J-PARC MLFにおける
データリダクションソフトウェアの開発

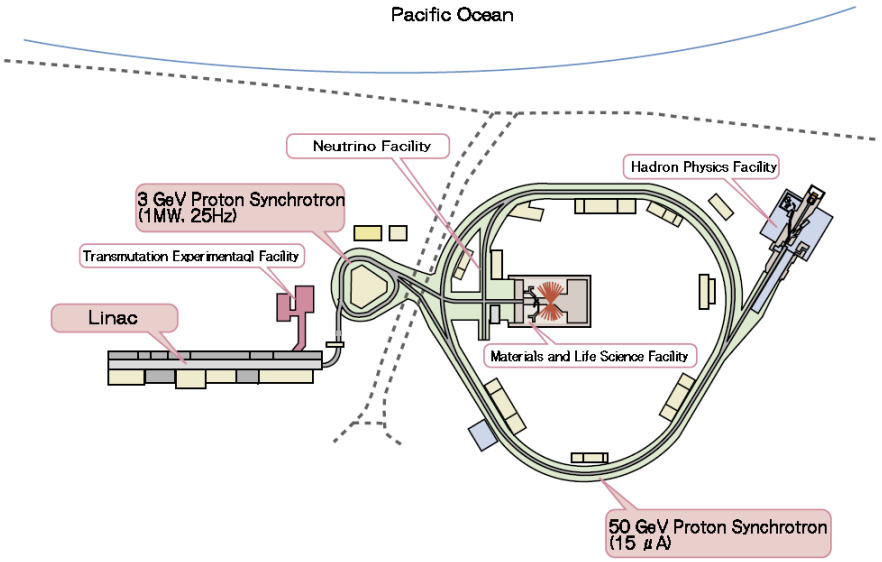
H. Ohshita (KEK IMSS)

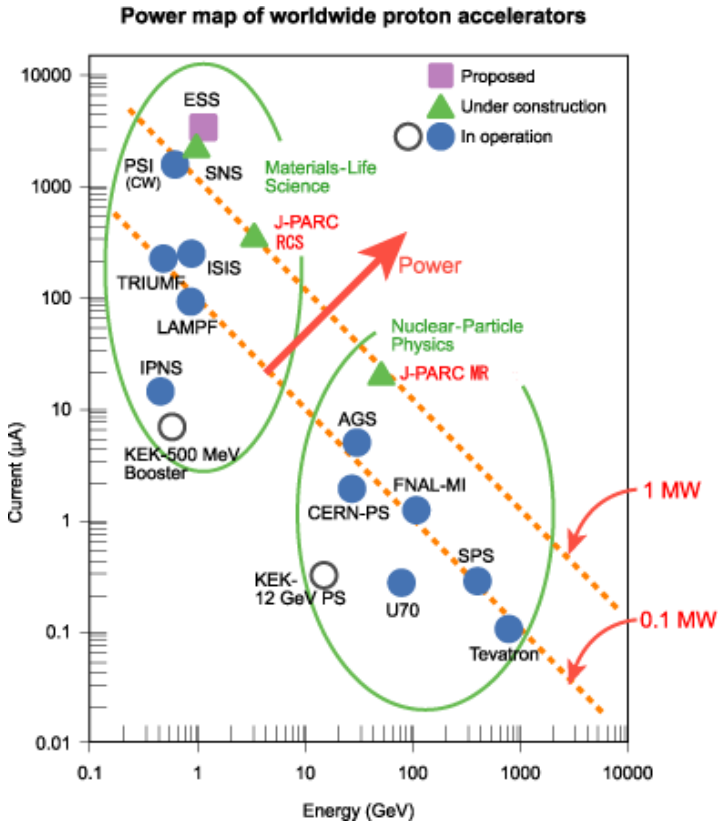
目次

T. Otomo、K. Ikeda、T. Honda、N. Kaneko、T. Seya、K. Suzuya[1]、Y. Yasu、K. Moriyama[2]、Y. Inamura[1]
(KEK IMSS、[1]JAEA J-PARC Center、[2]CROSS)

Japan Proton Accelerator Research Complex (J-PARC)

- 高エネルギー加速器研究機構と日本原子力研究開発機構との共同プロジェクト
- 3種類の加速器施設
  ⇒線形加速器・ 3 GeV シンクロトロン・ 50 GeV シンクロトロン
- 2次ビーム（中性子、μ粒子、k粒子、ニュートリノ）の利用



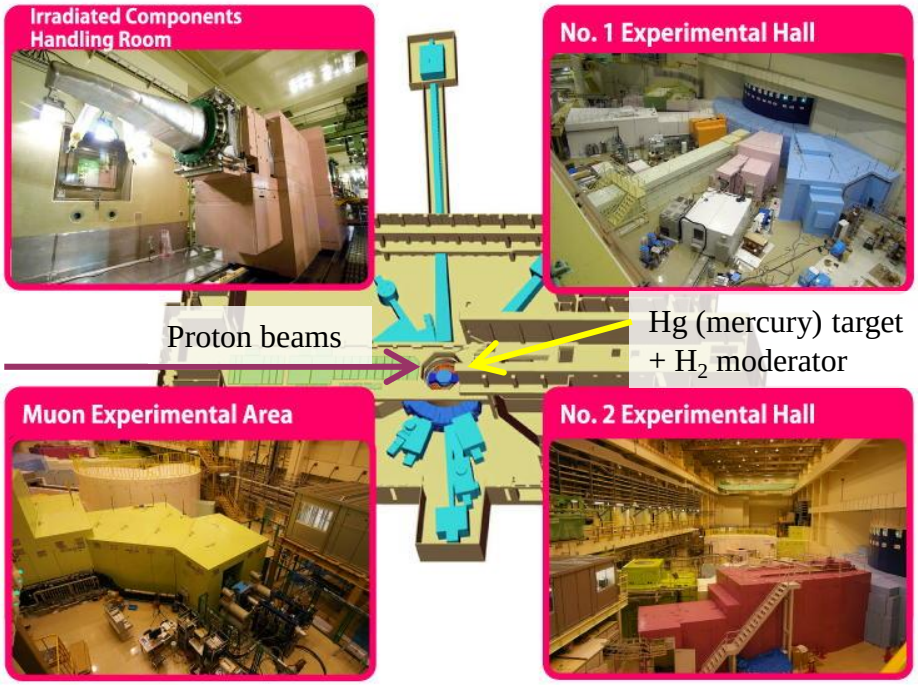


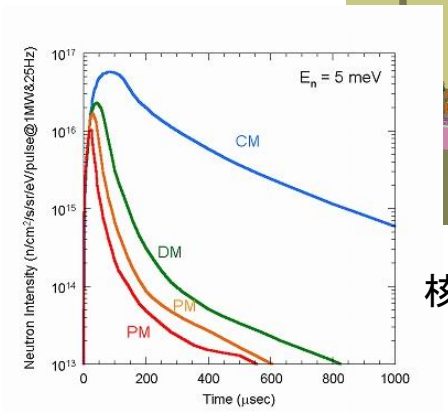
http://j-parc.jp/index.html

J-PARCでは物質構造科学、生命科学、素粒子・原子核物理の研究をおこなう

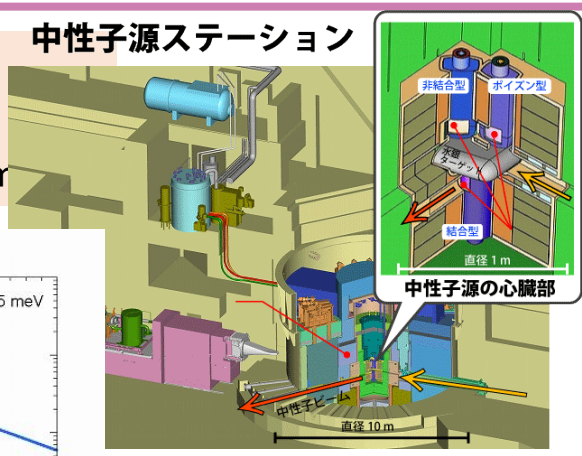
物質・生命科学実験施設（MLF）

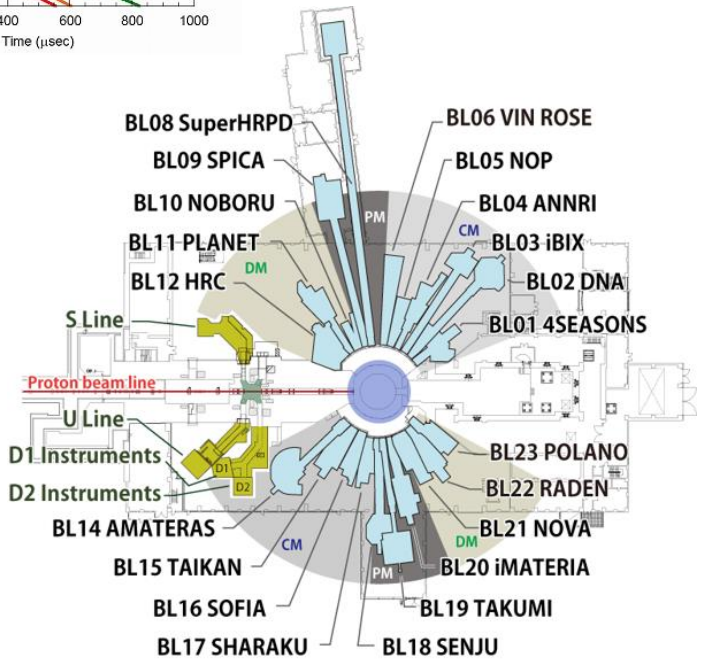
- MLFでは物質構造科学、生命科学、基礎物理分野の研究を実施
  ⇒23本の中性子ビームラインと4本のミュオンビームライン
- MLFは世界最高強度を誇るパルス中性子源：10^8～10^9 neutrons/s·cm





核破砕反応による中性子生成

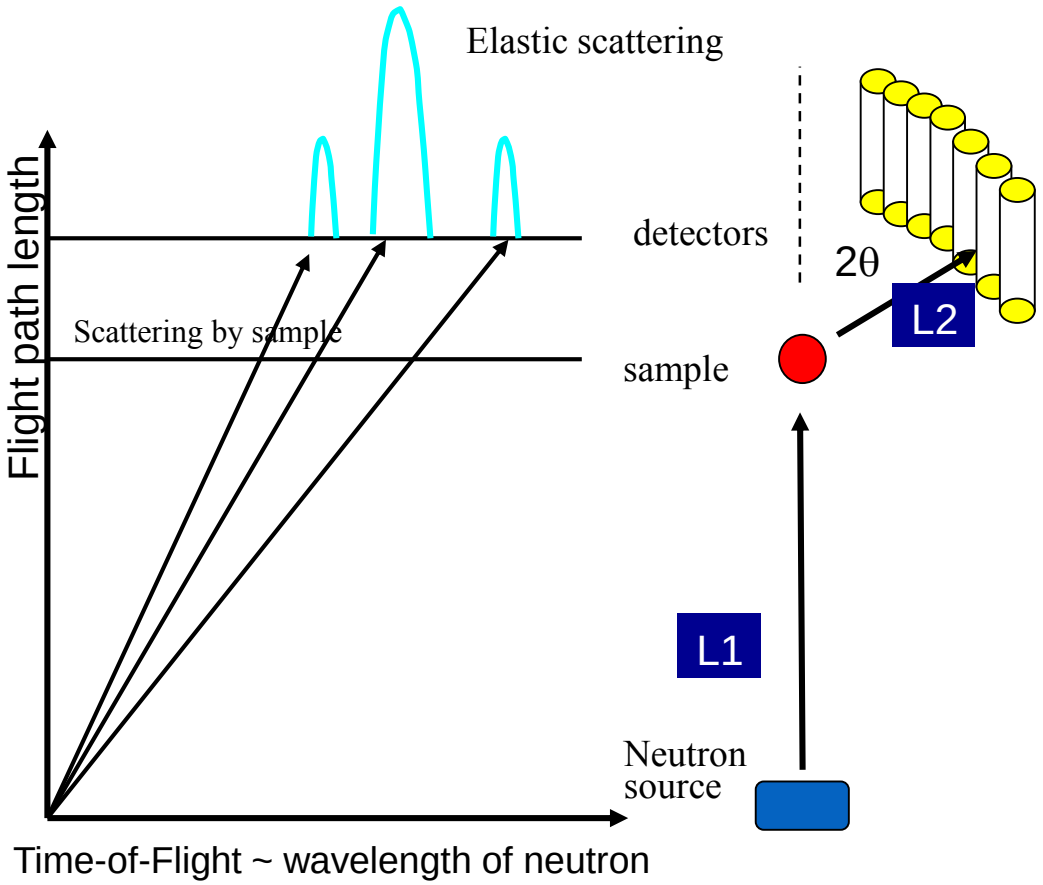




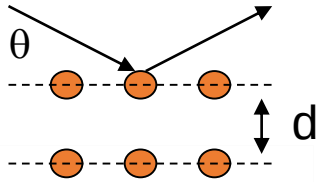
- Beam power: 1 MW
- Beam repetition rate: 25 Hz
- Neutron source: Mercury (Hg) target
- Neutron moderator: Supercritical hydrogen
- 3 kinds of moderator structure: coupled, decoupled, poisoned

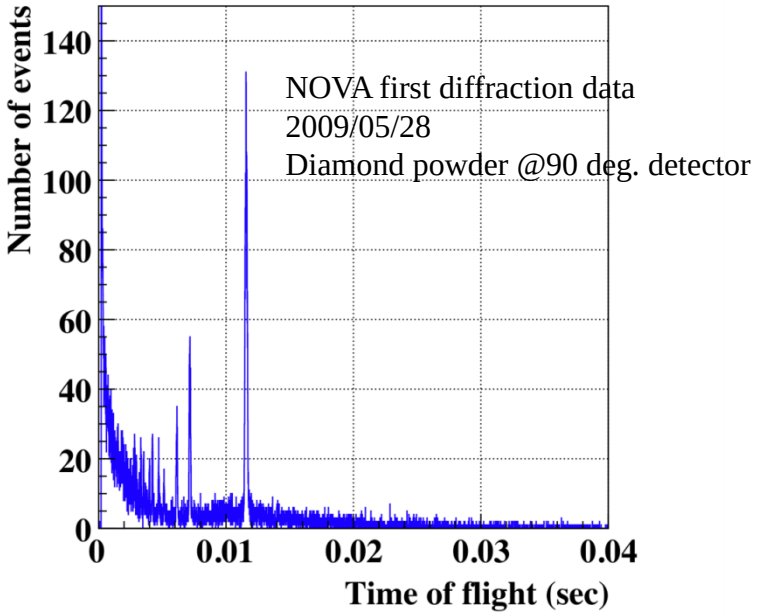
飛行時間法による中性子回折実験

- MLFでは飛行時間法（Time-Of-Flight（TOF） method）による中性子回折実験を実施
- ブラッグ条件に従い回折された散乱強度データから物質の構造解析が可能



$$\lambda = 2d\sin\theta$$





中性子源から時刻ゼロで放出された中性子は波長（エネルギー）に依存して
試料位置や検出器位置に到達する時間が異なる

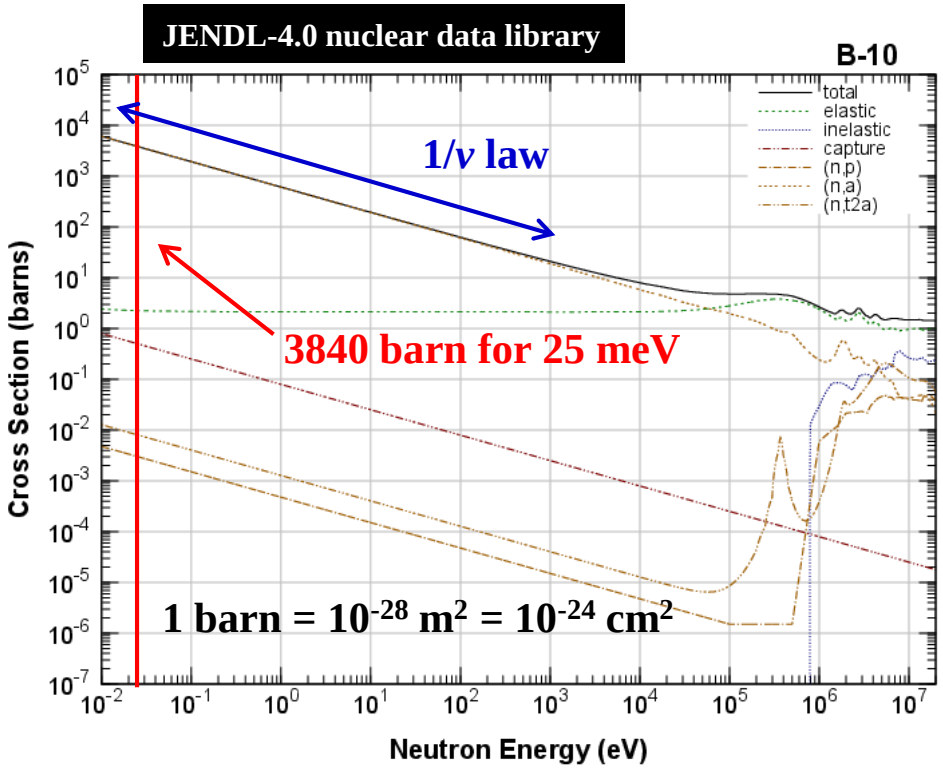
中性子の検出原理

- 中性子の検出は中性子反応で放出された荷電粒子を検出する間接測定

$^{10}B + n \rightarrow {}^{7}Li + \alpha + 2.79\ MeV\ (6\%)$
$^{10}B + n \rightarrow {}^{7}Li^{*} + \alpha + 2.31\ MeV\ (94\%)$
$^{7}Li^{*} \rightarrow {}^{7}Li + 0.48\ MeV\ (prompt\ \gamma)$

$^{3}He + n \rightarrow {}^{3}H + p + 0.765\ MeV\ (5330\ barn)$
$^{6}Li + n \rightarrow {}^{3}H + \alpha + 4.78\ MeV\ (940\ barn)$

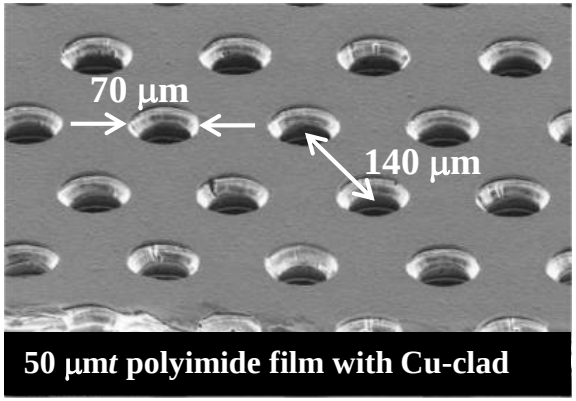
^{10}Bは^{3}Heの代替となる中性子コンバーターとして有望である
中性子反応は1/v則に従い、中性子エネルギー（波長）依存性を持つ



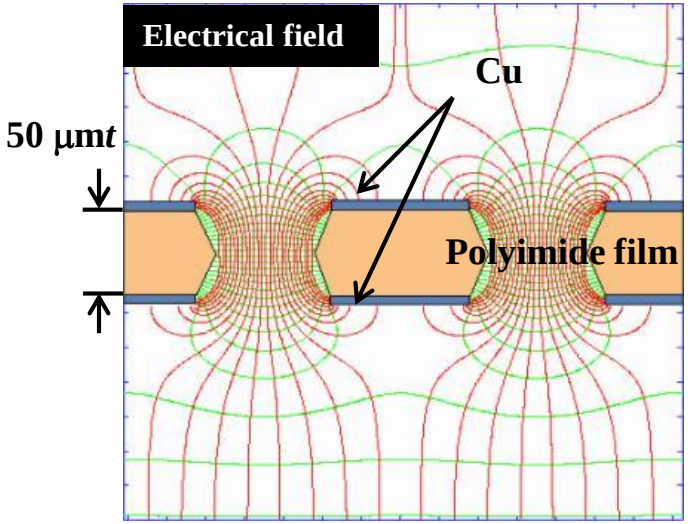
K. Shibata, et al., J. Nucl. Sci. Technol. **48** (2011) 1.

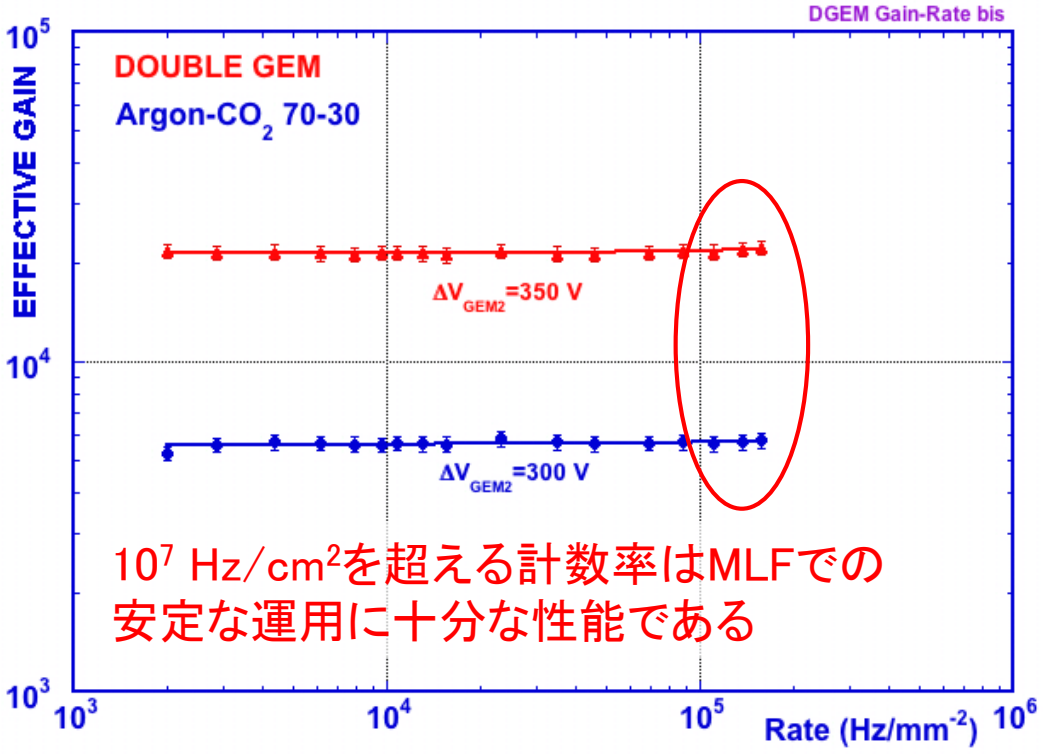
Gas Electron Multiplier (GEM)

- F. Sauliによって開発されたMicro Pattern Gas Detectors (MPGDs)の1つである
- 高い入射粒子頻度特性を持つため、MLFのような大強度の放射線環境下で安定な動作が期待できる



F. Sauli, Nucl. Instr. and Meth. A **386** (1997) 531.
http://gdd.web.cern.ch/GDD/

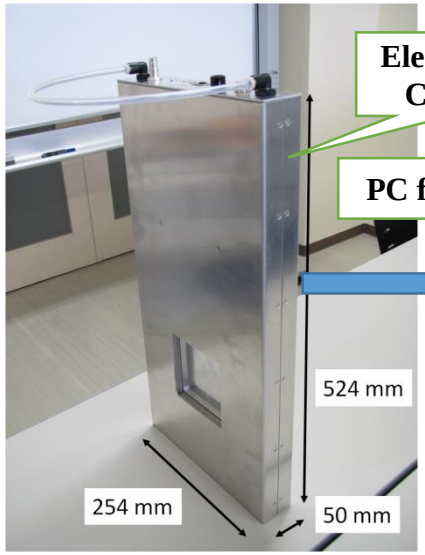




中性子ビームモニターnGEMの開発

- J-PARC MLFで使用可能な新しい中性子二次元検出器

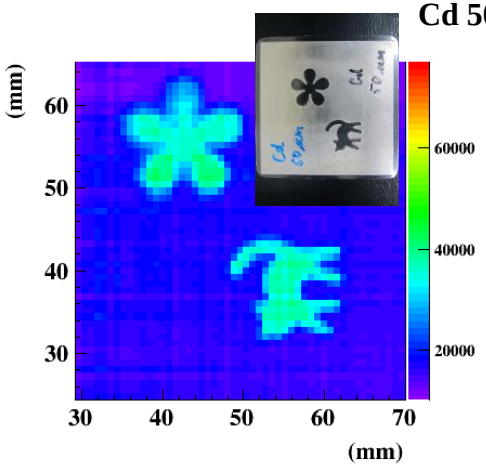
中性子ビームモニター、イメージング用検出器など高計数特性が要求される用途に利用可能

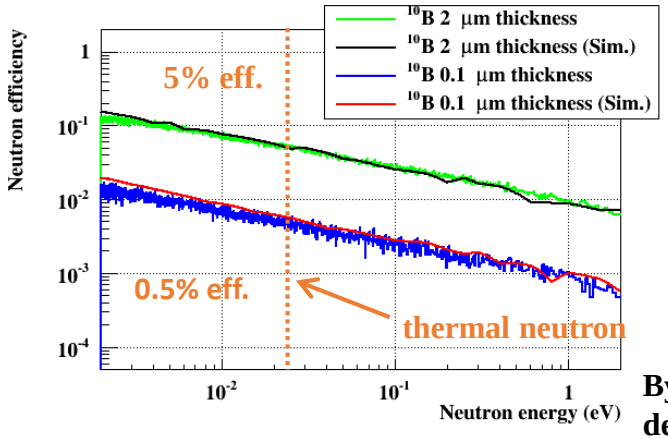


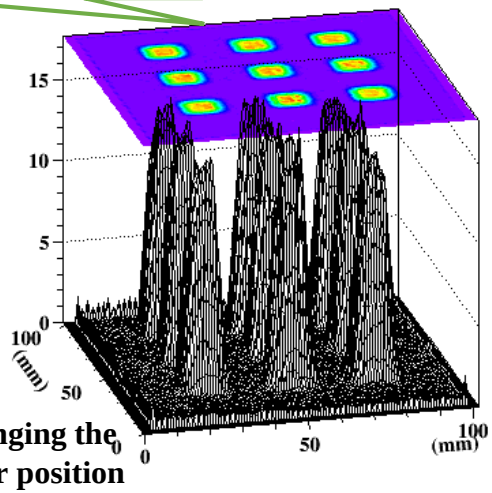
Direct connection by a network cable

- Gas flow radiation detector that can measure charged particles from a n(^{10}B, α)^{7}Li nuclear reaction
- Two Gas Electron Multipliers for signal amplification
- Thermal neutron efficiency: 0.5%~5%(depending on ^{10}B layer thickness)
  Data taking rate: Over 1 MHz (limited by Gigabit Ethernet)
  Minimum time step: 5 ns
- Position resolution: ~0.85 mm (FWHM)
- Operation voltage: ~2.7 kV (negative)
- Chamber gas: Ar/CO_2 (7:3)
- Active area: 100 mm × 100 mm
- Readout channels: 120 ch × 120 ch with a 0.8 mm pitch

Standard deviation of the counting rate:~0.85%
(^{10}B 0.1 μm thickness)



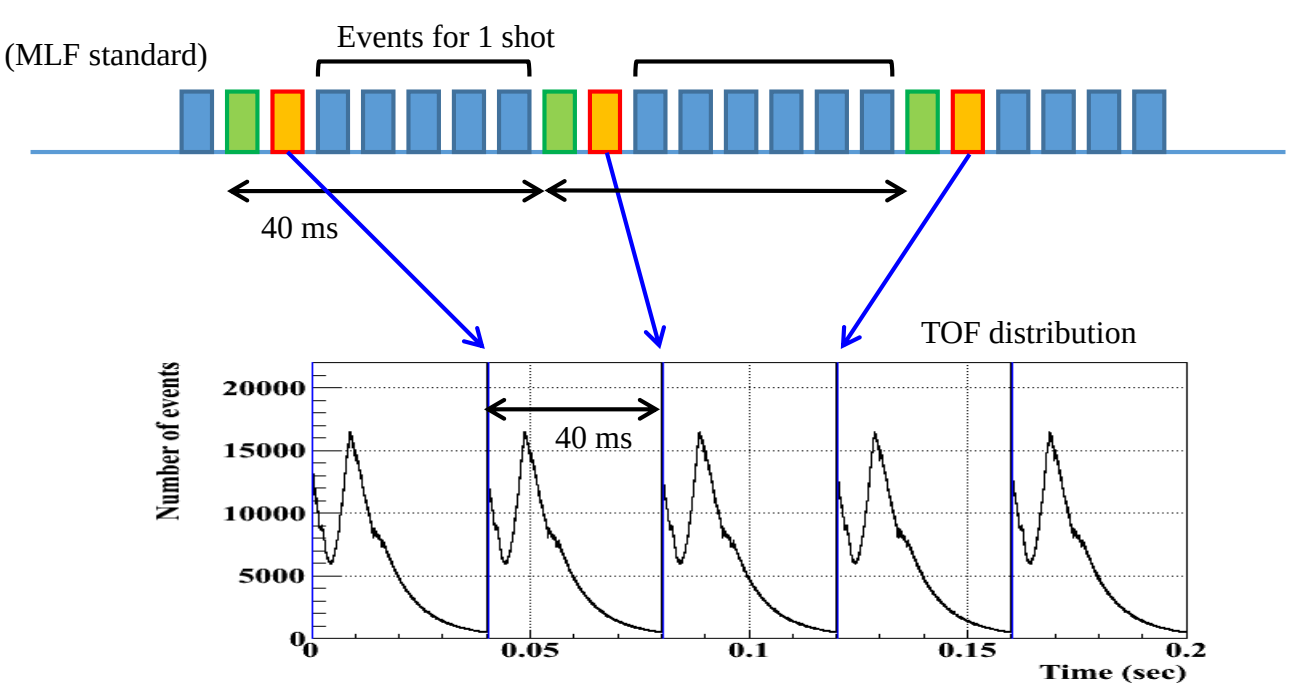


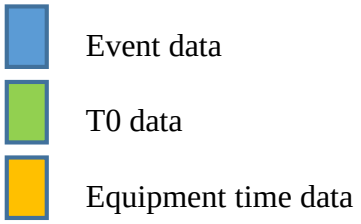


By changing the detector position

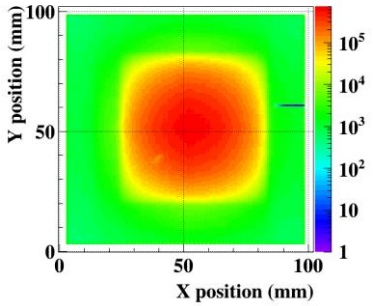
MLFで取得されるデータ

- 中性子ヒットに相当するEvent dataと陽子ビーム衝突タイミングをあらわすT0 data、Equipment time dataから構成

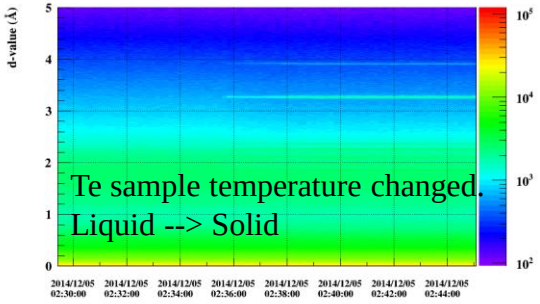




Neutron beam profile



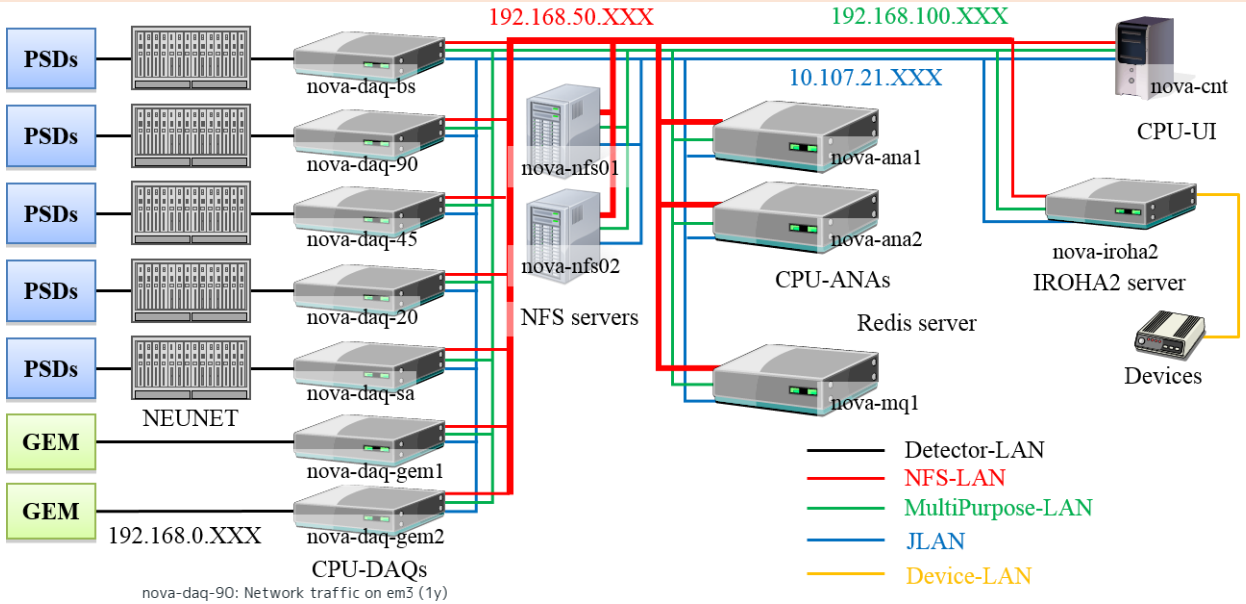
Time slice measurement

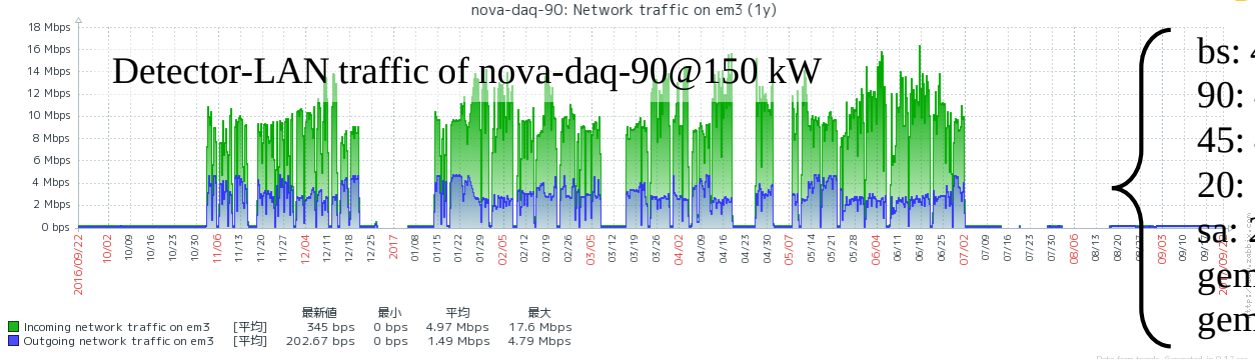


MLFではTOFとヒット位置をあらわすEvent dataと
測定日時をあらわすEquipment time dataから構成される
Equipment time dataによって、時分割解析が可能となる
ファイルサイズが1 GBを越えるとファイル名を変更して保存する

NOVAの計算機ネットワーク環境

- ビーム停止期間中に計算機ネットワーク環境を更新
  - ✓ 老朽化したNFS計算機の更新
  - ✓ データ収集と保存、解析用のネットワーク帯域の増強（10 Gbpsネットワーク化）





bs: 4.2 MB/s·MW (=363 GB/day·MW),
90: 4.2 MB/s·MW (=363 GB/day·MW),
45: 4.7 MB/s·MW (=406 GB/day·MW),
20: 3.5 MB/s·MW (=302 GB/day·MW),
sa: 2.0 MB/s·MW (=173 GB/day·MW),
gem1: 2.1 MB/s·MW (=181 GB/day·MW),
gem2: (2.1) MB/s·MW (=181 GB/day·MW).

ビーム再開後、NFS計算機においてデータ収集による書き込み（23 MB/s·MW）と（オフライン）解析による読み込み（500 MB/s）が発生

オフラインにおけるデータリダクションソフトウェア

- 大量のイベントデータを処理するため、マルチスレッド処理、マルチタスク処理の導入を検討
  ⇒プログラムはC/C++ベースでROOT-6を利用
- データ処理速度のリミットの原因を検討

DELL Precision M4700
CPU: Intel® Core™ i7-3940XM 3.00 GHz
Thread: 8 (4 cores × 2)
Memory: 32 GB
HDD: SSD (SMB41) 512 GB

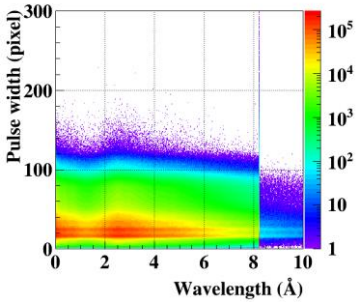
ディスクI/Oの評価：

```
$ sudo time dd if=/dev/zero of=/mnt/zero.txt bs=1M count=1000 oflag=direct
$ sudo sysctl -w vm.drop_caches=3
$ sudo time dd if=/mnt/zero.txt of=/dev/null bs=1M count=1000
```

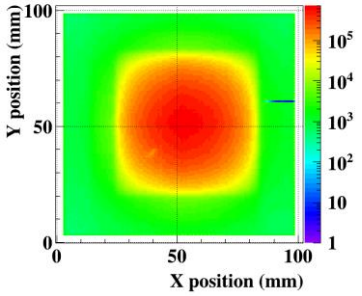
データ書き出し：(333.0±9.1) MB/s、
データ読み込み：(531.0±13.3) MB/s.

- シングルスレッドプログラム
  - ファイルI/O
  - メモリマップトI/O
- マルチスレッドプログラム
  - pthread
  - TThread
- マルチタスクプログラム

TH2D:1000 bin × 1000 bin



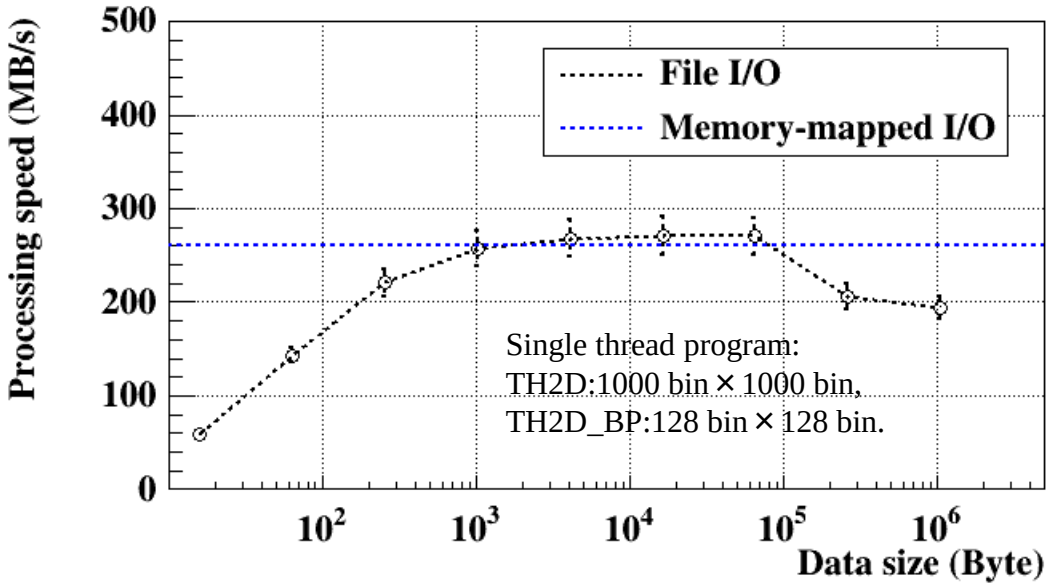
TH2D_BP:128 bin × 128 bin



nGEMで取得された16.487 GB（16ファイルに分割）を用いて
効率の良いデータリダクションソフトの開発をおこなった

ファイルI/Oとメモリマップト I/O（1）

- メモリI/Oはメモリ空間上にファイルのアドレスをマッピングすることで、簡便なデータアクセスを実現
  ⇒1 GBのファイルを一度にマッピングすることも可能
- ランダムリードを多用する場合において威力を発揮




```
#include <sys/mman.h>
...
int fd;
fd =open("file_name", O_RDONLY);

unsigned char *map_buffer
    =new unsigned char[lseek(fd, 0, SEEK_END)];
map_buffer
    =(unsigned char*)mmap(0, lseek(fd, 0, SEEK_END),
    PROT_READ, MAP_SHARED, fd, 0);

for(long i =0;i<(file_size);i++){
    map_buffer[i];
}

long page_size =getpagesize();
long map_size =((file_size)/page_size +1)*page_size;
msync(map_buffer, map_size, MS_SYNC);
munmap(map_buffer, map_size);
```

シングルスレッドプログラムによるデータ処理速度はディスクI/Oに達していない
⇒CPUバウンドの状態であるので、並列化処理を適用する
File I/Oはバッファサイズの最適化が必要である
実用的にはメモリマップトI/Oに分がありそうだが、、

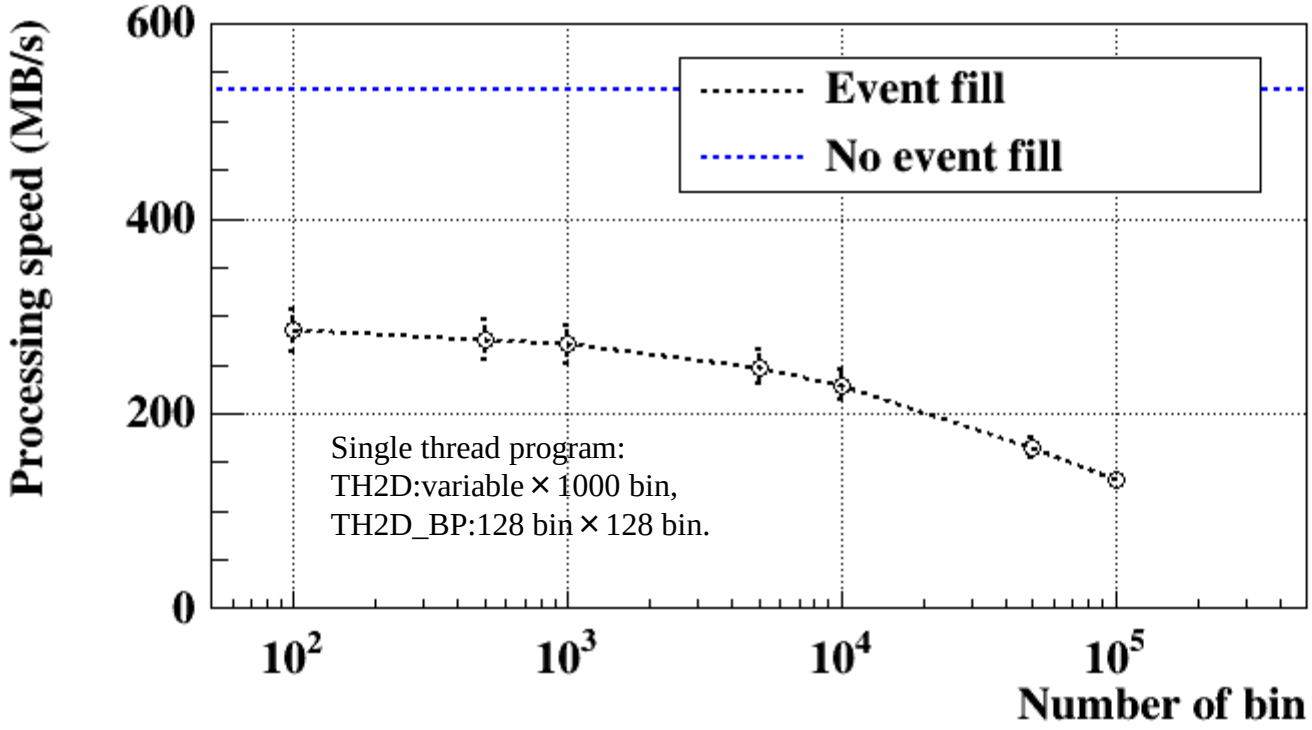
ファイルI/Oとメモリマップト I/O（2）

```
***** DAQ begins ...
*** Error in `./EventDecoderGEM’: munmap_chunk(): invalid pointer: 0x000000000228f770 ***
*** Error in `./EventDecoderGEM': malloc(): memory corruption: 0x000000000228f800 ***

 *** Break *** segmentation violation



==========================================================
There was a crash.
This is the entire stack trace of all threads:
==========================================================

Thread 7 (Thread 0x7f7eb40e7700 (LWP 29474)):
#0  0x00007f7eccffce2d in poll () from /lib64/libc.so.6
#1  0x00007f7ecda42a7b in master_thread_run (thread_func_param=0x22be690) at /home/daq/root-
6.08.06/net/http/civetweb/civetweb.c:12647
#2  master_thread (thread_func_param=0x22be690) at /home/daq/root-
6.08.06/net/http/civetweb/civetweb.c:12714
#3  0x00007f7ecd2d8dc5 in start_thread () from /lib64/libpthread.so.0
#4  0x00007f7ecd00776d in clone () from /lib64/libc.so.6
...
```

繰り返しプログラムを走らせるとSegmentation violation …
⇒　ファイルI/Oを最適化して利用

ヒストグラムのビン数依存性

- データ処理速度はヒストグラムのビン数に依存
  ⇒TOF分布のビン数を変更して、依存性を確認



データ処理速度はヒストグラムのビン数を小さくすることで改善するが、
最適なビン数は統計量や実験からの要請で決まる
⇒改めてビン数はTH2D:1000 bin × 1000 bin, TH2D_BP:128 bin × 128 binとした
Fill()をコールしない場合、ディスクI/Oと同程度のデータ処理速度が得られる

マルチスレッドプログラミング

- データデコードとヒストグラムFillingを並列に処理
  ⇒初期化、ヒストグラムマージ処理、描画、保存は単独に処理
- POSIX準拠のpthreadとROOTのTThreadを比較

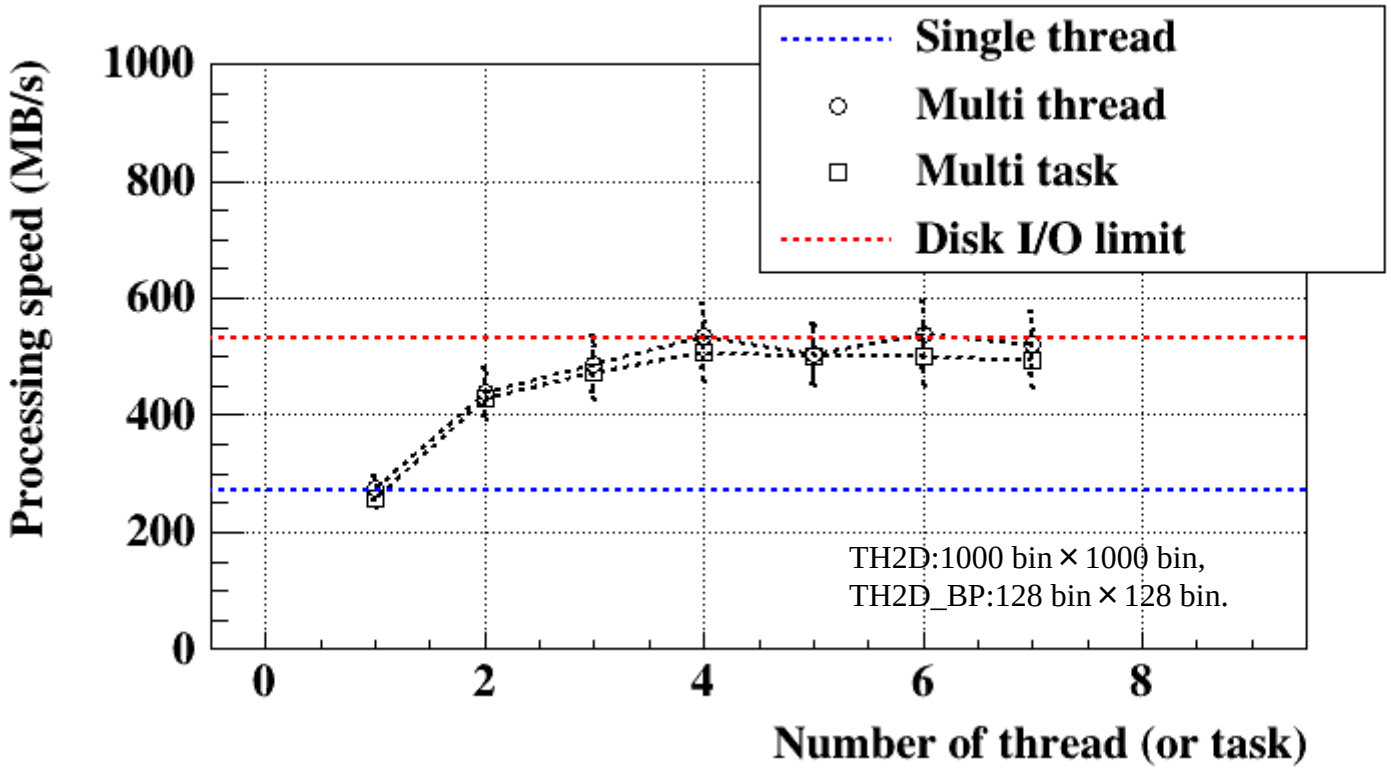
```
#include <pthread.h>
#define THREAD_NUM 4

typedef struct _thread_arg_t {
...
} thread_arg_t;
...
int th_num =THREAD_NUM;

pthread_t handle[th_num];
thread_arg_t targ[th_num];

for(int i=0;i <th_num;i++){
    ***** 構造体targ[i]に値を入力
}

for(int i=0;i <th_num;i++){
    pthread_create(&handle[i], NULL, Thread_function,
    (void *)&targ[i]);
}

for(int i=0;i <th_num;i++){
    pthread_join(handle[i], NULL);
}
```

```
#include <TThread.h>
#define THREAD_NUM 4

typedef struct _thread_arg_t {
...
} thread_arg_t;

typedef struct _th_set_t {
    int th_num;
    TThread *th[THREAD_NUM];
    bool f_flag;
} th_set;
...
int th_num =THREAD_NUM;

TThread *t[THREAD_NUM], *threadj;
th_set th_param;
thread_arg_t targ[th_num];

for(int i=0;i <th_num;i++){
    ***** 構造体targ[i]に値を入力
}

for(int i=0;i <th_num;i++){
    t[i] =new TThread("thread_name", Thread_function, (void *)&targ[i]);
    t[i]->Run();
}

th_param.f_flag =false;
for(int i=0;i <th_num;i++){
    th_param.th[i] =t[i];
}
th_param.th_num =th_num;
threadj =new TThread("tX", Join_function, (void *)&th_param);
threadj->Run();

while(th_param.f_flag ==false){
    gSystem->ProcessEvents();
}
threadj->Join();
```

上記pthreadのプルグラムは子プロセスが生成されたか確認していないため、
join()コールで沈黙する可能性がある⇒TThreadを採用

マルチスレッド処理とマルチタスク処理

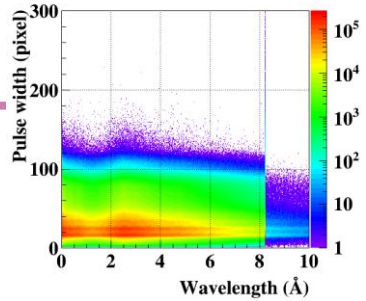
- マルチタスク処理では、C/C++ベースのシングルスレッドプログラムとヒストグラムマージ処理プログラムをPython上で制御
  ⇒全ての子タスクが終了したことを確認して、ヒストグラムマージ処理を実行



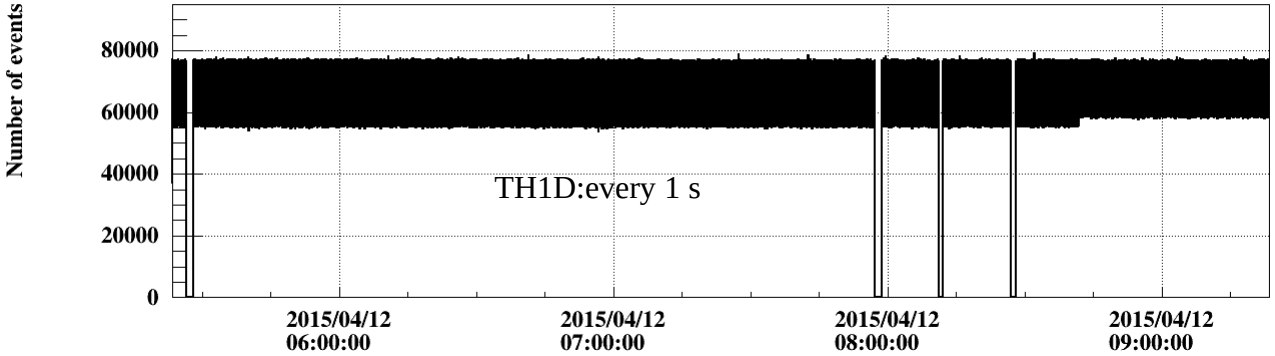
4スレッド以上のマルチスレッド処理では、データ処理速度がディスクI/Oによって制限されている
⇒(535.5±55.7) MB/s@4スレッド
マルチタスク処理はプロセス管理などのオーバーヘッドのために、マルチスレッド処理に比べて少し劣る

時系列プロットの表示

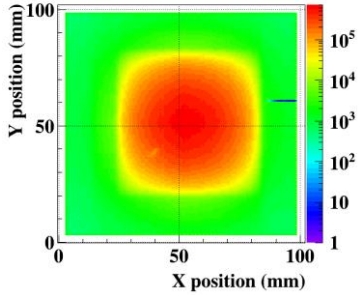
TH2D:1000 bin × 1000 bin



- 時系列プロットを作成、表示した場合のデータ処理速度を確認



TH2D_BP:128 bin × 128 bin



	Single thread	4-thread	8-thread
TH2D, TH2D_BP	(60.8±4.5) s	(30.8±3.2) s	(30.7±3.2) s
TH2D, TH2D_BP, TH1D	(81.4±5.2) s	(33.0±3.3) s	(30.9±3.2) s

時系列プロットのようなCPU負荷を増やした場合、データ処理速度は遅くなる
I/Oバウンドの状態では、CPUに余裕があるので、負荷を増やしてもデータ処理速度に影響しない
MLFで取得されるようなシンプルなデータ構造の場合、オフラインにおけるデータリダクション
ソフトウェアは適切な並列化処理をおこない、I/Oバウンドの状態で使用する

オフラインにおけるデータリダクションソフトウェアのまとめ

- 並列化処理を用いたオフライン解析システムの開発を実施
- ファイルI/O+TThreadによるマルチスレッド処理
- 並列度を上げることでI/Oバウンドで解析システムを運用

- 今後、高性能な計算機を購入し、より高度な解析処理を
  短時間で完了するシステムを構築
  ⇒現状、リーズナブルな計算機による処理速度 : ～500 MB/s

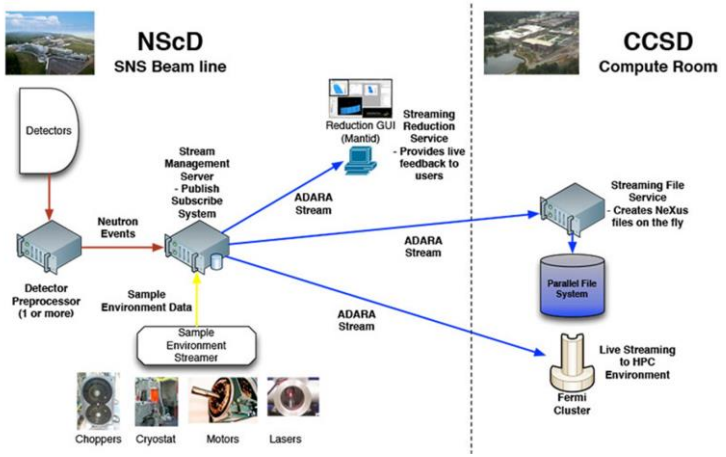
足りてる？

加速器ビームパワー1 MWの環境下では、～2000 GB/day

オンライン解析しかない　だろ

オンライン解析システムの必要性

- オンラインモニターを含むオンライン解析システムは効率の良い実験をおこなうための基盤技術
  - ✓ オンラインモニター：実験状況の把握、統計量の確認
  - ✓ オンライン解析：測定中における$S(Q)$、$g(r)$などの物理量の導出、データリダクション
- SNSではADARAと呼ばれるオンライン解析環境を整備
  - ✓ message queuing（MQ）の1つであるActiveMQを利用
- ESSではMQとしてKafkaを利用して、オンライン解析環境を構築する予定
- MLFにおいても1年前からRedisを用いたオンライン解析システムの開発に着手



ADARA, http://www.csm.ornl.gov/newsite/adara.html.
ActiveMQ, http://activemq.apache.org.
Kafka, https://kafka.apache.org.
redis, http://redis.io.

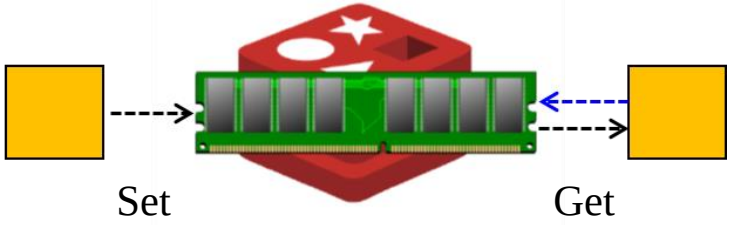
Duty	Contact person
Development of DAQ-MW component for redis, technical review	Y. Yasu
Development of Manyo-Utsusemi for redis	Y. Inamura
Installation and commissioning at BL17	S. Kasai
Installation and commissioning at BL21 and BL06	T. Seya, H. Ohshita

大強度パルス中性子源施設ではMQを利用してオンライン解析システムを構築

redis（レディス）

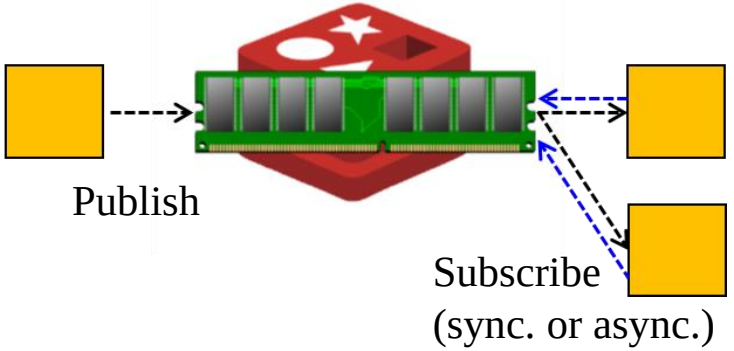
- redis（レディス）
  - ✓ MQを実現するためのソフトウェア
  - ✓ Key-Value Store（KVS）をメモリ上のredisサーバーで実現
  - ⇒ 高速動作するシステムの構築が可能
  - ✓ KVSとしてstring型、list型など様々なデータ構造の取り扱いが可能
  - ✓ オープンソースプログラム（BSD-licensed program）
  - ✓ C/C++、Pythonなど多くのプログラミング言語をサポート
  - ✓ Redisサーバーは単純なqueueサーバーとしてだけでなく、brokerとしての利用も可能
  - ✓ マルチスレッドプログラムには非対応

redis server as a simple queue server
(Queue model)



redis server as a broker
(Pub/Sub model)

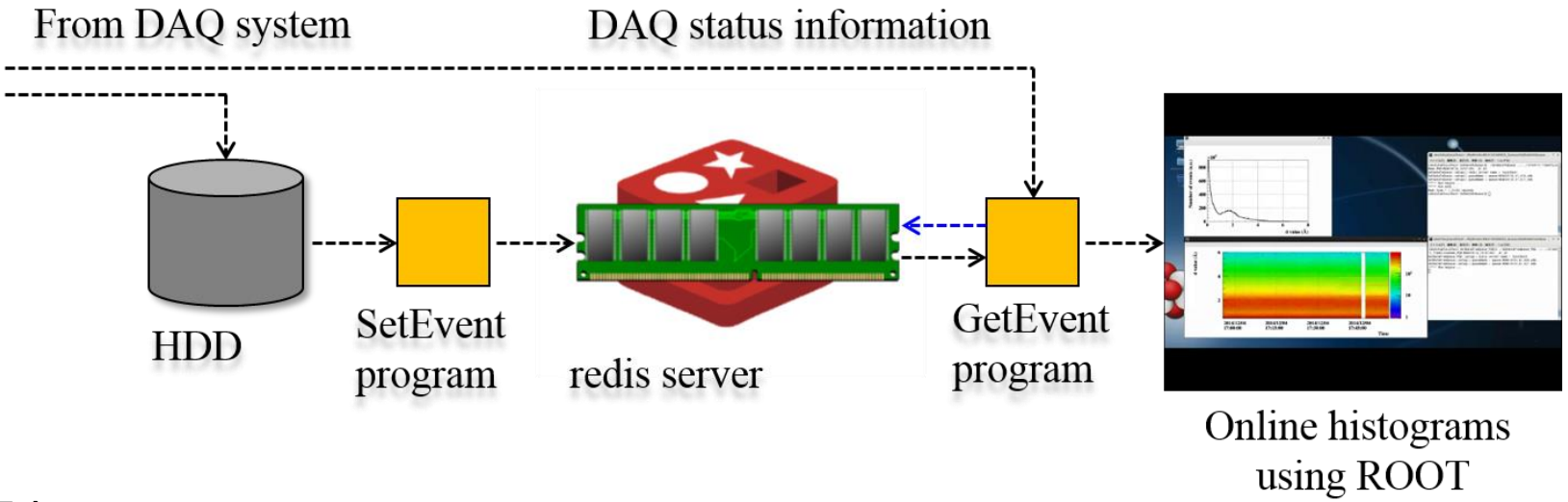
出版・購読モデル



redis Pub/Subモデルは、1：Nのシステムの構築が可能
さらに、メモリ消費を抑える使い方（非同期モード）も可能

これまでのオンライン解析システム

- 書き込み（測定）中のファイルにアクセスし、追加されたデータを処理するシステム
- redis Queueモデルを採用



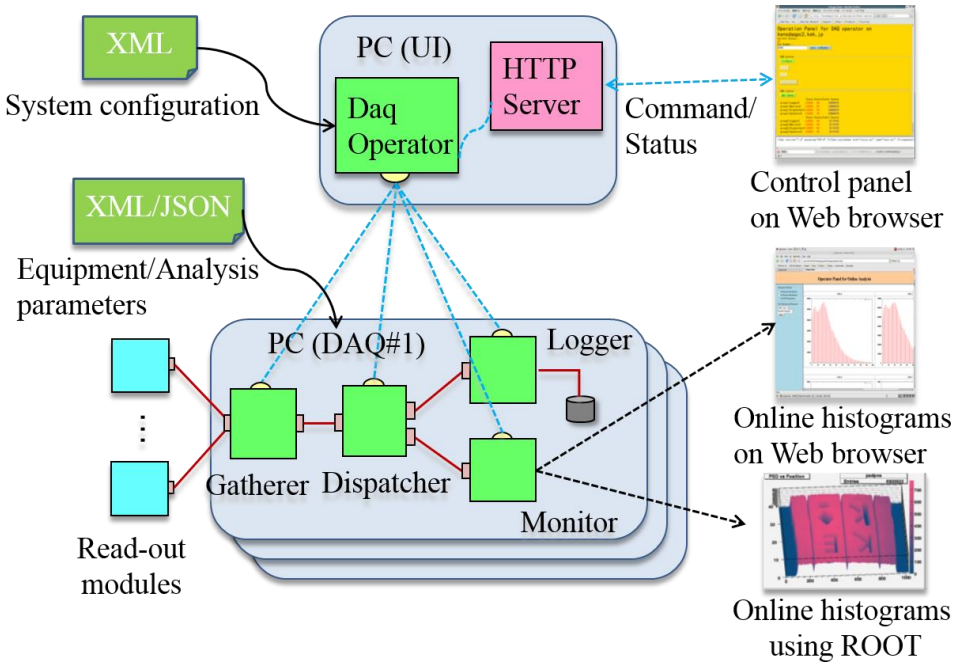
問題点：

① 書き込み処理と読み込み処理の併用は著しく性能を劣化させる可能性
② redis Queueモデルを採用しているため、複数のGetEventプログラムにデータを渡すことは不可
③ redis serverからデータを取り出さないとメモリの使用量が増加
④ 何らかの方法で実験ステータス（Begin、End）を伝達することが必要

解決方法の1つは、既存DAQシステムへの組み込み

DAQ-Middleware（DAQ-MW）

- ネットワーク分散環境でDAQシステムを容易に構築するためのソフトウェア
- RT-Middleware [*]をベースに作成
- Gatherer、Dispatcher、Logger、Monitorなど機能を単位とするコンポーネント群から構成
- DAQシステムの構成はXMLファイルに記述することで実現
  検出器パラメーターなどもXMLファイルに記述し反映させることが可能



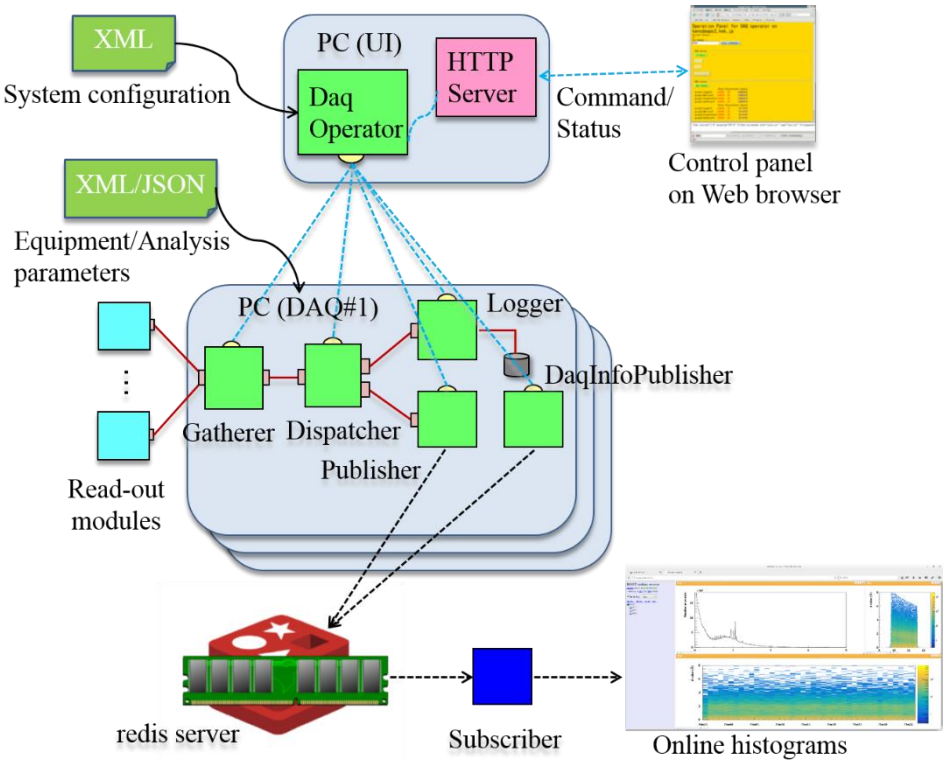
ネットワーク環境との親和性、拡張性、柔軟性

[*] Object Management Group, Specification of Robotic Technology Component (RTC), Version 1.0, http://wwwomg.org/spec/RTC/1.0/.
Web page of the DAQ-MW core team, http://daqmw.kek.jp.

新しいオンライン解析システム

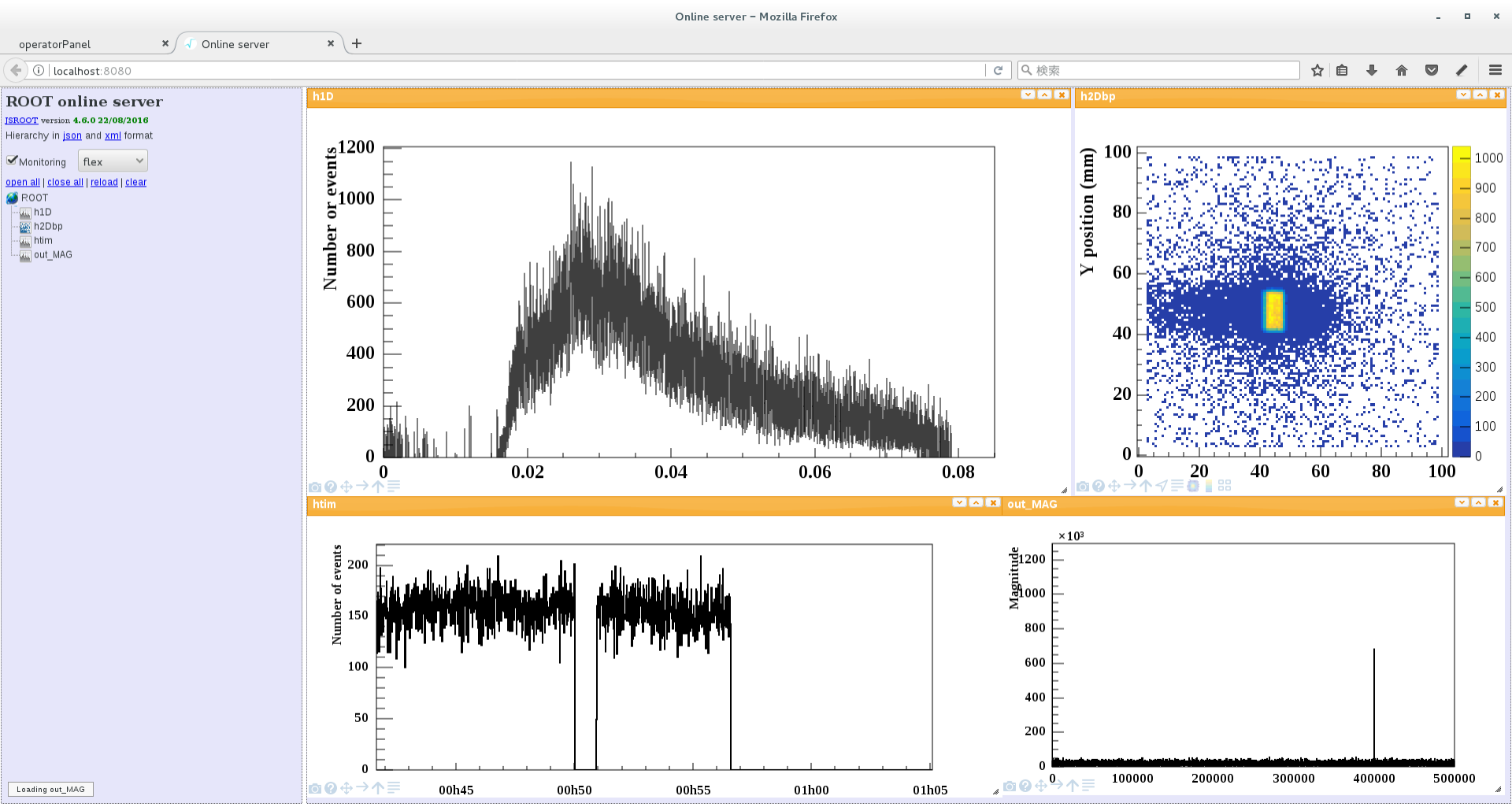
- redis Pub/Subモデルによる疎結合したオンライン解析システムを実現
- Publisherコンポーネント、DaqInfoPublisherコンポーネントの開発
  - ✓ Redisサーバーをバッファとして活用できるため、大量の演算処理が可能
  - ✓ Begin、Endなどの実験ステータスはDaqInfoPublisherコンポーネントから取得
- Subscriberプログラムはユーザーによる自由なカスタマイズが可能
  - ✓ JSROOT [**]を使用して、リモート計算機によるWebベースのオンラインモニターの実現

[**] Web page of JSROOT, https://root.cern.ch/js/.



既存コンポーネント群も修正し、バージョンアップされたDAQ-MWとして公開予定

適用例



JSROOTでは表示するヒストグラムなどの画像保存、表示の変更、
統計情報の読み取りが可能

オンライン解析システムのまとめ

- redisを用いたオンライン解析システムの開発を実施
- DAQ-MW + redis Pub/Subモデルで新しいシステムを構築

- ビーム再開後に各BLへ導入し、コミッショニングを開始
  - ✓ 10月下旬～　先行BLへ導入、コミッショニング開始
  - ✓ 2018年3月　　DAQ-MW最新版リリース（前倒しの可能性有）
  - ✓ 2018年4月　　他BLにおける導入サポート開始

Thanks for your attention !

菖蒲色（あやめいろ）

- 溶融点、凝固点近傍の観察（試料はテルル（Te））

⇒温度データとの比較は新潟大グループにより実施、過冷却現象が観察できる