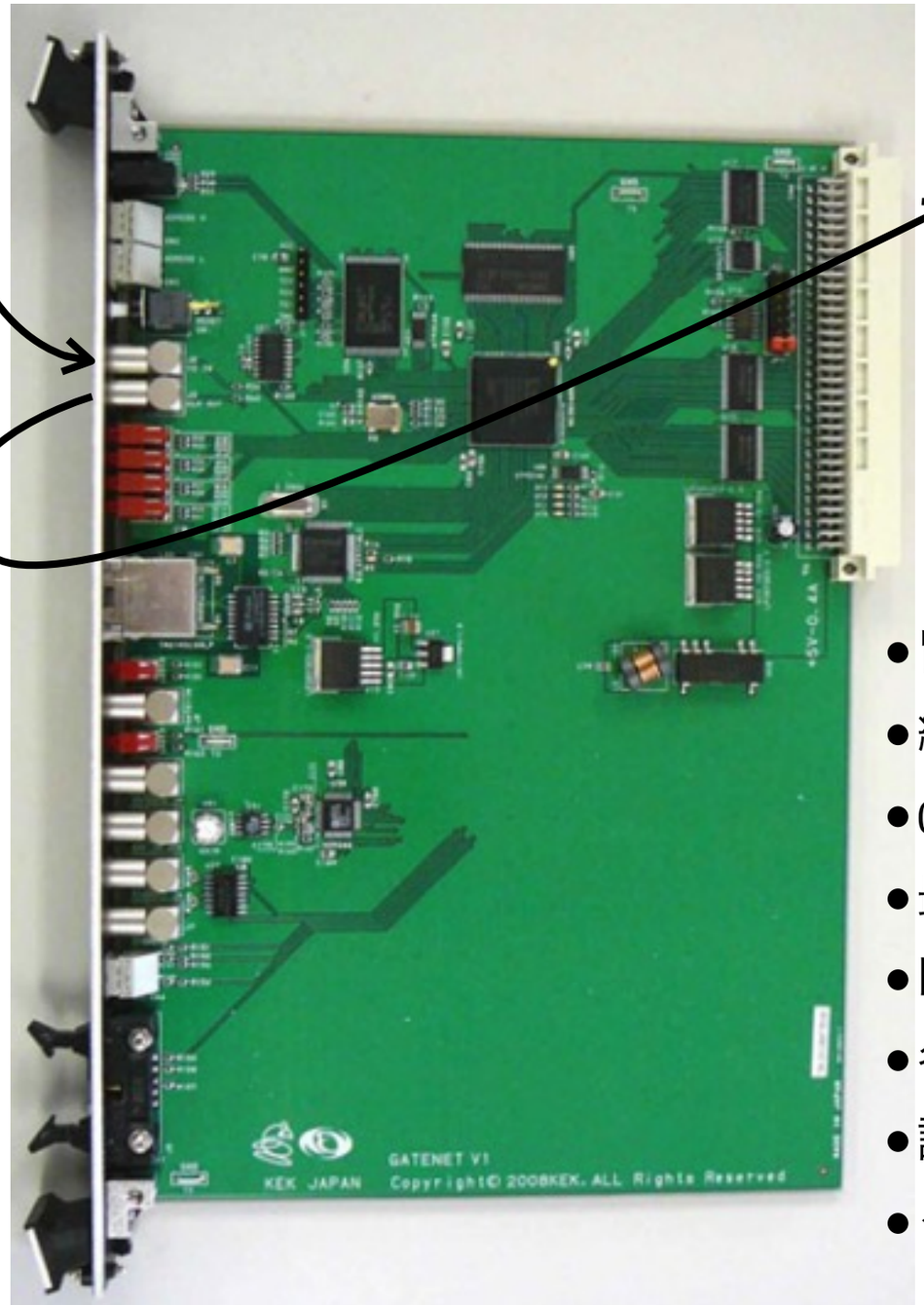


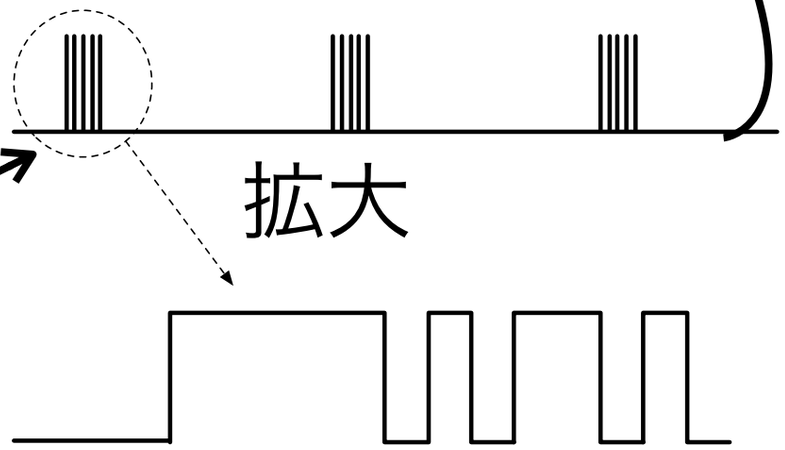
トリガー配布

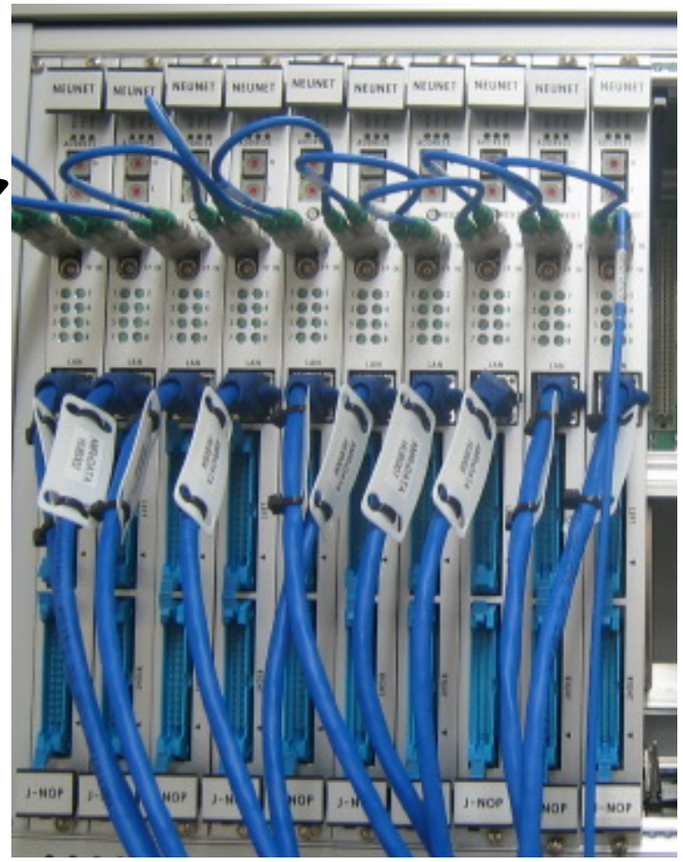
- 加速器からのキッカーパルス(25Hz)をFANOUTしてくべている
  - Trigger-Busy handshake のようなことはしていない
  - FANOUTモジュールが不自然に壊れると一巻の終わり
- 同期はNIM-TDCでとっている
  - 古典的にはVME Output Registerの出力をVETO信号とするアレ
  - NIM-TDCにはSiTCPが接続されているときだけHighになるような出力があるので、それをVETO信号として使用する
  - すべてのKalliopeに接続してからNIM-TDCに接続すると、同時にトリガーが支給され始める
- GateNetにもくべる
  - しかしGateNetのTTL出力は誰にも渡していない
  - パルス毎のビーム強度情報などをつきあわせたいと思うが、まだ果たせず
  - NIMはあってもVMEがない文化になりつつある

GATENET

Kicker pulse







- 冒頭に入力タイミングに合わせて5μs
- 続けてタグ104ビットをエンコードして羅列
- 0→1μs, 1→2μs, スキマ1μs
- 最大317μsかけて配布 (25Hzには十分速い)
- 時刻とクロックカウンタが埋めてある
- 各モジュールに前から「トリガーとして」daisy chainで入れる
- 読み出しモジュールはこのタグ情報をヒット情報とセットで送出
- タグにクレート番号を含める→通しのチャンネル番号

仁木モジュール

- MuSR実験だけならばTDCだけでよかったが、ADCも必要だ!
  - MUSEではg-2やX-ray実験も行われるので
- ADCや非常にダイナミックレンジの広いTDC、と、読み出し用のCPUモジュール (VME)
- これらはMuSR専用に開発されたわけではない
  - FPGA-TDCも専用ではないが、こちらの意見は不足なく実装されている
  - トリガーカウンタとヒット情報が出力される
- 読み出さないと出てこない
- 読み出すとゲートが閉じてなくても出てくる
- ヒットがない場合トリガーカウンタも出てこない
  - だいぶ異なる
- 全力ビジーループで見張ると読み出しモジュールが機嫌を損ねる

ADC/TDCへの入力信号

Event# Nの
Trigger (t0)

Event# N+1の
Trigger (t0)

Time

read_sitcp

read_sitcp

read_sitcp

read_sitcp

read_sitcp

read_sitcp

read_sitcp

ふむ、まだN番のトリガだな

ヒットがなくなったから次のN+Iが来るのかな?

がーん、まだN番だった

ヒットはなくなったけど、
N+Iかわかんないよね...

それで困るのかね?

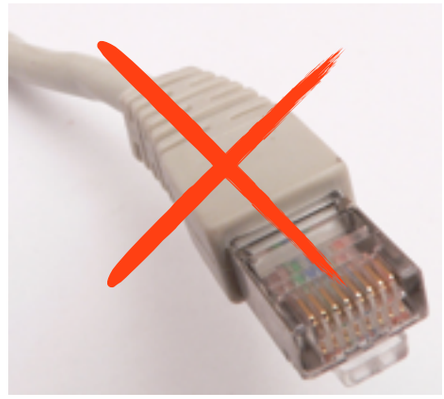
- めちゃ困る
- 「トリガーがあったがヒットがない」ことは次のトリガーが到達して読んでみないとわからない
- 次のトリガーがヒットがなかった場合、さらに次のトリガーがヒットを含んで来ないとわからない
  - 1トリガー分だけバッチリ読むことができない
- かなりの難物
- VETOがあるみたいなので、落ち着いて読み出し方法をよく考えた方が良さそう
  - いずれにしてもソフトウェアビジーループを求める読み出し機器は危険

データ量の増大

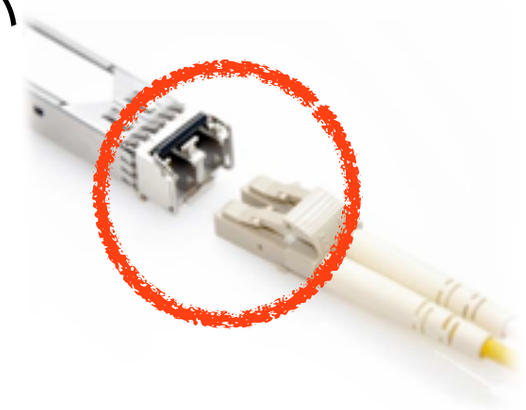
- Kalliope化した結果、チャンネル数増える
  - 256ch => 1280ch
  - そもそも増やして1チャンネルのパイルアップを減らすのが目的
- ビーム強度増える
  - パイルアップが減るからビーム強度あげてもいけるぞ
- 当然データ量増える
- 増強が...
  - ネットワークスイッチ
  - ディスク(容量+速度)
  - CPU速度

ネットワーク

- 1G x 40 を1Gで受けている
  - まあ大丈夫なんじゃないかなー(根拠無し)
    - 各Kalliopeからは平均スループットはそこまで出ないはずだし
  - ていうか10GbEは高いし



- だいたい磁場中にSiTCPを送り込むとトランスを使用する
  1000BaseTは使えないので1000BaseSXを使うしかない
  - 突然ポート単価急上昇
  - SFPが40ポートもあるスイッチは超高額(市場が小さいため)
  - せっかくなので10G級のXFPなスイッチを買った...けど
  - 10Gの光トランシーバも超高額
  - もじもじしていたらXFPからSFP+の時代になった。



10Gイーサネット

- 1Gトランシーバは2種類(GBIC, SFP)しかなかったのに、10Gはやたらとある
  - XENPAK, XPAK, X2, XFP, SFP+ (いずれも10GBaseTはない)
  - 古いタイプは3Gbps x 4本を束ねて10Gbpsにしている(XAUI)
  - 新しいタイプは10Gbpsで入出力(XFI,SFI)
- 新しい規格になるほどトランシーバの仕事が少ない
  - 安い、小さい、消費電力小
  - 古い規格は市場がもう成長しないので値下がりしないまま製品が消える
- XFPかSFP+は同軸ケーブルに載せる規格(10G-DA)で、1万円程度で接続可能。ただし射程7m。
  - XFPのDAはほとんど製品がなく、SFP+ばかり
  - 他は一桁上
  - DAだけベンダーロックがない

- DAでいいなら40Gも一社が価格攻勢をかけている

ディスク

- 20M hit per 100 pulse あたりで危険な臭い
  - 40MB/sくらいでEDBを書く
- じゃあ速く書け
  - DAQ PCは**NFSで**EDBを書いている
  - 大丈夫?
  - 進行状況確認のため、EDBを定期的にコピーして解析
  - この処理中が危険かも
- DAQ PCに直接HDDを付ければ速くはなるが、どうせコピーは必要だ
  - J-PARCのRUN(月単位で呼ばれるRUN)が終了するたび節電のため空調及び電源を落とす前提
  - DAQ PCにどんどこデータを貯めるとあまりよろしくない

CPU

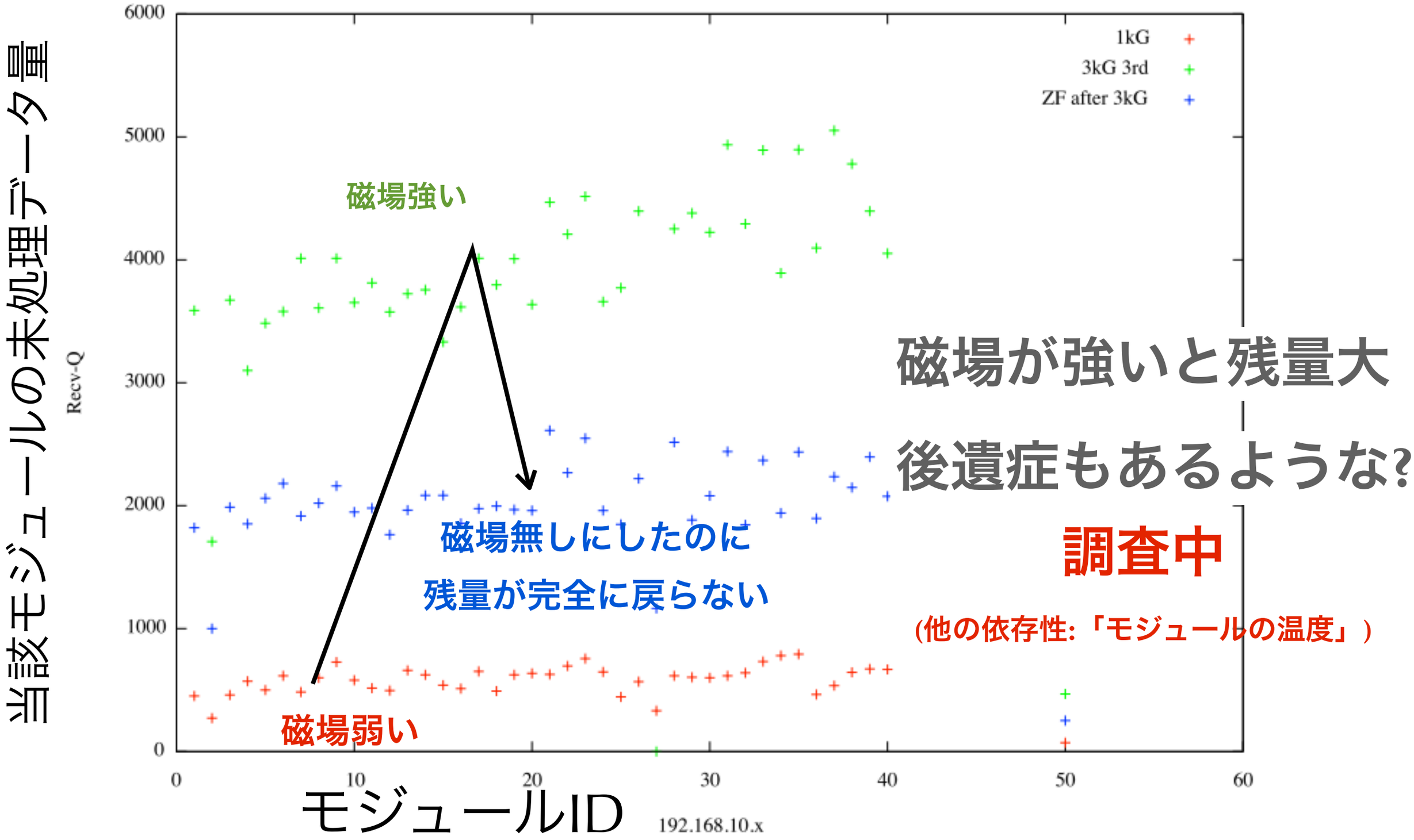
- MuSR実験ではe^+をノイズ由来の信号と間違えないよう、2つの検出器でコインシデンスを取っている
  - 一応ディスクリであまりに低いものは捨てられてはいる
  - ソフトウェアでコインシデンスを取る
- チャンネル数およびチャンネルあたりのヒット数に応じて探索の時間が増える
- 40ms以内に探索等もろもろが終わらないとSiTCPからの受信が遅れる
  - KalliopeのSiTCPは垂れ流しタイプなので、どしどし送ってくる
  - 結果、ソケットバッファにデータが残る
- コインシデンス探索をマルチスレッド化するしかない

netstat -nat で読み残し確認

```
% netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp    58896      0 192.168.12.251:59046       192.168.10.1:24             ESTABLISHED
tcp   133136      0 192.168.12.251:43429       192.168.10.2:24             ESTABLISHED
tcp    88460      0 192.168.12.251:52746       192.168.10.3:24             ESTABLISHED
tcp    77344      0 192.168.12.251:36533       192.168.10.4:24             ESTABLISHED
tcp   291180      0 192.168.12.251:46180       192.168.10.5:24             ESTABLISHED
tcp    83488      0 192.168.12.251:45636       192.168.10.6:24             ESTABLISHED
tcp    80048      0 192.168.12.251:36196       192.168.10.7:24             ESTABLISHED
tcp    55616      0 192.168.12.251:45347       192.168.10.8:24             ESTABLISHED
tcp    99148      0 192.168.12.251:48562       192.168.10.9:24             ESTABLISHED
tcp   124380      0 192.168.12.251:60081       192.168.10.10:24            ESTABLISHED
tcp   172512      0 192.168.12.251:47331       192.168.10.11:24            ESTABLISHED
tcp   106204      0 192.168.12.251:46785       192.168.10.12:24            ESTABLISHED
tcp    69496      0 192.168.12.251:52103       192.168.10.13:24            ESTABLISHED
tcp   132748      0 192.168.12.251:33251       192.168.10.14:24            ESTABLISHED
tcp   127324      0 192.168.12.251:42520       192.168.10.15:24            ESTABLISHED
tcp   106592      0 192.168.12.251:57645       192.168.10.16:24            ESTABLISHED
tcp       36      0 192.168.12.251:48860       192.168.10.17:24            ESTABLISHED
```

Receive Queueがニョキニョキ伸びる

条件次第で危険な伸び具合



磁場が強いと残量大

後遺症もあるような?

調査中

(他の依存性:「モジュールの温度」)

いずれスケールアップには
マルチスレッド化は不可避

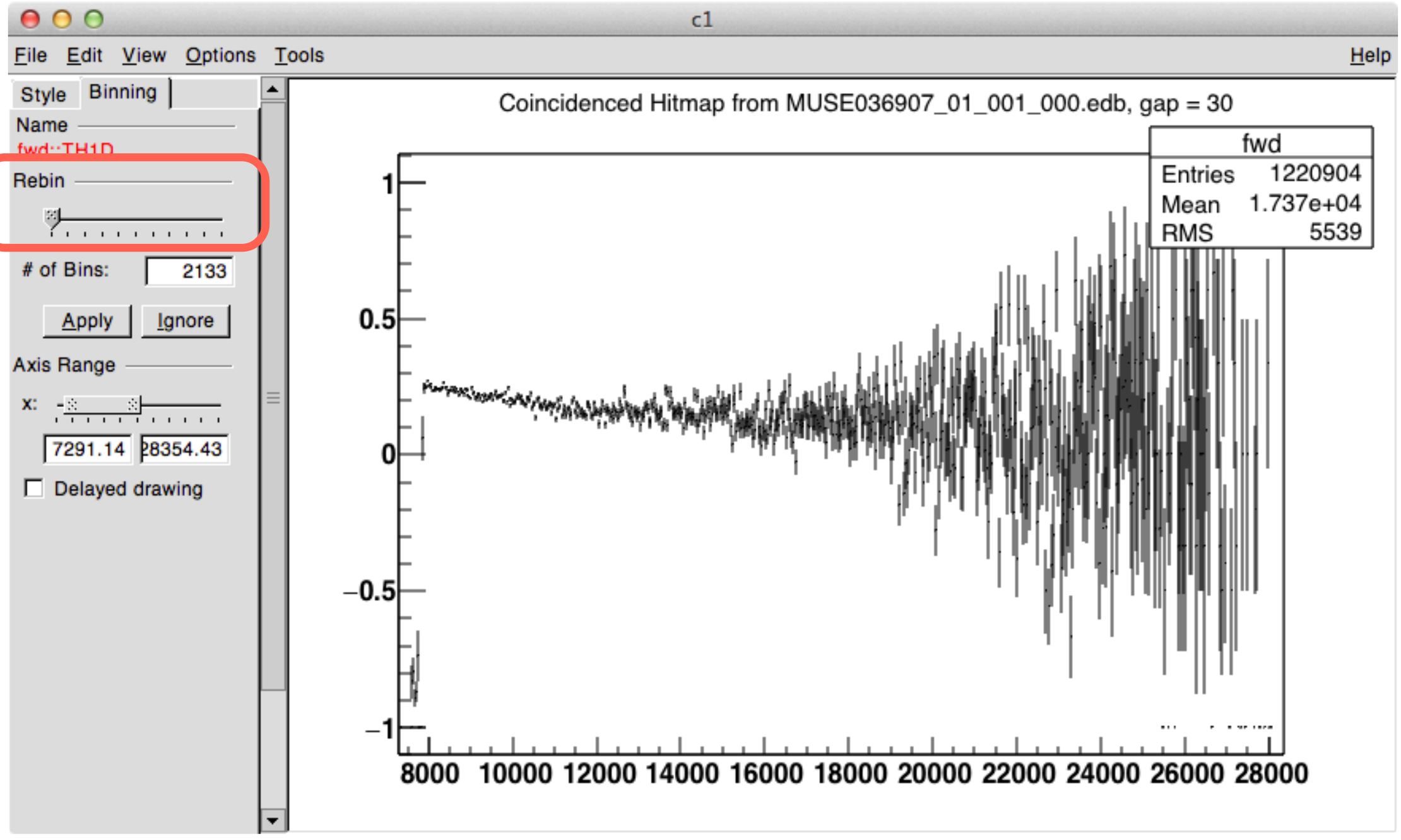
- ROOTにTThreadはあり、マルチスレッドで走れる
  - そもそもGUIを使うと否応なくマルチスレッドで走ってはいる
- しかし同一のTFileに複数のTThreadから書くことは出来ない
- 他にもROOT5には多くの大域変数があり、排他制御されている
  - インスタンス生成・破壊時に排他制御が行われる
  - I/Oがらみはデッドロックが発生することがままある
- 並列イベント処理にはROOT5では「PROOF」を使うのが定石
  - 別プロセスにオブジェクトを投げつけて返事を回収する
- ROOT6では複数スレッドで同一のTFileに書き込めるようにしたいようだ
- 火急の問題としてROOT5でどうにかしなくてはならないのだが...

ヒストグラムをモニタに渡す手軽な方法は?

- ヨソの典型例はおおまかに2系統
- 定期的に画像ファイルを生成し..(略)
  - チャンネル数が1280あり、コインシデンスの有無でさらに倍必要...勘弁して!
- ヒストグラムを生成するプロセスがソケット接続を受け付け、受信したプロセスはそれを表示する
  - JSROOTもこの方式
  - UI側のズームには追従できるが、ビニングの変更に追従できない
- TMapFile
  - 書く人が死んだら読む人はそのファイルを閉じないとまずい
  - 生成・終了時の取り回しが面倒
  - ビニング問題はやっぱりある

BINNINGの変更

これ



ROOTは中で何を
やっているのか

- もとのヒストグラムを一度待避し、
- スライダを動かすとTH1::Rebin()して表示する
  - この時点で元のヒストグラムの中身は書き換わっている
- Applyすると待避したヒストグラムを破棄する
- Ignoreすると待避したヒストグラムから戻す
- スライダで変更したが、まだApplyもIgnoreも押してない間にFillすると
  - 表示しているヒストグラムには反映される
  - 待避しているヒストグラムには反映されない
  - Ignoreを押すとその間のFill操作はなかったことになる
  - マルチスレッドでFillしているとRebin()作業中に範囲外のBINにFillしてしまうことがある(→SEGVで死亡)

MuSRの場合

- ヒストグラムを書く人: DAQプログラム本体
- 1ns単位のヒストグラムを作り、定期的にTFileに書き込む
- /dev/shm配下に置く
- ヒストグラム読んで表示する人: モニタプログラム
  - そのファイルを開き、格納されている全てのヒストグラムについて、
  - メモリ中ににある同じ名前のヒストグラムをReset()して0クリアし、
    - メモリにあるヒストグラムとファイル中のヒストグラムは別のもの
  - ファイル上のTH1からGetBinContent()してメモリ上のTH1にFill()しなおす
  - ...とBINNINGの変更には追従できる。
  - 表示はTBrowserに任せる
- 正直「苦し紛れ」

そこまでして

BINNING変更したいのか?

- 荒くした方がエラーバーは小さくなる
- どのくらい荒くして良いかは周期で変わる
- 周期は磁場の強さで変わる
- BINNING固定はあまりよろしくない
  - 可変にしようとするとこの有様

まとめ

- 2006年に着手して以来、MUSE用のDAQシステムは時々刻々と進化してきた
  - 疎結合の副産物として
    - ノート型のMac1台とKalliope 1台で容易にテスト
    - 検出器1280チャンネル分の電圧の調整の自動化
- MuSR実験に限れば当初予定の最終形にたどり着いた
  - 現時点で40MB/s程度だがまだスケールアップは必要
- MUSEの他のDAQにも充当されつつ有り、既製品の他の読み出しボードも組み込み中
  - 予想を超えて使われるのはありがたいことですが、反面怖い
  - 開発者としての勉強にはなります

反省点いろいろ

- GATENETをKalliopeにくべてもらうべきだった
  - でもVMEのいらないGATENETが欲しいです
- 生データを格納していない
  - tcpdumpで事足りるが、瞬間的には使えても一晩走らせるのはムリだし、高負荷
- データ格納形式
  - ROOTで書くのは危険なことがあるのでとりやめた
    - オブジェクトの破壊サイクルと同期するとTFileを閉じるときにSEGVすることがある
  - でもEDBにした意味はあったのだろうか...
- DAQプログラムの構造
  - ROOTとC++で悪のりしすぎた嫌いがある
- 調査不足
  - GSIのJSROOTを知っていればBINNINGの追従はやめたかも
  - XFP vs SFP+