MLF 中性子用
DAQ ミドルウェア
インストールおよび操作マニュアル

千代浩司　　仲吉一男　　安 芳次

高エネルギー加速器研究機構
素粒子原子核研究所

$Date: 2009/08/04 08:01:39 $

目次

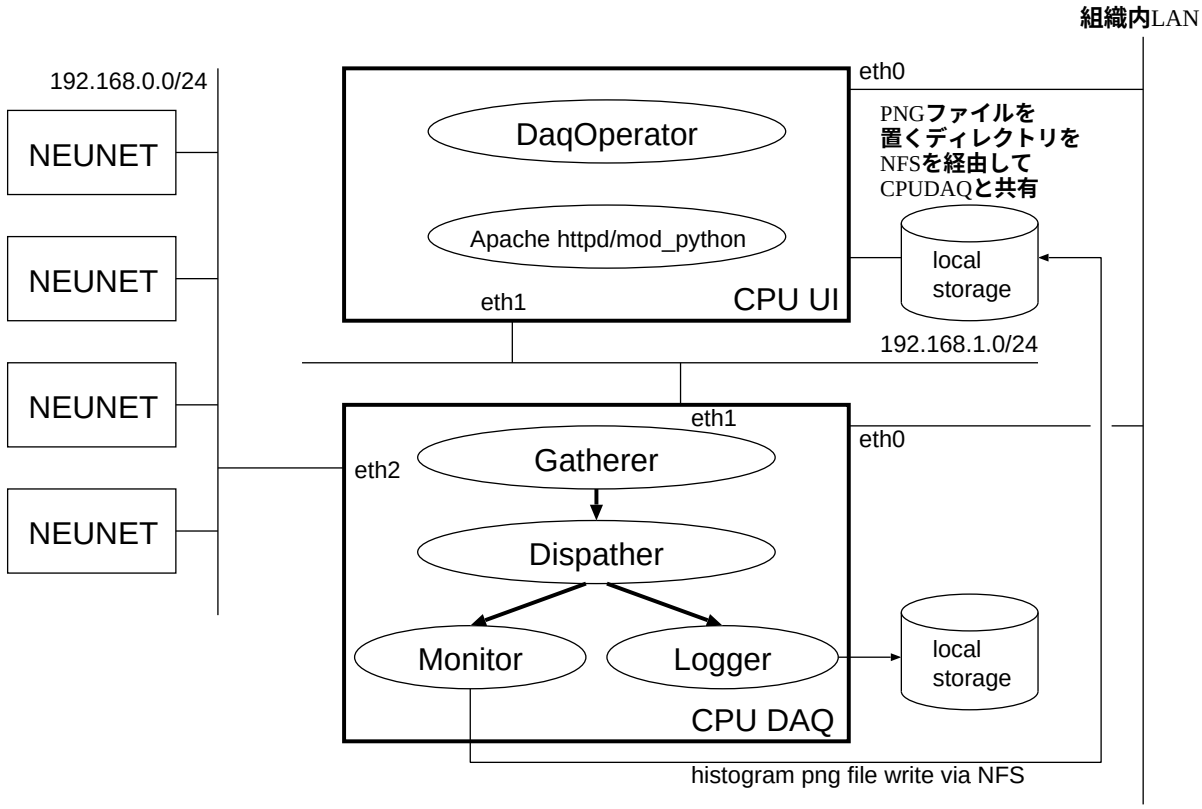

図 1　このマニュアルで解説する DAQ ミドルウェア配備モデル。2 台の PC を使いデータ収集を行う場合を例にとりインストール、設定の解説を行います。

1　はじめに

この文書では DAQ ミドルウェアのインストール方法、および操作操作方法を解説します。

1.1　このマニュアルでの DAQ ミドルウェア配備モデル

DAQ ミドルウェアの配置方法はいろいろありますが、このマニュアルでは図 1 に示したように 2 台の PC を使用してデータ取得を行う場合をモデルケースとしてソフトウェアのセットアップ方法を説明します (1 台の PC 上に全てのコンポーネント等をセットアップし動作させることも可能です)。データ発生レートが高い、あるいは読み取るべきデータが発生する機器が多い場合など 1 台の PC では読み取りをまかないきれない場合は CPU DAQ を増やして対応します。

このモデルでは各 PC のネットワークインターフェイスを次のように割り当てます。

eth0　**組織内 LAN に割り当てて、組織内の他の PC から CPU UI、CPU DAQ にアクセスする際に使用するインターフェイス。**

PC	説明	ディレクトリあるいはパス名
CPU UI	Apache DocumentRoot	/var/www/html/
CPU UI	Apache Daq Page	/var/www/html/daq/ → /home/daq/www/
CPU UI	mod_python 設定ファイル	/etc/httpd/conf.d/daq.conf
CPU UI	ヒストグラム png ファイル群	/nfs/data/png/
CPU UI	daq ユーザーのホームディレクトリ	/home/daq
CPU DAQ	daq ユーザーのホームディレクトリ	/home/daq
CPU DAQ	各コンポーネント実行ファイル	/home/daq/DaqComponents/bin/
CPU DAQ	イベントデータ保存ディレクトリ	/home/daq/edata/
CPU DAQ	gnuplot	/home/daq/gnuplot/bin/gnuplot
CPU DAQ	Manyo library	/home/daq/manyo/

表 1 このマニュアルで解説するディレクトリ、ファイルパス名。Apache Daq Page についてはファイルの実体は/home/daq/www/以下にあるが apache httpd がそれらのファイル群にアクセスできるようにするためにシンボリックリンクを張る (後述解説参照)。

eth1 DAQ コンポーネント間のデータ通信、および CPU DAQ から CPU UI にある共有ストレージへヒストグラムを書く際の NFS トラフィックが流れるインターフェイス。

eth2 (CPU DAQ のみ)
CPU DAQ が NEUNET からデータ読み取りを行うインターフェイス。

図中 CPU UI 上では DaqOperator コンポーネント、および mod_python が有効になった Apache httpd サーバーが起動します。ユーザーは適当なウェブブラウザを使ってこの httpd サーバーにアクセスし、start、stop 等のコマンドを発行したり、モニター画面を見たりします。

実際のデータ取得は図中 CPU DAQ という計算機で行われます。Gatherer コンポーネントは左側 192.168.0.0/24 のネットワークに接続された NEUNET モジュールで発生したデータの読み取りを行います。Gatherer コンポーネントで取得したデータは Dispatcher コンポーネントに送られ、そこで 2 つのコンポーネント (Logger コンポーネントおよび Monitor コンポーネント) にデータを送ります。Logger コンポーネントは送られてきたデータをローカルストレージに保存します。モニターコンポーネントは定期的に各ヒストグラムファイルを作成し NFS を経由して CPU UI に接続されたストレージに保存します。各ヒストグラムは httpd サーバーを通じてユーザーのウェブブラウザに送られます。

このマニュアルで解説するディレクトリ構成を表 1 に示します。続いて、ソフトウェアのインストール、設定方法について解説します。

2 DAQ ミドルウェアインストール

ソフトウェアのインストール、設定は

- OS の設定
- DAQ ミドルウェア依存物のインストール
- DAQ ミドルウェアのインストール
- DAQ ミドルウェアの設定

の順で行います。DAQ ミドルウェアは RedHat Enterprise Linux 5(RHEL 5) 上で開発、テストされています。ここでは RHEL 5、あるいはそのクローン OS(Scientific Linux 5 など) 上で DAQ ミドルウェアを稼働させるとして解説を行います。

2.1 OS の設定

この節では CPU UI、CPU DAQ ともに必要となる OS の設定について解説します。

2.1.1 RHEL 5 が提供する rpm パッケージ

以下のソフトウェアパッケージは RHEL 5 で提供されています。インストール方法によってはインストールされていないことがあるので確認しておいてください。インストールされていなければ RHEL 5 のディストリビューションパッケージから `rpm` コマンドなどを使ってインストールしておいてください。

- xinetd
- pytnon-devel
- gd
- libpng

2.1.2 SELinux が無効になっているかどうかの確認

SELinux (Security Enhanced Linux) が enabled になっていると mod_python の動作等に支障があるため SELinux は disabled になっている必要があります。現在の SELinux の状態がどうなっているか確認するには `selinuxenabled` コマンドを実行してその終了ステータスを確認してください。

```
# /usr/sbin/selinuxenabled; echo $?
```

`echo $?` の結果が 1 であれば SELinux は disabled になっています。0 であれば SELinux は enabled になっているので次のように `/etc/sysconfig/selinux` を編集し計算機をリブートして

ください。

```
SELINUX=disabled
```

2.1.3 ファイアウォールの無効化

ファイアウォールが設定されているとパケットフィルターが実行され、データ転送レートが落ちる場合がありますのでファイアウォールを無効にしてください。具体的には

```
# /sbin/chkconfig iptables off
```

とすると再起動後にファイアウォールが無効になります。安全保障は他の手段で行うようにしてください。

2.1.4 ネットワークソケットバッファサイズの設定

RHEL 5 のデフォルトの設定ではネットワークソケットバッファが小さいためパフォーマンスが出ない場合があります。そこで /etc/rc.local に以下のように書いておきます。

```
#
# DAQ Socket Send/Receive Buffer Size Setting
#
echo "Loading DAQ Socket Send/Receive Buffer Size Setting"
echo 0                      > /proc/sys/net/ipv4/tcp_timestamps
echo 8388608                > /proc/sys/net/core/wmem_max
echo 8388608                > /proc/sys/net/core/rmem_max
echo 8388608                > /proc/sys/net/core/wmem_default
echo 8388608                > /proc/sys/net/core/rmem_default
echo '4096 4194304 4194304' > /proc/sys/net/ipv4/tcp_rmem
echo '4096 4194304 4194304' > /proc/sys/net/ipv4/tcp_wmem
```

このファイルのパーミッションは rwxr-xr-x である必要があります。確認には以下のコマンドを実行します。

```
% ls -lL /etc/rc.local
-rwxr-xr-x 1 root root 750 Jan  7  2009 /etc/rc.local*
```

リブート時には自動的にこの設定が実行されます。リブートせずに直ちに設定するには /etc/rc.local を実行してください。

```
# /etc/rc.local
```

2.1.5 ディスクパラメータの設定

ローカルハードディスクにデータを書く場合には、/proc/sys/vm/dirty_background_ratio を小さめの値にしておくと、ディスクキャッシュが頻繁にディスクに書かれるようになり性能が向

上します。デフォルト値は 10 になっています[*1]。どの値に変更するかは実際にテストしてみて確かめる必要があります。テストの結果、1 がよいということであれば /etc/rc.local に

```
echo 1 > /proc/sys/vm/dirty_background_ratio
```

と書いておきリブート後も自動的にこの値にセットされるようにしておきます。

2.1.6 daq ユーザーの作成

この文書では daq というユーザーの権限で各 DAQ コンポーネントが起動されるものとして解説します。CPU UI および CPU DAQ で daq ユーザーを作成してください。また daq ユーザーが属するグループは daq でありホームディレクトリは /home/daq であるとして解説します。以下で /home/daq は apache httpd が読み取るファイルを置くことになります。apache httpd は通常 nobody ユーザーとしてローカルファイルの読み取りを行います。したがって /home/daq ディレクトリのパーミッションは other permssion が r-x である必要があります。RHEL 5 の GUI でユーザー登録した場合には /home/daq のパーミッションが rwx------ である場合が多いので、こうなっていた場合は root ユーザーで

```
root# chmod 755 /home/daq
```

としてください。

複数の PC を使ってデータ収集を行う場合は daq ユーザーのユーザー ID は統一しておいたほうが便利です。特に、NFS 経由でファイルの読み書きを行う場合にはユーザー ID を各 PC で同一にセットしておく必要があります。

2.1.7 CPU UI での NFS export の設定

CPU UI 上で CPU DAQ で稼働するモニターコンポーネントがヒストグラムを書くディレクトリとして /nfs/data ディレクトリを作成し、それを NFS export します。

```
root@cpuui# mkdir -p /nfs/data
root@cpuui# chown daq:daq /nfs/data
root@cpuui# /etc/exports を編集して以下の行を加える
/nfs/data     192.168.1.0/24(rw,sync,no_root_squash)
root@cpuui# /sbin/chkconfig nfs on
root@cpuui# /etc/init.d/nfs restart
```

restart としているので、nfs export が有効になっていない場合は stop の部分でエラーが発生しますが、それは無視してかまいません。起動の部分で “OK” となることを確認してください。

[*1] この値は総メモリに対するパーセントで、まだディスクに書いていないディスクキャッシュ (ダーティキャッシュ) が総メモリーの dirty_background_ratio パーセントになったらディスクに書き出すという動作をします。

2.1.8 CPU DAQ での NFS マウント

CPU UI から export されたディレクトリをマウントできるかどうか次のコマンドで確認します (CPU UI の eth1 の IP アドレスを 192.168.1.10 と仮定しています)。

```
root@cpudaq# mkdir -p /nfs/data
root@cpudaq# chown daq:daq /nfs/data
root@cpudaq# mount -t nfs -o intr 192.168.1.10:/nfs/data /nfs/data
```

リブート時に自動でマウントされるようにするには /etc/fstab を編集して以下の行を追加します。

```
192.168.1.10:/nfs/data /nfs/data   nfs     defaults,intr   0 0
```

NFS マウントするのに autofs を使う手段もありますが、ここでは解説を省略します。

2.1.9 時刻の調整

GATENET コンポーネントを使用しデータ取得を行う場合、daq_configure でのパラメータ設定時に、GATENET コンポーネントが GATENET の時刻合わせを行います。GATENET コンポーネントを起動する計算機の時刻が GATENET のレジスタへ書き込まれます。装置時刻データ中の時刻情報は GATENET モジュールが保持している時刻情報を元にしているため、GATENET モジュールの時刻があっていないと装置時刻データ中の時刻情報も正しくない値となります。したがって装置時刻情報に正しい時刻情報を入れるためには GATENET コンポーネントを動かす計算機の時刻が正しくセットされていることが必要です。計算機の時刻を常に正しい値にしておくためには ntp を使用します。ntp パッケージは OS の配布物にありますのでインストールされていない場合は yum 等を使ってインストールし、組織内の ntp サーバーと時刻同期を行うように設定します。ntp サーバーの設定は ntp パッケージをインストール後、/usr/bin/system-config-date を実行し、Network Time Protocol タブで Add ボタンを押して ntp サーバーを指定し Enable Network Time Protocol のチェックボックスをチェックすることで行います。利用可能な ntp サーバーについては組織内のしかるべき部署に確認してください。

2.2 DAQ ミドルウェア依存物のインストール

DAQ ミドルウェアの動作に必要なソフトウェアのインストールについて

- CPU UI、CPU DAQ 両方で必要になるもの
- CPU UI でのみ必要になるもの
- CPU DAQ でのみ必要になるもの

の順で解説します。

${YumServerURL}	http://www-jlc.kek.jp/%7Esendai/OpenRTM/EL5/
${RepoConfFile}	kek-daqmiddleware-repo-1-3.el5.noarch.rpm
rpm ディレクトリ	http://www-jlc.kek.jp/%7Esendai/OpenRTM/EL5/i386/
tars ディレクトリ	http://www-jlc.kek.jp/%7Esendai/OpenRTM/EL5/tars.2009.07/

表 2 配布物 URL 一覧

2.2.1 CPU UI、CPU DAQ ともに必要となる依存物のインストール

CPU UI および CPU DAQ の両者で DAQ ミドルウェアの動作に必要になるソフトウェアは以下のとおりです。

- ACE
- OmniORB
- OpenRTM-aist KEK 版
- DAQ ミドルウェアソケットライブラリ

DAQ ミドルウェアソケットライブラリ以外は rpm ファイルが用意されており、yum コマンドを使ってインストールができるようになっています。ネットワークに接続されており yum サーバーにアクセスできる環境では以下のコマンドを実行すると上記 3 つのソフトウェアパッケージのインストールが完了します。${YumServerURL} および、${RepoConfFile} については表 2 を見て適切な値を代入してください。

```
# rpm -Uhv ${YumServerURL}/noarch/${RepoConfFile}
# yum --enablerepo=kek-daqmiddleware install OpenRTM-aist
(Yes/No を聞かれるので y と入力する)
```

最初の rpm コマンドで /etc/yum.repos.d/ 以下に yum サーバーの位置を指定する設定ファイルがインストールされます。2 番目の yum コマンドで先に述べた ACE、OmniNames、OpenRTM-aist のインストールが完了します。

omniORB は計算機を再起動後 omniNames が自動起動されるようにセットされているのでこれを止めるために次のコマンドを実行してください。これは次に述べる手動で rpm コマンドを使ってインストールした場合にも同様に行う必要があります。

```
root# /sbin/chkconfig omniNames off
```

yum コマンドが使えない場合等の場合には表 2 に示した rpm ディレクトリから以下の rpm ファイルを取得して手動でインストールしてください。

- ACE-5.6-4.DTP.el5.i386.rpm
- ACE-devel-5.6-4.DTP.el5.i386.rpm

- OpenRTM-aist-0.4.1-9.KEK.el5.i386.rpm
- omniORB-4.0.7-3.el5.i386.rpm
- omniORB-bootscripts-4.0.7-3.el5.i386.rpm
- omniORB-devel-4.0.7-3.el5.i386.rpm
- omniORB-doc-4.0.7-3.el5.i386.rpm
- omniORB-servers-4.0.7-3.el5.i386.rpm
- omniORB-utils-4.0.7-3.el5.i386.rpm

一つのディレクトリに上の全ての rpm ファイルをまとめておくと

```
# rpm -ihv OpenRTM-aist-0.4.1-9.KEK.el5.i386.rpm
```

で一括してインストールすることができます。この方法でインストールした場合も上で述べたように OmniNames を off にするのを忘れないでください。

DAQ ミドルウェアの動作に必要なライブラリについては表 2 の tars ディレクトリに示した場所の lib.YYYY.MM.tar.gz にコンパイル済みのバイナリファイルがありますのでこれを取得し /home/daq で展開しておきます。

```
daq% tar zxvf /path/to/lib.YYYY.MM.tar.gz
```

以上で CPU UI および CPU DAQ で共通に使用する依存物のインストールは完了です。

2.2.2 CPU UI で必要になる依存物のインストール

この節では CPU UI で必要になる依存物のインストールの解説をします。

CPU UI で必要になるソフトウェアは

- Apache httpd サーバー
- mod_python
- xerces-c

です。Apache httpd サーバーおよび mod_python は RHEL 5 の配布物にありますのでインストールされていない場合は RHEL 5 のディストリビューションメディアから rpm コマンド等を使用してインストールしてください。

xerces については RPM forge に rpm ファイルがあります。rpm ファイルのコピーを表 2 に示した yum リポジトリにおいてありますので

```
# yum --enablerepo=kek-daqmiddleware install xerces-c-devel
```

でインストールしてください。

設定方法については次節で解説します。

2.2.3 CPU DAQ で必要になる依存物のインストール

CPU DAQ で必要になるソフトウェアは

- xinetd
- gsl (GNU Scientific Library)
- mxml
- Xalan
- gnuplot
- Manyo Library

です。

このうち xinetd は RHEL 5 で提供されているのでインストールされていない場合は RHEL 5 のディストリビューションメディアから rpm コマンド等でインストールしてください。

gsl、mxml、Xalan については表 2 の rpm ディレクトリにあります。yum コマンドが利用可能な場合には

```
# yum --enablerepo=kek-daqmiddleware install gsl-devel
# yum --enablerepo=kek-daqmiddleware install mxml
# yum --enablerepo=kek-daqmiddleware install xalan-c-devel
```

でインストールが可能です。

gnuplot は RHEL 5 のディストリビューションに入っていますが機能が不足するため使えません。DAQ ミドルウェアで使用する gnuplot および Manyo library については rpm にはなっていませんが RHEL 5 上でコンパイルしたバイナリを tar コマンドでまとめたものを表 2 の tars ディレクトリに `gnuplot.bin.tar.gz` および `manyo.bin.tar.gz` として置いてありますので、これらを取得して `/home/daq` ディレクトリで展開してください。

```
daq% tar zxvf /path/to/gnuplot.bin.tar.gz
daq% tar zxvf /path/to/manyo.bin.tar.gz
```

この gnuplot は RHEL 5 の libpng パッケージと gd パッケージに含まれているライブラリを使用しているのでこれらのパッケージがインストールされていない場合は RHEL5 のディストリビューションメディアからインストールしておいてください。gnuplot について不足のシェアードライブラリがないかどうか確認するためには、フルパスで実際に gnuplot を起動することで確認できます。起動できたらライブラリの不足はありません。なお、起動したあと`Terminal type set to 'unknown'` と表示される場合がありますが、これは問題とはなりません。

以上で DAQ ミドルウェアが使用する依存物のインストールは完了です。

2.3 CPU DAQ への DAQ ミドルウェアの導入、および設定

この節では CPU DAQ への DAQ ミドルウェアの導入および設定について解説します。

表 2 の tars ディレクトリから DAQ コンポーネントのソース、および実行ファイルをまとめた tar ファイル DaqComponents.tar.gz を取得して /home/daq ディレクトリで展開します。

```
daq@cpudaq% tar zxvf /path/to/DaqComponents.tar.gz
```

これで DaqComponents ディレクトリができます。各コンポーネントの実行形式ファイルは DaqComponents/bin ディレクトリに入ります。コンパイルの方法は付録をごらんください。

次に xinetd から bootComps.py が起動するようにするために xinetd の設定を行います。ブートメカニズムについては付録 B を見てください。xinetd の設定ファイルは起動するサーバーごとに /etc/xinetd.d/ ディレクトリ以下に入れることになっています。また xinetd は起動する各サーバーがどのポートを使用するかの情報を /etc/services から取得します。したがって CPU DAQ 上の xinetd の設定は

1. xinetd パッケージがインストールされているかどうか確認。
2. /etc/services ファイルの編集。
3. /etc/xinetd.d/ 以下に必要なファイルを置く。置いたファイルを確認。
4. xinetd から起動するファイルの確認。
5. xinetd の再起動。

の順に行います。

まず xinetd パッケージがインストールされているかどうか確認してください。xinetd パッケージは RHEL 5 の配布物に含まれています。インストールされていない場合は RHEL 5 のディストリビューションメディアから rpm コマンド等を使ってインストールしてください。

次に /etc/services ファイルに以下の行を追加します。

```
bootComps   50000/tcp           # boot DAQ-Components
```

/etc/xinetd.d/ ディレクトリに bootComps 起動用設定ファイルを置きます。

```
root@cpudaq# cp /home/daq/DaqCompoments/bootComps-xinetd /etc/xinetd.d/bootComps
```

bootComps.py の場所の指定は /etc/xinetd.d/bootComps ファイル内の server= ディレクティブで行います。ファイルをみて server= で指定した場所に bootComps.py があるかどうかを確認してください。また bootComps.py ファイルには実行許可ビットが設定されている必要があります。実行許可ビットの設定は通常どおり chmod で行います。

```
root@cpudaq# chmod 755 /home/daq/DaqCompoments/bootComps.py
```

確認後、リブート時に自動起動するようにセットし、xinetd を再起動します。

```
root@cpudaq# /sbin/chkconfig xinetd on
root@cpudaq# /etc/init.d/xinetd stop
root@cpudaq# /etc/init.d/xinetd start
```

xinetd が起動していなかった場合には上の stop のコマンドでエラーがでますがこれは無視してかまいません。start でエラーが出ないことを確認してください。

bootComps.py が起動するかどうかの確認は他の計算機からポート番号を指定して nc コマンドを使って行います。CPU DAQ の IP アドレスが 192.168.1.11 である場合には以下のように入力して出力をみます。

```
root@cpuui# echo pwd | nc 192.168.1.11 50000
Bad command: pwd
```

上のように "Bad command: pwd" と出れば OK です。"Bad command" と出るのは bootComps.py が特定のコマンド文字列以外は全て "Bad command" として認識するのと "pwd" というコマンドは存在しないことによります。

上記のようにならなかった場合は iptables 等のパケットフィルターが有効になっていないかどうか、また xinetd の設定等を確認してください。確認には tcpdump コマンドでイーサネット上に流れているパケットを観察するのが有効です。port 50000 番に関連するパケットを tcpdump で見るには

```
# /usr/sbin/tcpdump -n -i eth1 port 50000
```

とします。複数のネットワークインターフェイスがある場合は上のように `-i` オプションで観察するインターフェイスを指定する必要があります。

2.4 CPU UI への DAQ ミドルウェアの導入

この節では CPU UI への DAQ ミドルウェアの導入および設定について解説します。

表 2 の tars ディレクトリから DAQ コンポーネントのソース、および実行ファイルをまとめた tar ファイル DaqComponents.tar.gz を取得して `/home/daq` ディレクトリで展開します。

```
daq@cpuui% tar zxvf /path/to/DaqComponents.tar.gz
```

CPU UI では展開してできた実行ファイルのうち bin/DaqOperator を使用します。

オペレータパネルファイル群として表 2 の tars ディレクトリから www.tar.gz を取得して `/home/daq` ディレクトリで展開します。なお、今後 httpd ユーザー (デフォルトの設定では nobody) が `/home/daq` 以下にアクセスしますので `/home/daq` ディレクトリのパーミッションは 0755 である必要があります (2.1.6 節をごらんください)。

```
daq@cpuui% tar zxvf /path/to/www.tar.gz
```

これでできた www/operatorPanel/histogram シンボリックリンクファイルを一旦消去し、表 1 に示した「ヒストグラム png ファイル群」ディレクトリを差すようにシンボリックリンクファイルを作りなおします。

```
daq@cpuui% cd /home/daq/www/operatorPanel
daq@cpuui% rm histogram
daq@cpuui% mkdir /nfs/data/png
daq@cpuui% ln -s /nfs/data/png histogram
```

また www/operatorPanel/operatorPanel0.html にある

```
<frame name = "menu" src="http://onlxw1.kek.jp/daq/operatorPanel/operatormenu.html" />
```

のホスト名の部分 (onlxw1.kek.jp の部分) を使用する CPU UI のホスト名、あるいは IP アドレスに書き換えます。また client.conf の設定を確認します。このファイルには DaqOperator コンポーネントが稼働している計算機のアドレス (あるいはホスト名) とポート番号を書きます。通常は host として localhost を指定します。

```
host=localhost
port=30000
```

localhost の IP アドレス (127.0.0.1) を取得するのに DNS を使うのはトラブルの原因になりますので /etc/hosts で localhost の IP アドレスがひけるようになっているかどうか確認してください。次の行が /etc/hosts にあれば ok です。

```
127.0.0.1       localhost.localdomain   localhost
```

この client.conf の場所は daq.py 内ファイル先頭部分で指定されていますので、この指定が正しく client.conf を差しているかどうか確認してください。

続いて apache がこの www ディレクトリ以下にアクセスできるようにするため/var/www/html ディレクトリ内にシンボリックリンクを作ります。

```
root@cpuui# cd /var/www/html
root@cpuui# ln -s /home/daq/www daq
```

さらに mod_python 用設定ファイルを/etc/httpd/conf.d/以下に置き、httpd サーバーをスタートさせます。

```
root@cpuui# cp /home/daq/www/daq.conf /etc/httpd/conf.d/
root@cpuui# /sbin/chkconfig httpd on
root@cpuui# /etc/init.d/httpd start
```

run number が Working Desktop からこない場合の処置

現在の実装ではランナンバーは、OperatorPanel が稼働している計算機のウェブコンテンツ内の /home/daq/www/operatorPanel/runNumber.txt から取得し、またこのファイルに現在のランナンバーをセーブしています。ファイルへのセーブは httpd ユーザー権限で行われますのでこのファイルのパーミッションを誰でも書けるようにしておく必要があります。

```
daq% cd /home/daq/www/operatorPanel
daq% chmod 666 runNumber.txt
```

Working Desktop からランナンバーがやってくるようになれば runNumber.txt は必要なくなる予定です。

2.5 CPU UI での設定

図 1 のようなネットワーク構成の場合を例にとり設定方法を説明します。

2.5.1 run-comps.sh

run-comps.sh には CPU DAQ で稼働する各コンポーネントで必要になるシェアードライブラリ (libmanyo.so、libJsonSpirit.so、libSock.so、libpm_reg.so、libpsdmodule.so、libsitcpbcp.so、libsock.so) のあるディレクトリを指定する変数 MANYOLIB および SOCKLIB が定義されています。この変数で定義されたディレクトリが CPU DAQ の現状とあっているかどうか確認してください。違っている場合は run-comps.sh を編集する必要があります。また、各コンポーネントがあるディレクトリを DIR_COMPS 変数で指定しているので、これも CPU DAQ の現状と合致しているか確認してください。違っている場合は run-comps.sh を編集する必要があります。

3 設定ファイルについて

以上で DAQ Middleware ソフトウェアのインストールは完了です。DAQ Middleware を動作させるためには config.xml および conditon.xml のふたつの設定が必要です。これらの設定については次節以降で説明します。

4 設定ファイル概要 (config.xml および coniditon.xml)

DAQ Middlware には

- config.xml (DAQ 構成設定ファイル)
- condition.xml (DAQ 機器設定ファイル)

の 2 種類の設定ファイルが存在します。表 3 にそれぞれの役割を示します。config.xml には使用する DAQ コンポーネントとそれらの接続情報等が入っており、DAQ 担当者により準備、変更されるものです。DAQ コンポーネントの組合せ等は一旦 DAQ システムの構成が決まればあまり変更するものではありません。condition.xml は読みだし機器やオンライン解析用の設定を行うファイルで各 DAQ コンポーネント毎に用意します。このファイルは DAQ 担当者、一般ユーザーが準備、変更するもので、実験のラン毎に変更することが可能です。これらのファイルは XML で書きます。condition.xml については XML 構文解析をユーザーが個別に実装するのは手がかかるため一旦 JSON(JavaScript Object Notation) という簡潔なデータ形式に変換し、各 DAQ コンポーネントは変換された JSON 形式データを読んで構文解析します。

次節からこの二つの設定ファイルについて解説します。

ファイル名	機能	ファイルの作成と変更
config.xml	DAQ システムのコンフィギュレーション	DAQ 担当者
condition.xml	機器設定、オンライン解析用パラメータ設定	DAQ 担当者、一般ユーザー

表 3 config.xml と condition.json の機能

5 コンフィグレーションファイル (config.xml)

config.xml 中では以下の項目を設定します。

- 使用する DAQ コンポーネントの名前
- 使用する DAQ コンポーネントを起動するホストの IP アドレス
- 使用する DAQ コンポーネント間の接続情報
- 使用する DAQ コンポーネントの起動順番情報

DAQ オペレータコンポーネントが `config.xml` を読みシステムを把握します。

従来はエディタを使って人間が手書きしていましたが、編集ミス等があるので `config.xml` を生成するプログラム `mkconfig_gui.py` を準備しました。このプログラムについて次の節で解説します。

5.1 コンフィグレーションファイル生成スクリプト

これまでは、ユーザが手動で自身の config.xml を書いていました。この方法では、勘違いやミスにより DAQ コンポーネントが起動しなかったりエラーが起こったりすることがありました。そこで config.xml を生成するプログラムを作りました。python で実装しているので python を使える人は自分で改良することができます。このスクリプトを使用することで次の利点があります。

- このスクリプトにより生成したコンフィグレーション・ファイルの XML の構造、DAQ ミドルウェアに関する必要事項についてもれなく設定されることが保証されます。
- ユーザが入力すべき情報は最小限になり、入力ミスを最小限にできます。
- 12 月版と 4 月版以降では、config.xml のスキーマに変更がありました（詳細は 5.3 を参照)。このスクリプトを使うことで新しいスキーマにしたがった config.xml を作ることができます。

これらの利点がありますので 12 月版から 4 月版以降の版へ移行するときには、config.xml はこのスクリプトで新規に作成することを推奨します。

本節ではこのスクリプトについて解説します。

5.1.1 ファイル構成等

コンフィグレーションファイル生成スクリプトは、次の 3 つのファイルから構成されています。

- mkconfig_gui.py : Tkinter による GUI 処理
- genconf.py : XML 生成処理
- dom_tree.py : 生成したファイルをツリー表示

この生成スクリプトを使用する場合は、Tkinter が必要です。Tkinter は RHEL5、あるいはそのクローン OS(Scientific Linux 5、CentOS 5) で rpm ファイルが提供されていますのでまだインストールしていない場合は yum などを使ってインストールしておく必要があります。Tkinter のパッケージ名は tkinter(t が小文字) です。yum を使って RHEL、あるいはそのクローン OS のディストリビューションパッケージがインストールできる環境であれば

```
root# yum install tkinter
```

でインストールできます。

5.1.2 制限事項

現在のところ、この config.xml 生成 GUI には以下の制限があります。

- このスクリプトは、DAQ ミドルウェア 4 月版以降に対応した PSD 検出器系の config.xml を生成します。
- 既存の config.xml を読み込んで編集する機能はありません。
- GUI による入力項目を減らすため、次の項目は genconf.py 中で固定値が使われています。利用者の環境に合わせて適宜変更してください。

`MAX_FILE_BSIZE_IN_MB = '1024'`

イベント保存用ファイルはこのサイズ（単位メガバイト）以上になると分割され枝番がつく。

`MON_GNUPLOT_PATH = '/home/daq/gnuplot/bin/gnuplot'`

Monitor コンポーネントで称する gunplot の絶対パス。

`MON_PNG_OUT_DIR = '/home/daq/Data/png'`

Monitor コンポーネントが PNG 形式のヒストグラム・ファイルを書き出すディレクトリ。

`NUM_PSD_PER_MOD = '8'`

NEUNET の PSD のチャンネル数。

`GATENET_DEFAULT_ADDR = '192.168.0.15'`

GATENET の IP アドレス。

このスクリプトは自由に修正・変更してください。また修正・変更した場合は、お知らせください。

5.1.3 使用方法

5.1.4 スクリプトの起動

config.xml 作成スクリプトを起動する際には CPU DAQ の数を必ず `-n` オプションで指定しなければなりません。出力ファイル名のデフォルト値は `config.xml.generated` です。出力ファイル名を変更したいときには `-o` オプションを指定してください。

```
$ ./mkconfig_gui.py -n CPU DAQ の数 -o 出力ファイル名
```

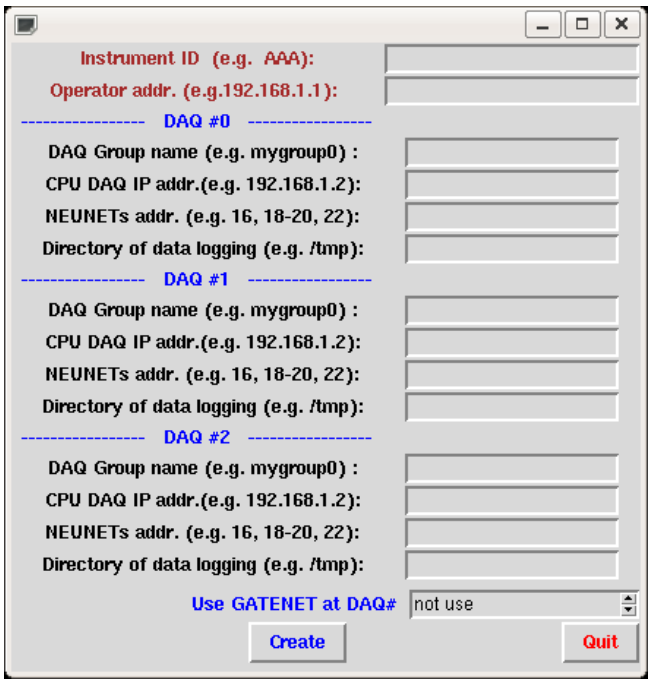

図 2 config.xml **作成用** GUI **のフレーム**

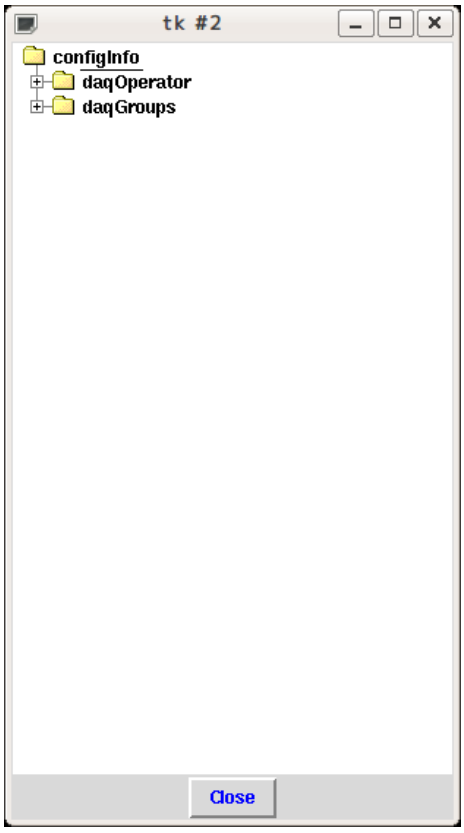

図 3 生成したファイル確認用フレーム (展開前)

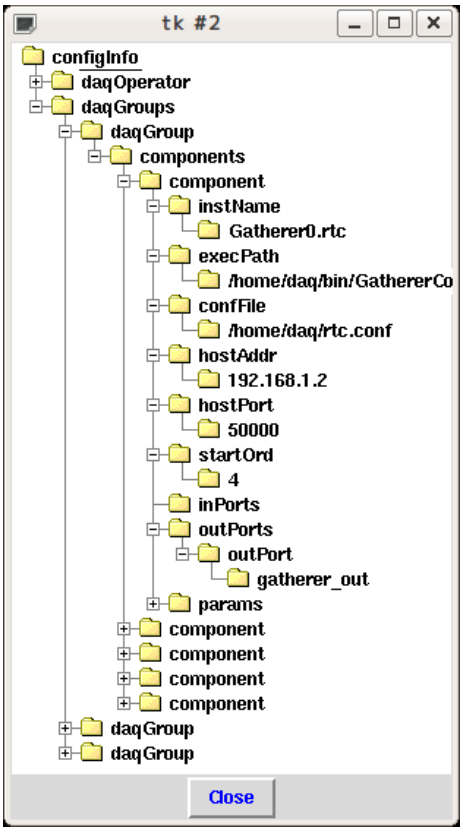

図 4 生成したファイル確認用フレーム (展開後)

図 2 に、./mkconfig_gui.py -n 3 -o config3.xml を入力した際に現れるフレームを示します。以下このコマンドを使用して起動した場合を例にとり使用方法を説明します。

5.1.5 テキストボックスへの入力

CPU DAQ の数を "3" と指定したので、画面には DAQ#0, DAQ#1, DAQ#2 と 3 つのグループに関する入力欄が現れます。

フレームの上から 1 番目と 2 番目は、グループ共通の情報です。1 番上は、装置 ID（各装置ですでに決まっている）を入力します。装置 ID はイベントデータを保存するディレクトリ名、ファイル名に使用されます。2 番目は、DAQ オペレータを起動する計算機の IP アドレスをドット区切りで（192.168.1.11 のように）指定します。

続いて各 CPU DAQ の情報を入力していきます。"DAQ Group name" は、スペースを含まない任意の文字列を入れてください。例えば、daqgroup0 とします。"CPU DAQ IPaddr." は、対応する IP アドレスをドット区切りで入力します。"NEUNETs addr." は、この CPU DAQ で読み出す NEUNET の IP アドレスの第 4 オクテットを入力します。NEUNET のフロントパネルのロータリースイッチで設定可能な値は第 4 オクテットのみだからです。残りのオクテットは "192.168.0" と固定です。指定の際、","(コンマ) を使って "16,18,20" というように入力すると飛び飛びの値の指定が可能です。また連続する値は "-"(ハイフン) を使って "16-19" と指定できます。"Directory of data logging" はイベントデータを保存するディレクトリを絶対パスで指定します。フレームの最下段にあるボタンの上に GATENET の情報を指定する欄があります。GATENET コンポーネントの制御をどの CPU DAQ で行うかを指定します。この例だと、"not use", "DAQ#0", "DAQ#1", "DAQ#2" のうちから一つを選択します。

5.1.6 コンフィグレーションファイルの生成

全ての欄を入力または選択した後、"Create" ボタンを押すとカレントディレクトリにコンフィグファイルが生成されます。今回は-o sample3.xml を指定して起動したので sample3.xml というファイルが生成されそのファイルに保存されます。-o オプションを指定しない場合は上に述べたように config.xml.generate というファイルに保存されます。

5.1.7 生成したコンフィグレーションの確認

"Create" ボタンを押した後、図 5.1.3 のような XML の構造を可視化した図が現れます。その構造で展開可能なノードをクリックすると図 5.1.3 のように展開することができます。ノードを展開して生成したコンフィグレーションに問題がないか確かめてください。

5.1.8 スクリプトの終了

コンフィグレーション確認用のフレームの "Close" ボタンを押してフレームを閉じます。テキストボックスのあるフレームの "Quit" ボタンを押してスクリプト終了します。

5.2 config.xml の詳細

上述の config.xml **作成** GUI **を使えばほとんどの場合エディタ等で手で編集することはないと思いますがより詳細な情報が必要な場合は以下を参考にしてください。**

5.2.1 測定データ保存用ディレクトリおよびファイル名について

実験データ保存用ディレクトリおよびファイル名にはラン番号が付加されることが決まっています。ラン番号は実験制御システムにより管理される予定です。DAQ **ミドルウェアでは、**"Begin" **コマンドとともに実験制御システムから与えられるラン番号を使用する予定です。現時点では実験制御システム・フレームワークが使用できるか自明ではないので、ウェブサーバでラン番号を管理しています。**Begin **コマンド発行時に、ラン番号も** DaqOperator **へ送信します。測定データ保存用ディレクトリおよびファイル名は、その命名則が表** 4 **のように決まっています**[*2]**。**

- **ディレクトリ名**
  XXXnnnnnn_YYYYMMDD
- **ファイル名**
  XXXnnnnnn_NN_MMM_mmm.edb

記 号	**説 明**
XXX	**装置コード**
nnnnnn	**ラン番号**
YYYYMMDD	**年月日**
NN	CPU DAQ **番号（バンク番号）**
MMM	**モジュール番号**
mmm	**枝番（ファイルを分割した際に付加）**

表 4 **イベントデータファイルのファイル名に使用する記号について**

5.2.2 データ保存に関する設定

データ保存に関する設定は、config.xml **の** Logger **コンポーネントに関する記述で行います。**⟨parampid="isLogging"⟩yes⟨/param⟩ **の値が** "yes" **ならデータを保存し、**"no" **なら保存しません。**

⟨param pid="dirName"⟩/tmp⟨/param⟩ **はデータ保存用のディレクトリとなります。このディレクトリの下にラン毎にディレクトリが作られその下にモジュール毎にファイルが保存されます。**

[*2] **中谷 健、測定データ保存ディレクトリとファイル名（**MS Word file**）、**2008 **年** 4 **月** 21 **日**

タグ名	属 性	説 明
configInfo	なし	ルート・エレメント
daqGroups	なし	子エレメントとして 1 個以上の daqGroup をもつ
daqGroup	gid	グループ ID を文字列で指定する
components	なし	子エレメントとして 1 個以上の daqComponent をもつ
component	cid	コンポーネント ID を文字列で指定する
hostAddr	なし	コンポーネントを起動させるホストの IP Address
hostPort	なし	コンポーネントを起動させるホストの xinetd のポート番号
instName	なし	コンポーネントのインスタンス名
execPath	なし	コンポーネントの実行形式ファイルの絶対パス
confFile	なし	コンポーネントの使用する rtc.conf ファイルの絶対パス
startOrd	なし	コンポーネントのスタートコマンド投入の際の順番
inPorts	なし	子エレメントとして 0 個以上の inPort をもつ
outPorts	なし	子エレメントとして 0 個以上の outPort をもつ
inPort	なし	inPort の名前
outPort	なし	outPort の名前
params	なし	子エレメントとして 0 個以上の param をもつ
param	pid	pid 属性に対応した値をもつ

表 5 コンフィグレーションファイルに使用するタグ

複数の Gatherer から NFS でマウントしたディスクにデータを保存する際には注意が必要です。ラン毎に作成するディレクトリの直上のディレクトリ名がユニークになるようにします。例えば

```
<param pid="dirName">/nfs/cpudaq0</param>
:
:
<param pid="dirName">/nfs/cpudaq1</param>
```

⟨param pid="maxFileSizeInMegaByte"⟩1024⟨/param⟩ は、ファイルがこの大きさ以上になった場合、ファイルを閉じて枝番をインクリメントして新たにファイルを開きます。これは、1 つの巨大なファイルの場合、ファイルに問題が発生すると全てのデータが読めなくなる可能性がありますが、特定の大きさでファイルを分割することでその危険性を低減させるためです。現在の設定では 1Gbyte となっています。

5.2.3 Gatherer コンポーネントで指定できる param

`daqId`

daqId を指定する。

`srcAddr`

データを取得する機器の IP アドレスを `param` で指定します。IP アドレスの後ろに “/” を付け、その後ろにモジュール番号を書く必要があります。モジュール番号は 0 から始まり、連番である必要があります。IP アドレスとモジュール番号の組はユニークである必要があります。

5.2.4 Dispather コンポーネントで指定できる param

`eventByteSize`

イベントバイトサイズを指定します。

`samplingRate`

1 以上の整数で Dispatcher から Monitor へ転送するデータの割合を指定します。データを全て Monitor へ転送する場合は “1” を、10 分の 1 のデータを転送する場合は “10” と指定します。この `samplingRate` は Dispatcher および Monitor とで同一の値を指定しなければなりません。

5.2.5 Logger コンポーネントで指定できる param

`daqId`

daqId を指定する。

`srcAddr`

データを取得する機器の IP アドレスを `param` で指定します。IP アドレスの後ろに “/” を付け、その後ろにモジュール番号を書く必要があります。モジュール番号は 0 から始まり、連番である必要があります。IP アドレスとモジュール番号の組はユニークである必要があります。

`eventByteSize`

イベントバイトサイズを指定します。

`isLogging`

文字列 (“yes” あるいは “no”) で取得したデータを Logger で保存するかどうかを指定します。

`maxFileSizeInMegaByte`

0 以上の整数でデータ保存ファイルの大きさがここで指定されたサイズ (MByte) 以上になる

とファイルを閉じ、新たにファイル開きます。その際、ファイル名の枝番が 1 つ増えます。

`instId`

英大文字 3 文字で装置 ID を指定します。

`dirName`

取得データ保存用のディレクトリを指定します。ディレクトリが存在していなければ Logger コンポーネントがエラーとなります。

5.2.6 Monitor コンポーネントで設定できる param

`daqId`

daqId を指定する。

`srcAddr`

データを取得する機器の IP アドレスを `param` で指定します。IP アドレスの後ろに “/” を付け、その後ろにモジュール番号を書く必要があります。モジュール番号は 0 から始まり、連番である必要があります。IP アドレスとモジュール番号の組はユニークである必要があります。

`samplingRate`

1 以上の整数で Dispatcher から Monitor へ転送するデータの割合を指定します。データを全て Monitor へ転送する場合は “1” を、10 分の 1 のデータを転送する場合は “10” と指定します。この `samplingRate` は Dispatcher および Monitor とで同一の値を指定しなければなりません。

`gnuplot_path`

gnuplot executable の絶対パスを指定します。

`png_output_dir`

各種ヒストグラム png ファイルを出力するディレクトリ名を指定します。

`num_of_psd_per_module`

1 以上の整数で NEUNET モジュール 1 台あたりの PSD 数を指定します。

5.3 コンフィグレーションファイルのスキーマの変更について

2009 年 4 月版以降では config.xml のスキーマに次の変更がありました。

1. DAQ オペレータを起動する計算機の IP アドレスの追加
2. PSD 系 DAQ システムの NEUNET の IP アドレスを指定する際に付加していたモジュール番号の廃止
3. GATENET の IP アドレスを指定可能にした

上記の 1 **の変更は、**run.sh **を廃止して** run.py **を新たに導入したことによります。この** run.py **は、**config.xml **をパーズして** CPU DAQ **の** IP **アドレスと** DAQ **オペレータの** IP **アドレスを取得します。**12 **月版以前のように起動スクリプト** run.sh **で** CPU DAQ **の** IP **アドレスと** DAQ **オペレータの** IP **アドレスをユーザが指定する必要はありません。**

上記 2 **は、**NEUNET **の** IP **アドレスとそのモジュール番号の指定を同時に行わなければならず繁雑で面倒でした。このモジュール番号の指定を廃止し、モジュール番号は** 0 **から自動で** config.xml **に記述順に付加するようにしました。詳細は** 5.3.2 **節で後述します。**

上記 3 **は、これまで** GATENET **の** IP **アドレスは** 192.168.0.15 **とし、変更しないことを前提にしていました。指定がない場合は、**192.168.0.15 **と設定されます。**

5.3.1 DAQ オペレータを起動する計算機の IP アドレス追加

4 **月版以前は下記のように指定方法していました。**

```
<?xml version="1.0"?>
   <configInfo>
      <daqGroups>
         <daqGroup gid="group0">
            <components>
               <component cid="Gatherer0">
                  <instName>Gatherer0.rtc</instName>
                  <hostAddr>192.168.1.206</hostAddr>
                  <startOrd>4</startOrd>
          ...
```

4 **月版では、下記のように**<daqGrooups>**の前に**<daqOperator>**という要素を追加して**<hostAddr>**に** DAQ **オペレータを起動する計算機の** IP **アドレスを記述します。**

```
<?xml version="1.0"?>
   <configInfo>
      <daqOperator>
         <hostAddr>192.168.1.206</hostAddr>
      </daqOperator>
      <daqGroups>
         <daqGroup gid="group0">
            <components>
               <component cid="Gatherer0">
                  <instName>Gatherer0.rtc</instName>
                  <hostAddr>192.168.1.206</hostAddr>
                  <startOrd>4</startOrd>
          ...
```

5.3.2 NEUNET の IP アドレス指定方法

4 **月版以前は下記のように指定方法していました。**IP **アドレスの後に** “/” **をデリミタとしてモジュール番号を** 0 **から順に** Gatherer **コンポーネントの**<params>**の中に書く必要がありました。**

```
  <params>
     <param pid="srcAddr">192.168.0.16/0</param>
```

```
    <param pid="srcAddr">192.168.0.17/1</param>
    <param pid="srcAddr">192.168.0.18/2</param>
    <param pid="srcAddr">192.168.0.19/3</param>
</params>
```

4 月版では、下記のようにモジュール番号は省略して IP アドレスのみを指定します。モジュール番号は記述順に自動的に割り当てられます。最初のモジュールのモジュール番号は 0 が割り当てられます。次の例では 192.168.0.16 がモジュール番号 0、192.168.0.17 がモジュール番号 1 が割り当てられ、最後の 192.168.0.19 のモジュール番号は 3 が割り当てられます。

```
<params>
   <param pid="srcAddr">192.168.0.16</param>
   <param pid="srcAddr">192.168.0.17</param>
   <param pid="srcAddr">192.168.0.18</param>
   <param pid="srcAddr">192.168.0.19</param>
</params>
```

5.3.3 GATENET の IP アドレス指定方法

4 月版以前のコンポーネントでは GATENET の IP アドレスは 192.168.0.15 に固定されており、config.xml 中に GATENET の IP アドレスを記述することはできませんでした。

4 月版以降では GATENET の IP アドレスが config.xml で指定できるようになりました。GATENET の IP アドレスは<param pid="gatenetAddr">で指定します。次の例は GATENET に 192.168.0.15 を割り当てる場合の例です。

```
<component cid="Gatenet0">
   <instName>Gatenet0.rtc</instName>
   <execPath>/home/daq/bin/GatenetComp</execPath>
   <confFile>/home/daq/rtc.conf</confFile>
   <hostAddr>192.1.1.1</hostAddr>
   <hostPort>50000</hostPort>
   <startOrd>5</startOrd>
   <params>
      <param pid="gatenetAddr">192.168.0.15</param>
   </params>
</component>
```

なお指定がない場合は 192.168.0.15 が固定的に設定されます。

6 コンディションファイル (condition.xml, condition.json)

コンディションファイルは、これから肥大化するであろうコンフィグレーション・ファイルからユーザ固有の情報を分離するために導入されました。iBIX（シンチ系）でその実装が行われ、実験に使用されています。PSD 系でも、それを参考に実装を行いました。

リードアウトモジュールの設定、オンラインモニタ用パラメータ設定やラン停止条件の設定のためにコンディションファイルが導入されました。これは今後増加すると思われる装置のパラメータおよびオンラインモニターのパラメータをユーザーが自由に作成・変更できることを目的としています。一方、コンフィグレーションファイルは、使用する計算機の情報や使用する DAQ コンポーネントの種類や数、その構成等のデータ収集システムを記述するもので DAQ エキスパートが作成・変更することを前提としています。

6.1 ファイル名

ユーザーが編集するコンディションファイルのファイル名は condition.xml です。これをスクリプトにより condition.json へ変換します。各 DAQ コンポーネントは（現在は、Gatherer および Monitor のみ）は、condition.json をパーズして自分に必要なパラメータを抽出します。condition.xml から condition.json への変換は、現在はユーザが手動で変換する必要がありますが、将来的にはソフトウェア・フレームワークにより自動的に変換が行われることになる予定です。

6.2 XML から JSON への変換方法

condition.xml から condition.json への変換は DAQ ミドルウェア 4 月版以降のパッケージに入っている `condition_xml2json` を使用します。

```
./condtion_xml2json condition.xml [condition.json]
```

“[” および “]” で囲った引数は省略可能であることを示します。第 2 引数が省略された場合は、出力ファイル名は XML 文章のファイル名中の `xml` が `json` に置き換わったものになります。たとえば `mycondition.xml` からは `mycondition.json` というファイル名のファイルが生成されます。

6.3 コンディション情報が読み込まれるタイミング

コンディション情報は、ラン開始時に読み込まれます。これはたとえばターゲットの温度等を変化させながら、ランの開始、停止を繰り返すような実験に対応するためです。

6.4 condition.xml の構造

現在の実装では、

- T0 **カウントによるラン停止**
- Gatherer **コンポーネントによる** NEUNET **モジュールの** LLD, TOF **最小値、**TOF **最大値**
- **モニターコンポーネントにおける**
  - TOF **ヒストグラムの最小値、最大値**
  - TOF **ヒストグラムのビンの数**
  - 1 **次元、**2 **次元ヒストグラムのビンの数** (1 **次元、**2 **次元で共通の値となる**)
  - **ヒストグラムをアップデートするレートの指定**

が設定可能です。下記にその例を示します。この例は、CPU DAQ ID#0 **と** CPU DAQ ID#1 **という** 2 **台の計算機でそれぞれ** 2 **台の** NEUNET **モジュールを読み出す場合の例です。モジュール** ID#0 **として** GATENET **がアサインされます。これは、**config.xml **の**<param pid="srcAddr"> **にこの順番で指定されているからです。**

```
<?xml version="1.0" encoding="utf-8" ?>
<condition>
  <common>
    <Gatherer>
      <runStopCondition>
        <type>T0</type>
        <value>1000</value>
      </runStopCondition>
    </Gatherer>
    <Gatenet>
      <initialPulseId>10000</initialPulseId>
      <initialExTrgId>20000</initialExTrgId>
    </Gatenet>
    <Monitor>
      <tof_min>1</tof_min>
      <tof_max>60000</tof_max>
      <num_of_tof_bin>500</num_of_tof_bin>
      <num_of_pixel_per_psd>500</num_of_pixel_per_psd>
      <monitor_update_rate>1000</monitor_update_rate>
    </Monitor>
  </common>
  <daq id="0">
    <module id="0">       <!-- GATENET MODULE -->
      <lld>0</lld>
      <minTof>0</minTof>
      <maxTof>0</maxTof>
    </module>
    <module id="1">       <!-- NEUNET MODULE # 0 on CPUDAQ 0 -->
      <lld>0</lld>
      <minTof>0</minTof>
      <maxTof>0</maxTof>
    </module>
    <module id="2">       <!-- NEUNET MODULE # 1 on CPUDAQ 0 -->
      <lld>0</lld>
```

```
      <minTof>0</minTof>
      <maxTof>0</maxTof>
    </module>
  </daq>
  <daq id="1">
    <module id="0">       <!-- NEUNET MODULE # 0 on CPUDAQ 1 -->
      <lld>0</lld>
      <minTof>0</minTof>
      <maxTof>0</maxTof>
    </module>
    <module id="1">       <!-- NEUNET MODULE # 1 on CPUDAQ 1 -->
      <lld>0</lld>
      <minTof>0</minTof>
      <maxTof>0</maxTof>
    </module>
  </daq>
</condition>
```

6.5 Gatherer データ読みだし停止条件の指定方法

Gatherer のデータ読みだし停止条件を設定する場合は下記のように、<common><Gatherer>タグの後に<runStopCondition>タグで指定します。指定する内容は<type>および<value>です。<type>が T0 の場合は、<value>にカウント数を書きます。

```
<common>
   <Gatherer>
      <runStopCondition>
         <type>T0</type>
         <value>100</value>
      </runStopCondition>
   </Gatherer>
</common>
```

<type>が COMMAND の場合は、従来のコマンドによる停止です。その際は<value>に STOP と記述します。

```
<common>
   <Gatherer>
      <runStopCondition>
         <type>COMMAND</type>
         <value>STOP</value>
      </runStopCondition>
   </Gatherer>
</common>
```

6.6 LLD, TOF の指定方法

各 NEUNET モジュールの LLD, TOF は下記のように指定します。ここで指定する module ID は config.xml で自動的に付加されるモジュール番号と同じであることに注意してください。

```
<module id="0">
   <lld>516</lld>
   <minTof>11016</minTof>
   <maxTof>41016</maxTof>
</module>
```

7 GATENET モジュール

DAQ ミドルウェアで GATENET モジュール [1] を制御する GATENET コンポーネントの使用法について説明します。

7.1 ゲートの制御

ゲートの開閉の制御は GATENET コンポーネントを使用して行います。GATENET コンポーネントを使用する場合は下記のように config.xml に GATENET コンポーネントを追加します。

GATENET モジュールの IP アドレスは<param>で指定できます。GATENET モジュールを使用するようになっているが IP アドレスが指定されていない場合は GATENET モジュールの IP アドレスは 192.168.0.15 であると設定されます。

各コンポーネントの起動の順番は<startOrd>タグで指定しますが、GATENET コンポーネントについては<startOrd> は最後のコンポーネントであると記述する必要があります。

なお、config.xml の生成スクリプト mkconfig-gui.py を使用した場合は<startOrd>については自動的に正しい値になり、IP アドレスは 192.168.0.15 と設定するような config.xml が生成されます。

```
<component cid="Gatenet0">
   <instName>Gatenet0.rtc</instName>
   <hostAddr>192.168.1.207</hostAddr>
   <!-- startOrd -->
   <startOrd>9</startOrd>
   <!-- 起動の順番は最後にする。他のコンポーネントが 8 つあった場合は 9 番目 -->
   <params>
      <param>192.168.0.15</param>
   </params>
</component>
```

7.2 GATENET の時刻調整

毎パラメータ設定時に、GATENET コンポーネントが GATENET の時刻合わせを行います。GATENET コンポーネントを起動する計算機（上記例では、IP アドレスが 192.168.1.207 の計算機）の時刻が GATENET のレジスタへ書き込まれます。装置時刻データ中の時刻情報は GATENET モジュールが保持している時刻情報を元にしているため、GATENET モジュールの時刻があっていないと装置時刻データ中の時刻情報も正しくない値となります。したがって装置時刻情報に正しい時刻情報を入れるためには GATENET コンポーネントを動かす計算機の時刻が正しくセットされていることが必要です。計算機の時刻を常に正しい値にしておくためには ntp を使用します。ntp の設定方法については 2.1.9 節を参照してください。

7.3 ゲートの制御のタイミング

ラン開始時は、全てのコンポーネントで開始の状態になった後で、一番最後に GATENET コンポーネントがゲートをオープンします。ラン停止時は、最初に GATENET コンポーネントがゲートをクローズした後、各コンポーネントがそれぞれ停止します。

7.4 パルス ID カウント、外部トリガカウント

T0 データ中のパルスカウントの初期値はコンディションファイルを使ってランスタート時に設定することができます。

の設定はランスタート時にコンディションファイルから書き込むことができます。コンディションファイルにパスルカウント初期値の指定がない場合はそれぞれのカウントは初期化されず、前のランからの継続した値となります。パルス ID カウントを 100、外部トリガカウントを 200 で初期化する場合は次のように condition.xml に書きます。

```
<condition>
   <common>
      <Gatherer>
         <runStopCondition>
            <type>T0</type>
            <value>1000</value>
         </runStopCondition>
      </Gatherer>
      <Gatenet>
         <initialPulseId>100</initialPulseId>
         <initialExTrgId>200</initialExTrgId>
      </Gatenet>
   </common>
...
```

パルス ID カウント、外部トリガカウントを初期化せず前の値からの続きで使用する場合はコンディションファイル中に initialPulseId および initialExTrgId を書かないようにしてください。

```
<condition>
   <common>
      <Gatherer>
         <runStopCondition>
            <type>T0</type>
            <value>1000</value>
         </runStopCondition>
      </Gatherer>
      <Gatenet>
         <!-- 書かないようにする
         <initialPulseId>100</initialPulseId>
         <initialExTrgId>200</initialExTrgId>
         -->
      </Gatenet>
   </common>
```

```
...
```

7.5 GATENET モニターデータ

GATENET のモニターデータは、Gatherer コンポーネントで取得します。その例の config.xml の例を下記に示します。このように config.xml に書くと GATENET モジュールからモジュール番号 0 として NEUNET モジュールと同様にデータ取得を行います。モニターデータのデータフォーマットは、NEUNET のイベントデータと同じですが、波高値データは左右ともに 1024 に固定です。オンラインモニタの位置のヒストグラムでは常に中心にピークが立っているように見えます。

```
  <params>
    <param pid="srcAddr">192.168.0.15</param>  <!-- GATENET Module -->
    <param pid="srcAddr">192.168.0.16</param>  <!-- NEUNET  Module -->
    <param pid="srcAddr">192.168.0.17</param>  <!-- NEUNET  Module -->
  </params>
```

GATENET でモニターデータを取得する際には、コンディションファイルにも GATENET の情報を記述する必要があります。書ける項目は NEUNET モジュールの場合と同じです。下記にその例を示します。

```
<?xml version="1.0" encoding="utf-8" ?>
<condition>
   <daq id="0">
      <runStopCondition>
         <type>COMMAND</type>
         <value>100</value>
      </runStopCondition>
      <module id="0">
         <!-- NEUNET モジュールと同じように設定するパラメータを書く -->
         <lld>516</lld>
         <minTof>11016</minTof>
         <maxTof>41016</maxTof>
      </module>
      <module id="1">
         <lld>516</lld>
         <minTof>11016</minTof>
         <maxTof>41016</maxTof>
      </module>
      <module id="2">
         <lld>580</lld>
         <minTof>11080</minTof>
         <maxTof>41080</maxTof>
      </module>
   </daq>
</condition>
```

上の例ではモジュール番号 0 が GATENET モジュールになるように config.xml が記述されていると仮定しています。

8 DAQ コンポーネント操作方法

CPU UI に daq ユーザーでログインして、run.py を実行します。
run.py は

- ウェブを使って DAQ コンポーネントに指示を出すウェブモード
- 端末エミュレータから DAQ コンポーネントに指示を出すコンソールモード

の二つのモードがあります。ウェブモードで起動するには

```
daq@cpuui% ./run.py
```

とします。コンソールモードで起動するには

```
daq@cpuui% ./run.py -c
```

と -c オプションを付けて実行します。どちらかを実行すると CPU DAQ 上で各 DAQ コンポーネントが起動し、待機状態になります。

8.1 ウェブモードでの DAQ コンポーネントの操作

ウェブモードでは DAQ コンポーネントの操作はウェブブラウザを通じて行います。

現在、動作が確認されているウェブブラウザは Firefox のみです。Internet Explorer、Safari、Opera、Chrome では動作しないことが確認されています。これらのウェブブラウザでの動作が確認されるまではウェブブラウザとして Firefox を使用してください。

適当な計算機 (CPU UI でよい) からウェブブラウザを使って CPU UI の /daq/operatorPanel/operatorPanel0.html にアクセスすると図 5 の画面が出ます。この画面のボタンを押して指示を出します。

configure の実行
: 各 CPU DAQ に対応するオペレータパネルの “Configure” ボタンを押す。DaqOperator が設定ファイル (config.xml) を読み、各コンポーネントの設定を行う。その後、各コンポーネントの状態を表す “Current State” が “Loaded” から “Configured” に変化すれば、コンフィグレーションは終了したことを示す。

ラン開始
: 各 CPU DAQ に対応するオペレータパネルの “Begin” ボタンを押す。その後、各コンポーネントの状態を表す “Current State” が “Configured” から “Running” に変化すれば、ランが開始したことを示す。また、各コンポーネントのイベント数は 0 に初期化される。

ラン停止

各 CPU DAQ **に対応するオペレータパネルの** “End” **ボタンを押す。その後、各コンポーネントの状態を表す** “Current State” **が** “Running” **から** “Configured” **に変化すれば、ランが終了したことを示す。**

Unconfigure

設定ファイルの変更を行う場合は、設定ファイルを変更後に “Unconfigure” **ボタンを押す。ステートが**”Configured” **から**” Loaded” **へ変化すれば、**“Configure” **ボタンを押して再度、設定ファイルが読み込まれる。**

次のランの開始

前回と同じ条件でランを開始させる場合は、“Begin” **ボタンと** “End” **ボタンにより操作を行う。**

`/daq/operatorPanel/operatorPanel1.html` **にアクセスするとヒストグラムファイルが表示されます。**

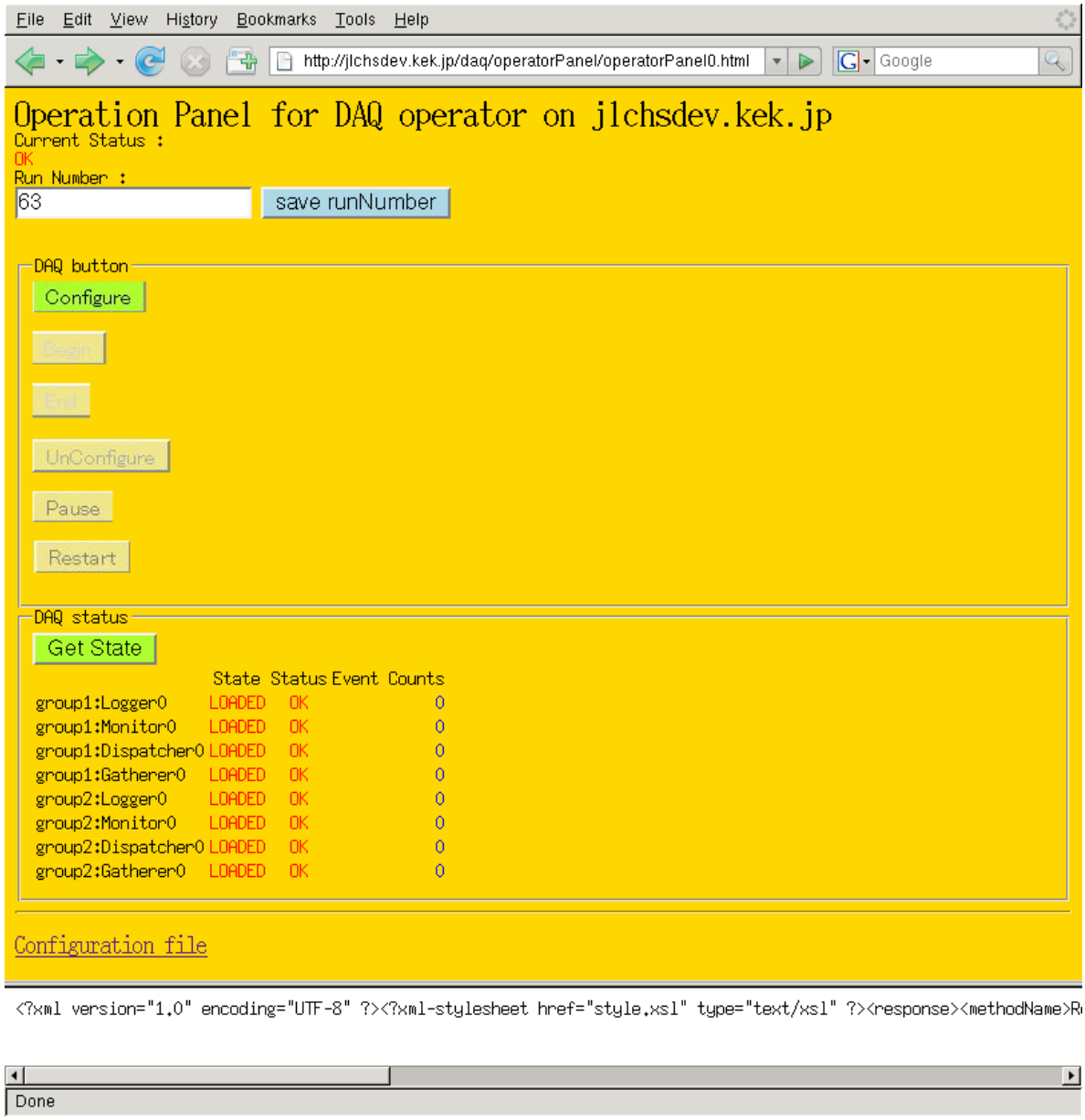
File Edit View History Bookmarks Tools Help
http://jlchsdev.kek.jp/daq/operatorPanel/operatorPanel0.html
Google
Operation Panel for DAQ operator on jlchsdev.kek.jp
Current Status :
OK
Run Number :
63
save runNumber
DAQ button
Configure
Begin
End
UnConfigure
Pause
Restart
DAQ status
Get State
State Status Event Counts
group1:Logger0 LOADED OK 0
group1:Monitor0 LOADED OK 0
group1:Dispatcher0 LOADED OK 0
group1:Gatherer0 LOADED OK 0
group2:Logger0 LOADED OK 0
group2:Monitor0 LOADED OK 0
group2:Dispatcher0 LOADED OK 0
group2:Gatherer0 LOADED OK 0
Configuration file
<?xml version="1.0" encoding="UTF-8" ?><?xml-stylesheet href="style.xsl" type="text/xsl" ?><response><methodName>R
Done

図 5 オペレータパネル。

```
$ ./run.py -c
Console mode
cons_mode: -c
130.87.234.118
CPU DAQ IP address:  192.168.1.2
confFile: ./config.xml
xmlwf -s ./config.xsd ./config.xml
omniNames: no process killed
start new naming service and wait for booting
Starting omniORB omniNames: no-such-host.kek.jp:9876

（中略。準備ができると端末がクリアーされ次の行が現れ入力待ちになる）

 Current State: LOADED
 Command:  0:configure  1:start  2:stop  3:unconfigure  4:pause  5:resume
0 0 0 0
```

図 6 コンソールモードでの起動例

8.2 コンソールモードでの DAQ コンポーネントの操作

コマンドプロンプトで run.py -c としてコンソールモードで起動するとコマンドに対応した番号（0-5）をキーボードから入力することで、ウェブサーバを介さず直接 DAQ コンポーネントへコマンドを発行することができます。run.py -c 実行後、準備ができると端末がクリアーされコマンド入力待ちになります。0 を押すと Configure が実行されます。このあと 1 を押すとランナンバーを聞いてきますので適当なランナンバーを入力します。ランナンバーを入力するとデータ読みだしが始まります。2 を押すとデータ読みだしを停止します。

コンソールモードでの起動例を図 6 に示します。

付録 A　config.xml の例

mkconfig-gui.py で作成した config.xml の例を以下に載せます。

`./mkconfig_gui.py -n 3` と CPU DAQ 数を 3 として起動して入力欄は

```
Instrument ID: ABC
Operator addr.: 192.168.1.1
DAQ #0
DAQ Group name: abcgroup0
CPU DAQ IP addr.: 192.168.1.2
NEUNETs addr.: 16-20
Directory of data logging: /kensdaq/edata
DAQ #1
DAQ Group name: abcgroup1
CPU DAQ IP addr.: 192.168.1.3
NEUNETs addr.: 21-25
Directory of data logging: /kensdaq/edata
DAQ #2
DAQ Group name: abcgroup2
CPU DAQ IP addr.: 192.168.1.4
NEUNETs addr.: 26-30
Directory of data logging: /kensdaq/edata
Use GATENET at DAQ#: 0
```

と入力しました。

```
<?xml version="1.0"?>
<configInfo>
   <daqOperator>
      <hostAddr>192.168.1.1</hostAddr>
   </daqOperator>
   <daqGroups>
      <daqGroup gid="group0">
         <components>
            <component cid="Gatherer0">
               <instName>Gatherer0.rtc</instName>
               <execPath>/home/daq/bin/GathererComp</execPath>
               <confFile>/home/daq/rtc.conf</confFile>
               <hostAddr>192.168.1.2</hostAddr>
               <hostPort>50000</hostPort>
               <startOrd>4</startOrd>
               <inPorts>
               </inPorts>
               <outPorts>
                  <outPort>gatherer_out</outPort>
               </outPorts>
               <params>
                  <param pid="daqId">0</param>
                  <param pid="srcAddr">192.168.0.16</param>
                  <param pid="srcAddr">192.168.0.17</param>
                  <param pid="srcAddr">192.168.0.18</param>
                  <param pid="srcAddr">192.168.0.19</param>
                  <param pid="srcAddr">192.168.0.20</param>
               </params>
            </component>
            <component cid="Dispatcher0">
               <instName>Dispatcher0.rtc</instName>
               <execPath>/home/daq/bin/DispatcherComp</execPath>
               <confFile>/home/daq/rtc.conf</confFile>
               <hostAddr>192.168.1.2</hostAddr>
               <hostPort>50000</hostPort>
               <startOrd>3</startOrd>
               <inPorts>
                  <inPort from="Gatherer0:gatherer_out">dispatcher_in</inPort>
               </inPorts>
               <outPorts>
                  <outPort>dispatcher_out1</outPort>
                  <outPort>dispatcher_out2</outPort>
               </outPorts>
               <params>
                  <param pid="eventByteSize">8</param>
                  <param pid="samplingRate">dispatcher_out2/10</param>
               </params>
            </component>
            <component cid="Logger0">
               <instName>Logger0.rtc</instName>
               <execPath>/home/daq/bin/LoggerComp</execPath>
               <confFile>/home/daq/rtc.conf</confFile>
               <hostAddr>192.168.1.2</hostAddr>
               <hostPort>50000</hostPort>
               <startOrd>1</startOrd>
               <inPorts>
                  <inPort from="Dispatcher0:dispatcher_out1">logger_in</inPort>
               </inPorts>
               <outPorts>
               </outPorts>
               <params>
                  <param pid="daqId">0</param>
                  <param pid="srcAddr">192.168.0.16</param>
                  <param pid="srcAddr">192.168.0.17</param>
                  <param pid="srcAddr">192.168.0.18</param>
                  <param pid="srcAddr">192.168.0.19</param>
                  <param pid="srcAddr">192.168.0.20</param>
                  <param pid="instId">ABC</param>
                  <param pid="dirName">/kensdaq/edata</param>
                  <param pid="eventByteSize">8</param>
                  <param pid="isLogging">yes</param>
                  <param pid="maxFileSizeInMegaByte">1024</param>
               </params>
            </component>
            <component cid="Monitor0">
               <instName>Monitor0.rtc</instName>
               <execPath>/home/daq/bin/MonitorComp</execPath>
               <confFile>/home/daq/rtc.conf</confFile>
               <hostAddr>192.168.1.2</hostAddr>
               <hostPort>50000</hostPort>
               <startOrd>2</startOrd>
               <inPorts>
                  <inPort from="Dispatcher0:dispatcher_out2">monitor_in</inPort>
               </inPorts>
               <outPorts>
               </outPorts>
               <params>
                  <param pid="daqId">0</param>
                  <param pid="srcAddr">192.168.0.16</param>
                  <param pid="srcAddr">192.168.0.17</param>
                  <param pid="srcAddr">192.168.0.18</param>
                  <param pid="srcAddr">192.168.0.19</param>
                  <param pid="srcAddr">192.168.0.20</param>
                  <param pid="samplingRate">dispatcher_out2/10</param>
                  <param pid="gnuplot_path">/home/daq/gnuplot/bin/gnuplot</param>
                  <param pid="png_output_dir">/home/daq/Data/png</param>
                  <param pid="num_of_psd_per_module">8</param>
               </params>
            </component>
            <component cid="Gatenet0">
               <instName>Gatenet0.rtc</instName>
               <execPath>/home/daq/bin/GatenetComp</execPath>
               <confFile>/home/daq/rtc.conf</confFile>
               <hostAddr>192.168.1.2</hostAddr>
               <hostPort>50000</hostPort>
               <startOrd>13</startOrd>
               <params>
                  <param pid="gatenetAddr">192.168.0.15</param>
               </params>
            </component>
         </components>
      </daqGroup>
      <daqGroup gid="group1">
         <components>
            <component cid="Gatherer0">
```

```
      <instName>Gatherer0.rtc</instName>
      <execPath>/home/daq/bin/GathererComp</execPath>
      <confFile>/home/daq/rtc.conf</confFile>
      <hostAddr>192.168.1.3</hostAddr>
      <hostPort>50000</hostPort>
      <startOrd>8</startOrd>
      <inPorts>
      </inPorts>
      <outPorts>
         <outPort>gatherer_out</outPort>
      </outPorts>
      <params>
         <param pid="daqId">1</param>
         <param pid="srcAddr">192.168.0.21</param>
         <param pid="srcAddr">192.168.0.22</param>
         <param pid="srcAddr">192.168.0.23</param>
         <param pid="srcAddr">192.168.0.24</param>
         <param pid="srcAddr">192.168.0.25</param>
      </params>
   </component>
   <component cid="Dispatcher0">
      <instName>Dispatcher0.rtc</instName>
      <execPath>/home/daq/bin/DispatcherComp</execPath>
      <confFile>/home/daq/rtc.conf</confFile>
      <hostAddr>192.168.1.3</hostAddr>
      <hostPort>50000</hostPort>
      <startOrd>7</startOrd>
      <inPorts>
         <inPort from="Gatherer0:gatherer_out">dispatcher_in</inPort>
      </inPorts>
      <outPorts>
         <outPort>dispatcher_out1</outPort>
         <outPort>dispatcher_out2</outPort>
      </outPorts>
      <params>
         <param pid="eventByteSize">8</param>
         <param pid="samplingRate">dispatcher_out2/10</param>
      </params>
   </component>
   <component cid="Logger0">
      <instName>Logger0.rtc</instName>
      <execPath>/home/daq/bin/LoggerComp</execPath>
      <confFile>/home/daq/rtc.conf</confFile>
      <hostAddr>192.168.1.3</hostAddr>
      <hostPort>50000</hostPort>
      <startOrd>5</startOrd>
      <inPorts>
         <inPort from="Dispatcher0:dispatcher_out1">logger_in</inPort>
      </inPorts>
      <outPorts>
      </outPorts>
      <params>
         <param pid="daqId">1</param>
         <param pid="srcAddr">192.168.0.21</param>
         <param pid="srcAddr">192.168.0.22</param>
         <param pid="srcAddr">192.168.0.23</param>
         <param pid="srcAddr">192.168.0.24</param>
         <param pid="srcAddr">192.168.0.25</param>
         <param pid="instId">ABC</param>
         <param pid="dirName">/kensdaq/edata</param>
         <param pid="eventByteSize">8</param>
         <param pid="isLogging">yes</param>
         <param pid="maxFileSizeInMegaByte">1024</param>
      </params>
   </component>
   <component cid="Monitor0">
      <instName>Monitor0.rtc</instName>
      <execPath>/home/daq/bin/MonitorComp</execPath>
      <confFile>/home/daq/rtc.conf</confFile>
      <hostAddr>192.168.1.3</hostAddr>
      <hostPort>50000</hostPort>
      <startOrd>6</startOrd>
      <inPorts>
         <inPort from="Dispatcher0:dispatcher_out2">monitor_in</inPort>
      </inPorts>
      <outPorts>
      </outPorts>
      <params>
         <param pid="daqId">1</param>
         <param pid="srcAddr">192.168.0.21</param>
         <param pid="srcAddr">192.168.0.22</param>
         <param pid="srcAddr">192.168.0.23</param>
         <param pid="srcAddr">192.168.0.24</param>
         <param pid="srcAddr">192.168.0.25</param>
         <param pid="samplingRate">dispatcher_out2/10</param>
         <param pid="gnuplot_path">/home/daq/gnuplot/bin/gnuplot</param>
         <param pid="png_output_dir">/home/daq/Data/png</param>
         <param pid="num_of_psd_per_module">8</param>
      </params>
   </component>
  </components>
</daqGroup>
<daqGroup gid="group2">
  <components>
    <component cid="Gatherer0">
      <instName>Gatherer0.rtc</instName>
      <execPath>/home/daq/bin/GathererComp</execPath>
      <confFile>/home/daq/rtc.conf</confFile>
      <hostAddr>192.168.1.3</hostAddr>
      <hostPort>50000</hostPort>
      <startOrd>12</startOrd>
      <inPorts>
      </inPorts>
      <outPorts>
         <outPort>gatherer_out</outPort>
      </outPorts>
      <params>
         <param pid="daqId">2</param>
         <param pid="srcAddr">192.168.0.26</param>
         <param pid="srcAddr">192.168.0.27</param>
         <param pid="srcAddr">192.168.0.28</param>
         <param pid="srcAddr">192.168.0.29</param>
         <param pid="srcAddr">192.168.0.30</param>
      </params>
    </component>
```

```
231         <component cid="Dispatcher0">
232            <instName>Dispatcher0.rtc</instName>
233            <execPath>/home/daq/bin/DispatcherComp</execPath>
234            <confFile>/home/daq/rtc.conf</confFile>
235            <hostAddr>192.168.1.3</hostAddr>
236            <hostPort>50000</hostPort>
237            <startOrd>11</startOrd>
238            <inPorts>
239               <inPort from="Gatherer0:gatherer_out">dispatcher_in</inPort>
240            </inPorts>
241            <outPorts>
242               <outPort>dispatcher_out1</outPort>
243               <outPort>dispatcher_out2</outPort>
244            </outPorts>
245            <params>
246               <param pid="eventByteSize">8</param>
247               <param pid="samplingRate">dispatcher_out2/10</param>
248            </params>
249         </component>
250         <component cid="Logger0">
251            <instName>Logger0.rtc</instName>
252            <execPath>/home/daq/bin/LoggerComp</execPath>
253            <confFile>/home/daq/rtc.conf</confFile>
254            <hostAddr>192.168.1.3</hostAddr>
255            <hostPort>50000</hostPort>
256            <startOrd>9</startOrd>
257            <inPorts>
258               <inPort from="Dispatcher0:dispatcher_out1">logger_in</inPort>
259            </inPorts>
260            <outPorts>
261            </outPorts>
262            <params>
263               <param pid="daqId">2</param>
264               <param pid="srcAddr">192.168.0.26</param>
265               <param pid="srcAddr">192.168.0.27</param>
266               <param pid="srcAddr">192.168.0.28</param>
267               <param pid="srcAddr">192.168.0.29</param>
268               <param pid="srcAddr">192.168.0.30</param>
269               <param pid="instId">ABC</param>
270               <param pid="dirName">/kensdaq/edata</param>
271               <param pid="eventByteSize">8</param>
272               <param pid="isLogging">yes</param>
273               <param pid="maxFileSizeInMegaByte">1024</param>
274            </params>
275         </component>
276         <component cid="Monitor0">
277            <instName>Monitor0.rtc</instName>
278            <execPath>/home/daq/bin/MonitorComp</execPath>
279            <confFile>/home/daq/rtc.conf</confFile>
280            <hostAddr>192.168.1.3</hostAddr>
281            <hostPort>50000</hostPort>
282            <startOrd>10</startOrd>
283            <inPorts>
284               <inPort from="Dispatcher0:dispatcher_out2">monitor_in</inPort>
285            </inPorts>
286            <outPorts>
287            </outPorts>
288            <params>
289               <param pid="daqId">2</param>
290               <param pid="srcAddr">192.168.0.26</param>
291               <param pid="srcAddr">192.168.0.27</param>
292               <param pid="srcAddr">192.168.0.28</param>
293               <param pid="srcAddr">192.168.0.29</param>
294               <param pid="srcAddr">192.168.0.30</param>
295               <param pid="samplingRate">dispatcher_out2/10</param>
296               <param pid="gnuplot_path">/home/daq/gnuplot/bin/gnuplot</param>
297               <param pid="png_output_dir">/home/daq/Data/png</param>
298               <param pid="num_of_psd_per_module">8</param>
299            </params>
300         </component>
301      </components>
302   </daqGroup>
303 </daqGroups>
304 </configInfo>
```

付録 B　CPU DAQ での各コンポーネントのブートメカニズム

各コンポーネントの起動メカニズムを図 7 をもとに解説します。

1. daq ユーザーが CPU UI のコマンドプロンプトから run.py を起動します。run.py はまず、各コンポーネントがネームサーバーと通信するのに必要となる情報が書かれたファイル rtc.conf を作成し、各 CPU DAQ 計算機にネットワークを通じて rtc.conf を送ります。CPU DAQ 側ではこのファイルを受信するために xinetd から起動される受信サーバー bootComps.py を使います。受信したファイルは /tmp/rtc.conf に保存されます。なお rtc.conf の内容は各 CPU DAQ 計算機の構成にマッチしている必要があります。
2. run.py は続いてコンポーネント起動用スクリプト run-comps.sh ファイルを各 CPU DAQ に送ります。CPU DAQ 側では rtc.conf ファイルと同様に xinetd から起動された受信サーバー bootComps.py を使って run-comps.sh を受信します。受信したファイルは /tmp/run-comps.sh として保存されます。
3. xinetd から起動された bootComps.py は、run-comps.sh を受信後 /tmp/run-comps.sh を system() 関数で実行します。これで各コンポーネントが起動します。
4. 起動したコンポーネントは /tmp/rtc.conf を参照し、そこに書かれた情報をもとに Naming service へ自身を登録します。
5. CPU UI の run.py は以上の動作が終了するまで適当な時間 sleep() して待っています。sleep() 終了後、各コンポーネントが起動したと仮定し、run.py から DaqOperator が起動されます。
6. 起動した DaqOperator はローカルにある config.xml をパーズし、必要なコンポーネントを Naming service へ問い合わせ、コンポーネントを検索し、各コンポーネント間を接続します。これで各コンポーネントが接続されデータ収集レディ状態になります。

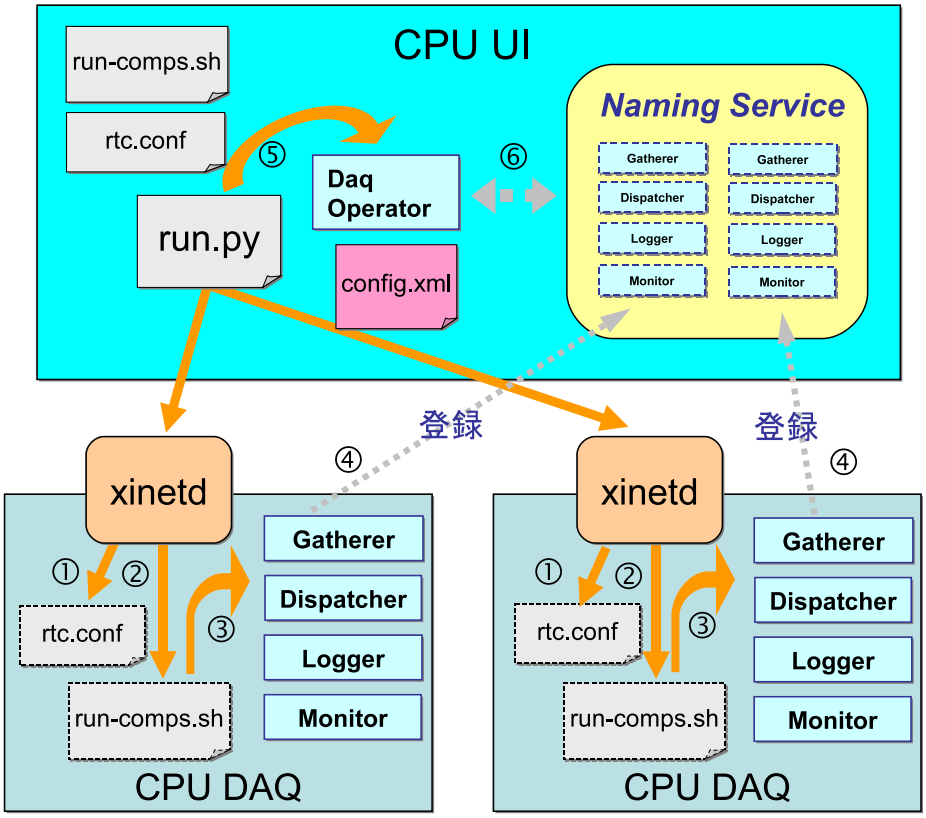

図 7　CPU DAQ 上での DAQ コンポーネントの起動メカニズム。

付録 C　DAQ コンポーネントのコンパイル方法

DAQ コンポーネントのコンパイルに必要なファイルをディスク上にセットするには

- 依存 rpm のインストール
- DaqComponents.tar.gz、SiTCP.tar.gz、manyo.tar.gz、json_spirit_v2.06.tar.gz を展開

が必要です。以下では上記 tar.gz ファイルを /home/daq ディレクトリで展開した場合のコマンドの例を記載します。

```
daq% cd /home/daq
daq% ls -F1 (必要なファイルが格納されているディレクトリがあるかどうか確認)
DaqComponents/
json_spirit_v2.06/
lib/
manyo/
SiTCP/
daq% cd DaqComponents
daq% make
```

で make コマンドを実行します。

付録 D　2009 年 7 月使用時の制約

2009 年 7 月時点での DAQ ミドルウェアを使用する際には次のような制約があります。

D.1　ウェブブラウザによるランの制御

2009 年 1 月時点では、ワーキングデスクトップによるランの制御は準備中であるため、DAY ONE で使用したウェブブラウザをインターフェイスとするコントロールパネルを使用します。DAY ONE と異り、複数台の CPU DAQ を一つのコントロールパネルで制御します。

D.2　ラン番号の管理

2009 年 1 月時点では、ワーキングデスクトップによるランの制御が行われるか自明ではないので、ラン番号の管理は、ウェブサーバ側で行うようにしました。ラン番号はウェブの javascript により自動でインクリメントされます。また、ラン番号を手動で変更することも可能です。

D.3　実験データ保存用ディレクトリの作成、パーミッションの設定

config.xml の ⟨param pid="dirName"⟩ で指定するデータ保存用のディレクトリはランを開始する前に存在する必要があります。またこのディレクトリはランを開始したユーザーで読み書きできる必要があります。ラン毎にこのディレクトリの下に後述の XXXnnnnnnn_YYYYMMDD という名前のディレクトリが作成され、そこにモジュール毎のデータファイルが保存されます。ディレクトリが存在しない場合、書き込めない場合は、エラーとなり、ランはスタートできません。その場合は、ランを終了し適切に設定を行いランを再開する必要があります。

D.4　実験用ユーザアカウント

実験のオペーレーションを行うアカウントとして daq というユーザを想定しています。

付録 E　トラブルシューティング

よくある質問と答をまとめておきます。

Q. ウェブでヒストグラムファイルが表示されない

次の各事項を確認してみてください。

- config.xml の png_output_dir が存在するかどうか、また DAQ コンポーネントを起動したユーザー (このマニュアルでは daq ユーザー) が書けるパーミッションになっているかどうか確認してください。
- config.xml の gnuplot_path で指定したところに gnuplot があるかどうか確認してください。また ldd /path/to/gnuplot 等で gnuplot_path で指定した gnuplot が起動可能かどうか、必要なシェアードライブラリを見つけられているかどうか確認してください。
- config.xml の png_output_dir で指定したディレクトリを見て、png ファイルができているかどうか確認してください。できていれば httpd サーバーがファイルをウェブブラウザに送るときの問題です。/home/daq/www/operatorPanel/histogram シンボリックリンクファイルが png_output_dir で指定したディレクトリを差しているかどうか確認してください。また png_output_dir に httpd のユーザー (通常 nobody) 権限でアクセスできるか確認してください。httpd のエラーログは /var/log/httpd/error_log にあるのでこれもみて参考にしてください。

Q. モニターコンポーネントが作る png ファイルが更新されない

次のことがらを確認してみてください。

1. RedHat が配布している gnuplot は機能が足りていないので使えません。DAQ コンポーネントとともに配布している gnuplot を使ってください。
2. DAQ コンポーネントとともに配布している gnuplot の動作には libpng と gd のライブラリが必要になります。RHEL 5 のインストール方法によってはこれらのライブラリがインストールされていないことがありますので、インストールされていなければ RHEL 5 のディストリビューションメディアからインストールしてください。パッケージ名は
   - libpng
   - gd

   で始まっています (この後ろにバージョン番号がつきます)。ライブラリが足りているか見るには gnuplot を起動してみてgnuplot>というプロンプトがでるかどうかで確認できます。プロンプトがでれば OK です。

3. config.xml のモニターのセクションで正しい gnuplot のフルパスが指定されているかどうか確認してください。
4. 2 次元ヒストグラムの作成に一時ファイルとして /dev/shm を利用しています。デフォルトでは誰でも書けるようになっているはずですがこのディレクトリのパーミッションを確認してください。
5. 2 次元ヒストグラムの一時ファイルのファイルパーミッションを確認してください。ファイル名は /dev/shm/pos_2d_*.dat です。例えば daq ユーザー以外で DAQ コンポーネントを起動したあとに daq ユーザーで DAQ コンポーネントを起動するとこのファイルを消去することができず、2 次元ヒストグラムが更新されない事態が生じます。

Q. DAQ ミドルウェアの gnuplot を起動してグラフを書かせようとしたらエラーメッセージが表示される

DAQ ミドルウェアで配布されている gnuplot を使って画面上にグラフを書こうとしたときに

```
Expected X11 driver: /usr/local/gnuplot/libexec/gnuplot/4.3/gnuplot_x11
Exec failed: No such file or directory
See 'help x11' for more details
```

となることがあります。これは X11 ドライバーが見つからないというエラーでコンパイル時に決定されるデフォルトの場所に X11 ドライバーが見つからなかったためエラーとなったものです。これを避けるには GNUPLOT_DRIVER_DIR 環境変数に gnuplot_x11 があるディレクトリ (/home/daq/gnuplot/libexec/gnuplot/4.3) を指定して gnuplot を起動してください。DAQ ミドルウェアでは X11 の画面上に直接グラフを書くことはありません。

Q. オペレータパネルが表示されない

オペレータパネルのファイルソースはこのマニュアルでは /home/daq/ 以下に設置しています。Apache httpd はファイルの読み取りをnobody ユーザー権限で行います。/home/daq/ ディレクトリの other permission がないとファイルソースを読むことができません。daq ユーザーを登録する際に RHEL 5 の GUI を使用すると、通常 /home/daq/ のパーミッションはrwx------となります。

```
root# ls -ld /home/daq
```

として /home/daq/ ディレクトリのパーミッションを確認してください。rwxr-xr-x のように other group がディレクトリの読み取り、探査を行える (r-x) 必要があります。また /home/daq/www/ ディレクトリ以下のディレクトリも同様のパーミッションになっている必要があります。ファイルについてはrw-r--r--のように読めるようになっている必要があります。

Q. libSock のロードに失敗して動かない

SELinux が無効になっているかどうか確認してください。確認方法については 2.1.2 節をごらんください。

Q. データの読み取りが遅いような気がする

まずディスクにデータを保存しない方法でデータ取得を行いネットワークスイッチ等に問題がないかどうか確かめましょう。

自動調節されるソケットレシーブバッファの最大値が小さ過ぎる場合にはデータの読み取りスピードが遅くなる場合があります。2.1.4 節で述べた方法で最大値を大きくすると性能が改善される場合があります。

NFS 経由でイベントデータを保存しているのであれば NFS がネックになっている可能性があります。

Q. “Configure” 時にオペレータパネルの Gatherer のステータスに “FATAL” が表示される。run.py を起動した端末には、“errStatus.fatalTypes:3” と表示される

これは config.xml の Gatherer に関するパラメータの ⟨param pid="daqId"⟩ が指定されていない際に起きる Fatal Error です。“Unconfigure” ボタンを押して、現在のパラメータランをクリアしてください。その後 config.xml に適切な daqId の値を入れてください。daqId は、

```
<param pid="daqId">0</param>
```

のように記述します。その後、“configure” で新しいパラメータを読み込み、問題がなければ “Begin” が可能です。

Q. “Begin” 時にオペレータパネルの Gatherer のステータスに “FATAL” が表示される。run.py を起動した端末には、“errStatus.fatalTypes:4” と表示される。

これは、Gatherer が config.xml に書かれた ⟨param pid="srcAddr"⟩ のアドレスに接続しようとしたが失敗した際に起きる Fatal Error です。“End” ボタンを押して、ランを終了させて config.xml に書かれているモジュールの IP アドレスが正しいか、モジュールが正常に動作しているか確認してください。その後、“unconfigure”,“configure” で新しいパラメータをを読み込み、問題がなければ “Begin” が可能です。

Q. オペレータパネルの Gatherer のステータスに “FATAL” が表示される。run.py を起動した端末には、“errStatus.fatalTypes:5” と表示される

これは、NEUNET モジュールに対して Gatherer がデータをリクエストして、転送可能なデータ長を取得した際、その値が最大値以上の場合に起きる Fatal Error です。この場合、NEUNET モジュールに問題があることが考えられます。“End” ボタンを押して、ランを終了させてることが可能です。

Q. “Configure” 時にオペレータパネルで Logger のステータスに “FATAL” が表示される。run.py を起動した端末には、“errStatus.fatalTypes:12” と表示される。

これは、Logger が configure 時にデータを保存するディレクトリのチェックを行い失敗した際に起きる Fatal Error です。`config.xml` の ⟨`param pid="dirName"`⟩ に書かれているディレクトリが存在するか確認してください。その後、“unconfigure”, “configure” で新しいパラメータを読み込み、問題がなければ “Begin” が可能です。

Q. “Begin” 時にオペレータパネルで Logger のステータスに “FATAL” が表示される。run.py を起動した端末には、“errStatus.fatalTypes:13” と表示される

これは、Logger が “Begin” 時にディレクトリを作ろうとして失敗した際に起きる Fatal Error です。`config.xml` の ⟨`param pid="dirName"`⟩ に書かれているディレクトリに書き込み許可があるかどうか確認してください。このエラーの場合は、run.py を起動した端末で、“Ctrl + \” と入力して run.py を終了して再起動してください。

Q. ウェブブラウザからの操作ができない

現在、動作が確認されているウェブブラウザは Firefox のみです。Internet Explorer、Safari、Opera、Chrome では動作しないことが確認されています。これらのウェブブラウザでの動作が確認されるまではウェブブラウザとして Firefox を使用してください。

参考文献

[1] 佐藤節夫、GATENET の電気的仕様書、2008 年 3 月 10 日。