DAQ-Middleware の開発

千代浩司
高エネルギー加速器研究機構
素粒子原子核研究所

1　DAQ-Middleware 開発目的

DAQ-Middleware は、産総研で開発されているロボットテクノロジミドルウエア(RT-Middleware)をベースにして KEK 測定器開発室で開発されているネットワークベースのデータ収集ソフトウェアフレームワークである。DAQ-Middleware の開発目的は従来行われてきた「それ以前に書かれたソフトウェアを全部捨てて新たなものを作り上げる」という開発スタイルをやめる点にあり、この目的を達成するためにネットワークベース、コンポーネントベースで、柔軟で再利用可能な DAQ システムを作りあげることにした。適用実験範囲としては、中小規模、あるいは検出器ハードウェアのテストシステムとして用いられることを想定している。

2　DAQ-Middleware の構成

DAQ-Middleware の階層構成図を図 1 に示す。DAQ-Middleware は RT-Middleware をデータ収集用に拡張したものである。 RT-Middleware はコンポーネントベースでロボットシステムおよび組込みシステムを構築するプラットフォームで、その規格は OMG で Robotic Technology Component Specification として定められている。OpenRTM-aist は RT-Middleware の参照実装で、産総研が実装、配布、メンテナンスを行っていて C++, Python, Java で書かれた実装が存在している。DAQ-Middleware ではこのうち C++で書かれたものを使用している。

DAQ-Middleware を使った DAQ システムでは、システムは複数の DAQ コンポーネントから構成される。コンポーネントの構成図を図 2 に示す。コンポーネントは InPort、OutPort、Service Port の口を持つ。 InPort から入って来たデータを図中 Logics のロジックで加工し、 OutPort を使って後段のコンポーネントに送り出す。Service Port は DaqOperator からの指令の送受信に使われる。Logics の部分はユーザーが自分のシステムに必要なロジックをプログラムする。

DAQ-Middleware を使用した DAQ システムの全体構成図を図 3 にしめす。 Gatherer コンポーネントが検出器リードアウトモジュールからデータを読み取り、後段の Dispatcher コンポーネントに送る。 Dispatcher コンポーネントはデータを Logger コンポーネントと Monitor コンポーネントに送る。Logger コンポーネントはデータをディスクに保存する。Monitor コンポーネントはイベントヒストグラム等実験が正常に行われているかどうかがわかるようなヒストグラム等の図を実験者に提示する役割を持つ。ヒストグラム等の図は実装により、ROOT を使って X window system の画面上に出すこともできるし gnuplot 等のグラフ化ソフトウェアを使って画像ファイル化することもできる。各コンポーネントの動作(データ収集開始、停止、ポーズ等)を指示するのが DaqOperator コンポーネントである。 DaqOperator への指示は実験者が HTTP プロトコルを使って Web ブラウザ等から指示を出す。DAQ システムの根幹となるような設定は XML で記述し、DaqOperator が起動時にこれをパーズしシステムを把握する。実験ラン毎に変わるようなパラメータは、XML では各コンポーネントがパーズするには荷が重いので XML で書かれたものを JSON でいったん NV リストに変換後、ファイルに書き出し各コンポーネントがそのファイルを読むことで設定が行われるようになっている。

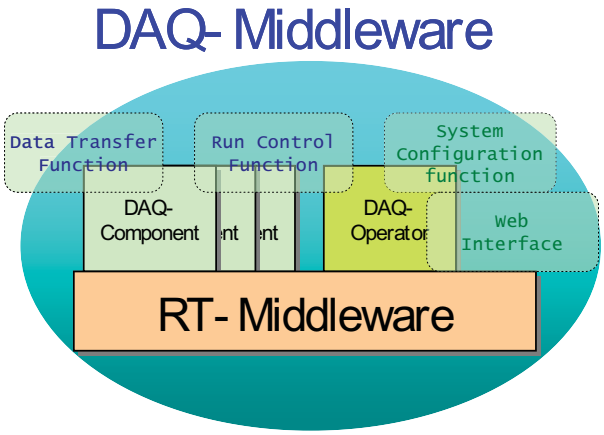

図 1　DAQ-Middleware のソフトウェア階層図。DAQ-Middleware は RT-Middleware を拡張したものである。

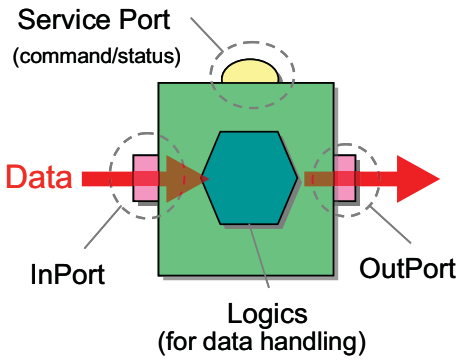

図 2　コンポーネントの構造図。各コンポーネントは InPort と OutPort、および Service Port を持つ。InPort、OutPort がデータ収集に利用される。Service Port は DaqOpereator と通信する。ユーザーが必要としているロジックは Logics で実装する。

実験によってモニターすることなしにデータをディスクに保存したい、あるいはデータをディスクにセーブすることをしないでモニターだけしたいというような状況が考えられるが、その場合 DAQ-Middleware では単にコンポーネントのデータフローを図 4 のように切替えればよいようになっている。

また DAQ コンポーネントは 1 台の計算機でもネットワーク分散させた複数の計算機でも動作可能なようになっている。たとえば 1 台の計算機で Gatherer コンポーネントと Logger コンポーネントを動作されてみたら負荷が大き過ぎてデータの取りこぼしが発生してしまった場合は、計算機を追加して、1 台の計算機で Gatherer コンポーネントを、もう一台の計算機で Logger コンポーネントを動作させ、コンポーネント間はネットワークでデータ転送を行うような構成に切替えることが簡単に行えるようになっている(図 5)。

開発されたコンポーネントは使用目的が合致していれば他の実験でも利用可能である。

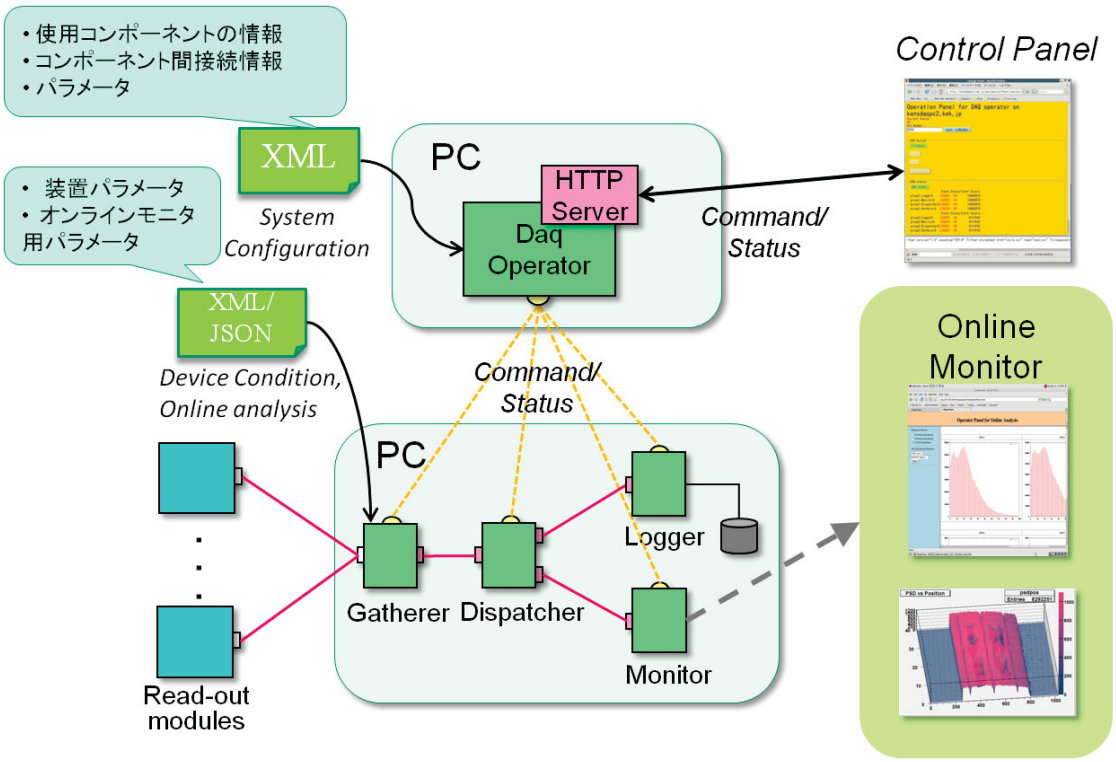

図 3　DAQ-Middleware を使用した DAQ システムの全体図。

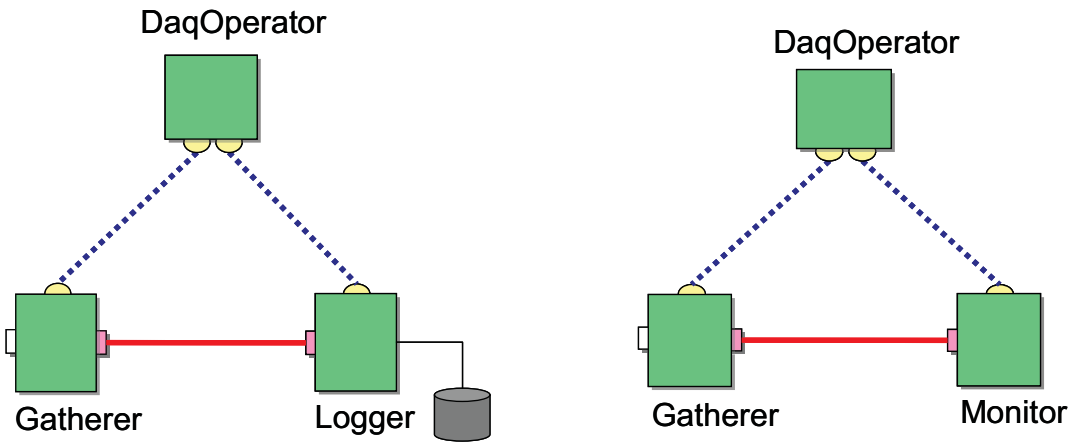

図 4　不要なコンポーネントを動かさないような構成にすることも容易である。左の図はモニターせずにデータをディスクに保存する例。右側の図はデータをディスクにセーブせずモニターだけする例である。

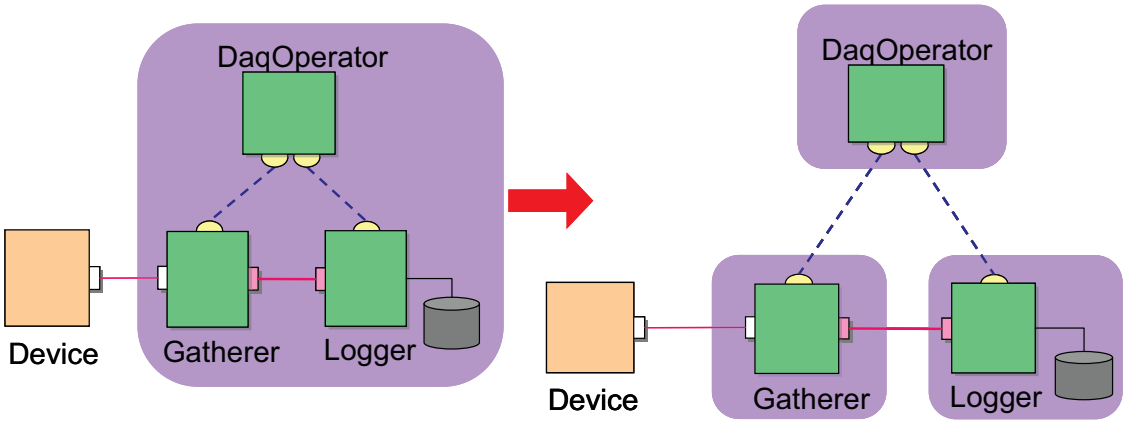

図 5　DAQ-Middleware のネットワーク透過性を利用した負荷分散のしくみ。網掛け部分が 1 台の PC を表す。左のように 1 台の PC で Gatherer と Logger を動かしてみたら Logger の負荷が大きく Gatherer で読み落としがあった場合には右の図のように計算機を追加し負荷を分散する構成に変更することが DAQ-Middleware では容易にできるようになっている。

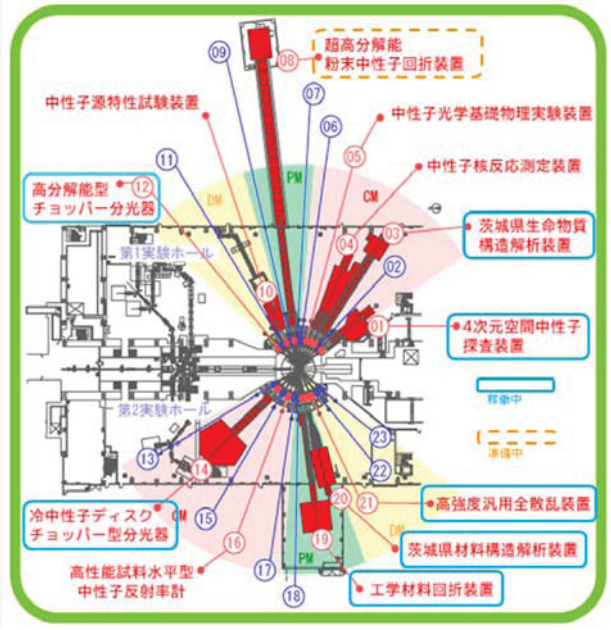

図 6　J-PARC MLF のビームライン。実線は 2009 年 9 月現在で DAQ-Middleware を使ってデータ収集を行っているビームラインを示す。今後 DAQ-Middleware を利用予定のビームラインを点線で囲った。

3　J-PARC/MLF における DAQ-Middleware

DAQ-Middleware のケーススタディ、および問題点の洗い出しのため、DAQ-Middleware は J-PARC/MLF の方々の協力のもと J-PARC/MLF でファーストビームから実際に使われている。

J-PARC/MLF 中性子のビームライン図を図 6 に示す。2009 年 9 現在

- BL01 (4 次元空間中性子探査装置)
- BL03 (茨城県生命物質構造解析装置)
- BL12 (高分解能型チョッパー分光器)
- BL14 (冷中性子ディスクチョッパー型分光器)
- BL19 (工学材料解析装置)
- BL20 (茨城県材料構造解析装置)
- BL21 (高強度汎用全散乱装置)

の 7 つビームラインで DAQ-Middeware を使ってデータ収集が行われており、今後

- BL12 (高分解型チョッパー分光器)

で DAQ-Middleware で使用が予定されている。

J-PARC/MLF では検出器ハードウェアとして

- **PSD 検出器**
- シンチ検出器
- GEM 検出器

が使われている。リードアウトモジュールでは例えば PSD 検出器を使用する場合には NEUNET モジュールが使用されている。 NEUNET モジュールは 8 台の PSD 検出器からのデータを集約しネットワークを通じて TCP で読みだしが行えるモジュールである。また、UDP を使って読みだしに必要な各種設定を行えるようになっている。データの読みだしプロトコルを図 7 に、イベントデータフォーマットを図 8 に示す。

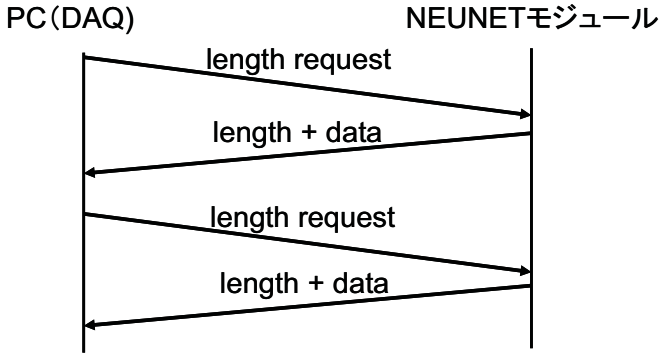

図 7　NEUNET プロトコル。TCP 接続完了後、データ読み出しを行いたい側（読み取り PC）が、読み取りたい長さ（長さリクエスト）を NEUNET モジュールに送る。NEUNET モジュールは送ることが可能なデータ長をまず送り、そのあとデータを送信する。NEUNET モジュールが送信するデータ長は必ず長さリクエストに等しいかまたは小さい。送るデータがない場合にはデータ長と

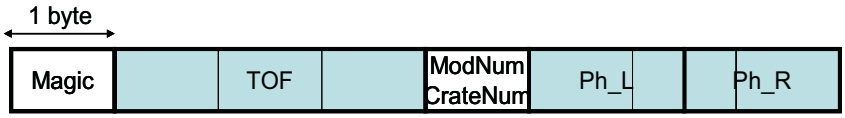

図 8　NEUNET イベントデータフォーマット。1 イベント 8 バイトである。先頭から順にデータの種類を示すマジック（1 バイト）、TOF データ（3 バイト）、モジュール番号およびクレート番号（1 バイト）、パルスハイトデータ（3 バイト）となっている。PSD 検出器は左右両極読み出しであるのでパルスハイトデータはふたつあり、3 バイト中にそれぞれのパルスハイトデータが入っている。

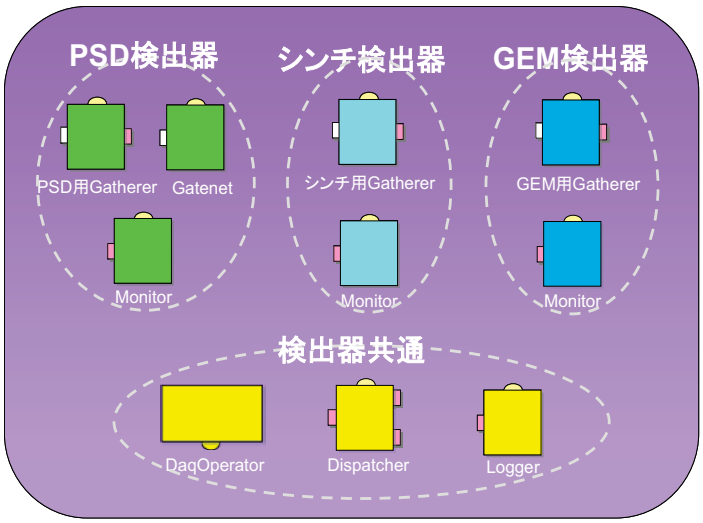

図 9　J-PARC/MLF で使われている DAQ コンポーネント群。リードアウトモジュールが違うと読み取り方式が違うので、リードアウトモジュールごとに専用の Gatherer コンポーネントが必要である。また実験中にモニターしたい事柄が異なるので Monitor コンポーネントも異なっている。それ以外のコンポーネントは共通のものを使用している。

使用する検出器それぞれのリードアウトモジュールからのデータ読みだし方式はそれぞれ異なるため、Gatherer コンポーネント内の読み取りロジックはそれ専用のものを用意する必要がある。このため Gatherer コンポーネントプログラムは検出器ごとに異なるものとなっている。また、各ビームラインで実験中にモニターしたい事柄が違うのでモニターコンポーネントも専用のものが用意されている。その他のコンポーネント(DaqOperator、Dispatcher、Logger)は共通のものが使われている（図 9)。

4　DAQ-Middleware 動作確認環境

動作確認のためにビームタイムを使うことはできないのでハードウェアエミュレータ、およびソフトウェアエミュレータを利用して動作確認を行っている。

ハードウェアエミュレータのハードウェアとしては XILINX Starter Kit(評価ボード)を使っている(写真 1)。このハードウェアの FPGA プログラムにより PSD 検出器で使われている NEUNET プロトコルでデータ読みだしができるようになっている。FPGA のプログラムは物質構造研究所の佐藤節夫氏が作成したものである。この FPGA プログラムのロジックは実験に使用されている NEUNET モジュールのファームウェアロジックと同一のものであるので、NEUNET モジュールファームウェアのロジックの検証も行えるようになっている。またデータ生成レートも設定できるようになっているので、後述の性能測定もこのハードウェアエミュレータを使って行った。

収集するべきデータが全て正常に収集されたかどうかを確認するためにソフトウェアエミュレータ(psd_replay_beam)を作成した。これはすでに取得しファイルとなったイベントデータをデータとして返す TCP サーバーで、プログラム的には通常のソケットプログラムである。このプログラムを使ってテストを行うと、テスト終了後に Logger コンポーネントがデータファイルとしてセーブしたファイルと、このプログラムがデータ元として使ったファイルを比較することにより読み落とし、データ化けがないかどうか確認できる。

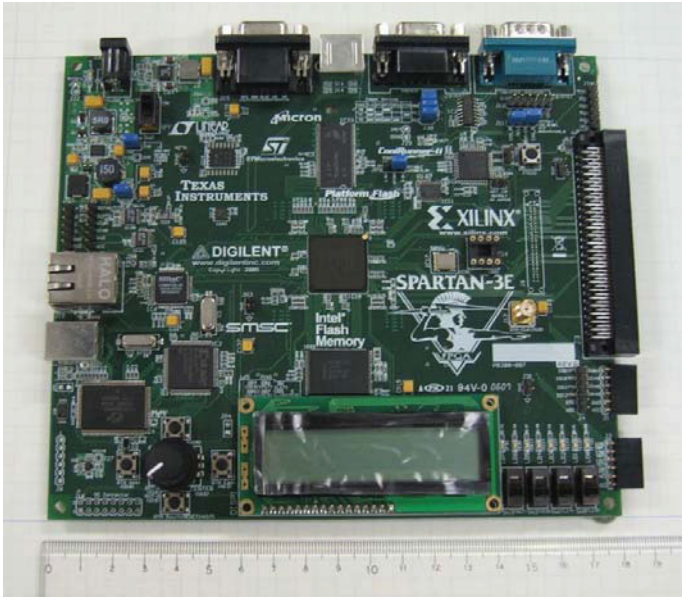

写真 1　ハードウェアエミュレータとして利用している Xilinx Starter KIT（評価ボード）。FPGA 内に NEUNET プロトコルでデータをやり取りできるようなプログラムが入っている。

5　リリースエンジニアリング

DAQ-Middleware の現在の対象 OS は RedHat Enterprise Linux 5 であるが特に OS に依存している部分はないので他の OS でも下記依存ソフトウェアが動作するなら動作するはずである。

DAQ-Middleware の動作には OpenRTM-aist が必要で、OpenRTM-aist の動作には ACE と OmniORB が必要である。 J-PARC/MLF 向けにはモニター用に J-PARC/MLF で開発された Manyo ライブラリが動作することが必要であり、Manyo ライブラリの動作には gnuplot、gsl (GNU Scientific Library)、mxml が必要である。このように DAQ-Middleware の動作のためには複数の依存ソフトウェアをセットアップしなければならない。これらの依存物のセットアップを容易にするためにこれらの依存物は RedHat Enterprise Linux で使われているパッケージ管理方式 RPM にまとめている。またインストール作業を容易にするために既存の Web サーバを yum サーバーとして使えるようにしセットアップが容易に行えるようにした。

開発した DAQ-Middleware のソースは mercurial でリビジョン管理を行っている。各ユーザー向けにリリース版を作成するときには mercurial のリポジトリからソースを取り出し数個のコマンドを投入すれば Web サーバーにアップロードされるようにしている。リリースのバージョン番号は単純にリリースされた年と月をドットでつなぎ、2009.10 版のようにしている。

開発側マンパワーが少なく DAQ システムのセットアップの援助を開発者らが行うことは時間的に難しいので、セットアップは J-PARC/MLF のかたがたが自分達でできるようにするためにインストール・セットアップマニュアルを整備した。 DAQ-Middleware の性能を引き出す OS のセットアップ法もこのマニュアル内に書かれている。

6　性能テスト

PSD 検出器を想定し、DAQ-Middleware の性能テストを行った。 J-PARC/MLF 側から要請として

- PSD 検出器 1 台あたり毎秒 20k イベント(1 イベント 8 バイトなので毎秒 160k バイト)のデータを生成する。リードアウトモジュール(NEUNET モジュール)では 1 台あたり 8 台の PSD 検出器からのデータを集積する。1 台の PC で最大 20 台の NEUNET モジュールからのデータ取得が行えること。
- 1 NEUNET モジュールからのデータを 1 ファイルとしてディスクに保存すること (20 NEUNET モジュールからのデータは合計 20 ファイルのファイルとして保存することになる)。

という二つの要請があった。

この要請が満足されているかどうかをテストするためにデータ生成器として前述の XILINX Starter KIT によるハードウェアエミュレータを使用しテストを行った。ハードウェアエミュレータ 1 台あたり 1 NEUNET モジュールをエミュレートできるのでこのハードウェアエミュレータを 30 台用意しテストを行った。データ生成レートとしては要請にあった最大データ生成レート PSD 検出器あたり毎秒 20k イベント(毎秒 160k バイト)と設定した。

取得したデータはシングルディスク、および 4 台のディスクをストライピングしたディスクに保存し性能を測定した。

ネットワーク読みだし性能にかかわる Linux の OS パラメータとしてソケットレシーブバッファがある。この値は RHEL デフォルトカーネルが許す最大値 4MB に設定し測定を行った。

実験に使用したネットワーク構成図を図 10 に示す。ハードウェアエミュレータは 2 台の Cisco Catalyst 2960G-24TC-L にそれぞれ 20 台、10 台と分散させ、それを 8 ポートの Cisco Catalyst 2960G-8TC-L につなぎ PC はこの 8 ポートのスイッチに接続した。使用した PC は HP の xw8600 で CPU は Quad-Core Xeon が 2 個搭載されているものを使用した。CPU コアは全部で 8 個あることになるが、8 個のコアを持つ計算機が DAQ システムとして J-PARC/MLF で使われていることは無いため、このうち 4 個のコアのみを使用するようにセットして性能テストを行った。

まずデータをディスクにセーブせずネットワーク的に正常にデータが取れているかどうかの確認を行った。その結果を図 11 に示す。読み取りレートは線形に延びており、正常にデータが取得できていることがわかる。

データをディスクにセーブしたときの様子を図 12 に示す。白抜き四角でプロットしているのがディスクに関する Linux パラメータをデフォルトのままでデータ取得を行った場合である。エミュレータの台数を増やしていくと明らかにデータの読み落としがあることがわかる。

Linux カーネルパラメータ dirty_background_ratio を 1 とセットした場合を三角の点で示した。ディスクへの書き込みはユーザープログラム側からは最終的に write()システムコールを発行し、終了する。 write()システム

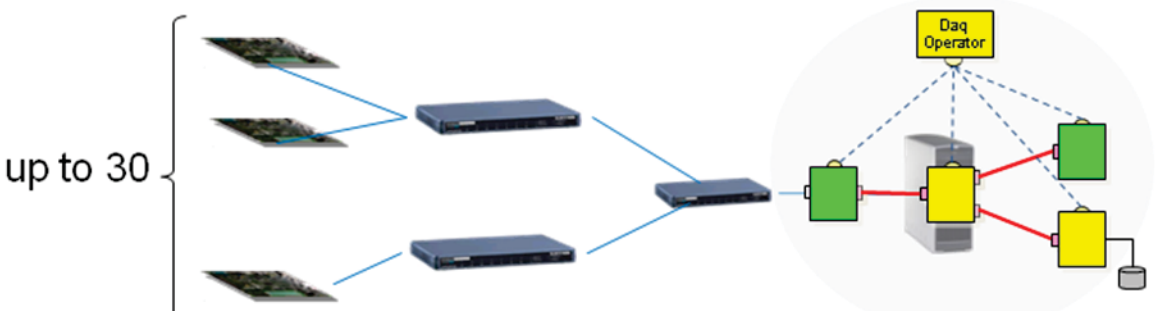

図 10　テストネットワーク構成図

コールはデータをすぐにはディスクには書き出さずメモリー上のキャッシュに書くことにより終了する。メモリ上のデータは揮発的であるからいつまでもそのデータをメモリ上に保存しておくわけにはいかず、あるタイミングでディスクにデータを書くことになる。dirty_background_ratio はそのタイミングを決定するパラメータの一つで、その意味は「ディスクに書いていないメモリ上のキャッシュデータが対総メモリの dirty_background_ratio パーセントまでになったらキャッシュデータをディスクに書き出せ」というものである。例えば 2GB の総メモリがある PC で dirty_background_ratio が 10 の場合、キャッシュされたデータが 2GB の 10 パーセント、200MB に達するまでデータをディスクには書き出さないという動作をする。 Linux カーネルパラメータを 1 とチューニングすることにより DAQ-Middleware を使った DAQ システムが J-PARC/MLF の要請を満たすことが確認できた。

図 13 に JPARC/MLF DAY-1(2008 年 5 月)、茨城県材料構造解析装置（BL20）で DAQ-Middleware を使って収集したデータのオフライン解析結果を示す。図中 30m 秒付近での減少は中性子がアルミに吸着するために起こる現象であり、 DAQ-Middleware を使用した DAQ システムで正常にデータ収集が行えたことを示すものである。

その他の実験でも DAY-1 以降、順調にデータ収集が行われている。

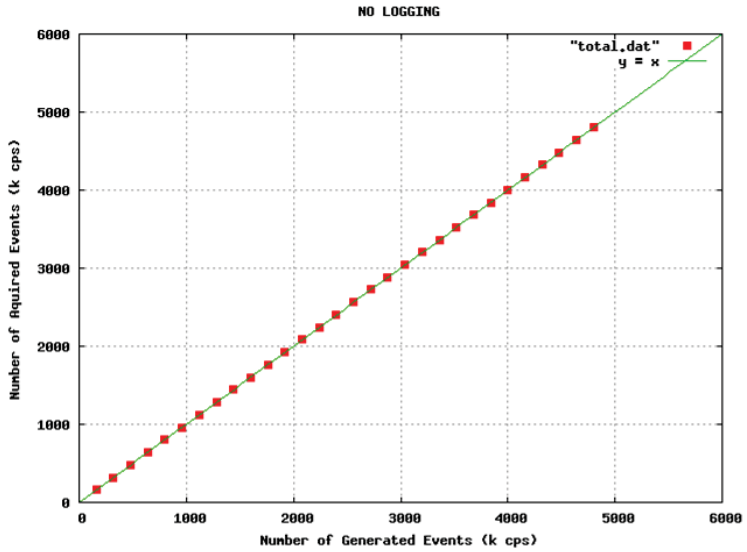

図 11　まずデータをディスクに保存しないでデータ取得を行った。ハードウェアエミュレータを 1 台～30 台に増やしていっておのおので取得できたデータ数をプロットしてある。横軸はエミュレータで生成されたイベント数。縦軸は取得できたイベント数である。きれいに直線にのっていて読み落としがないことがわかる。

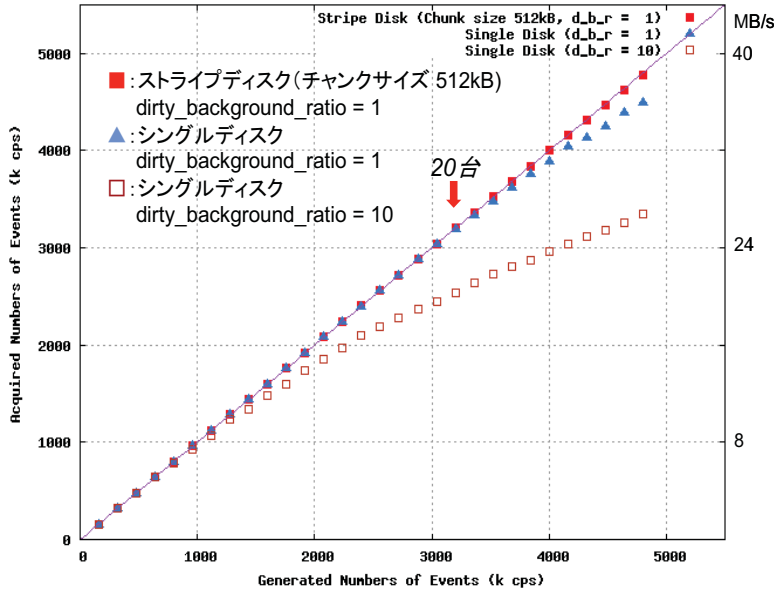

図 12　性能テスト。データをディスクに保存した場合。横軸、縦軸の意味は前の図と同じである。白抜き四角（□）が RedHat Enterprise Linux 5.2 のデフォルトの状態で取得したもの。三角（▲）が dirty_background_ratio を 1 にセットした場合である。dirty_background_ratio を 1 にセットすると J-PARC/MLF の要請を満たすことが確認できた。四角（■）は 4 台のディスクをストライプしたディスクにデータを保存した場合で、この場合 NEUNET30 台程度まで読み落としなくディスクにデータを保存できていることがわかる。

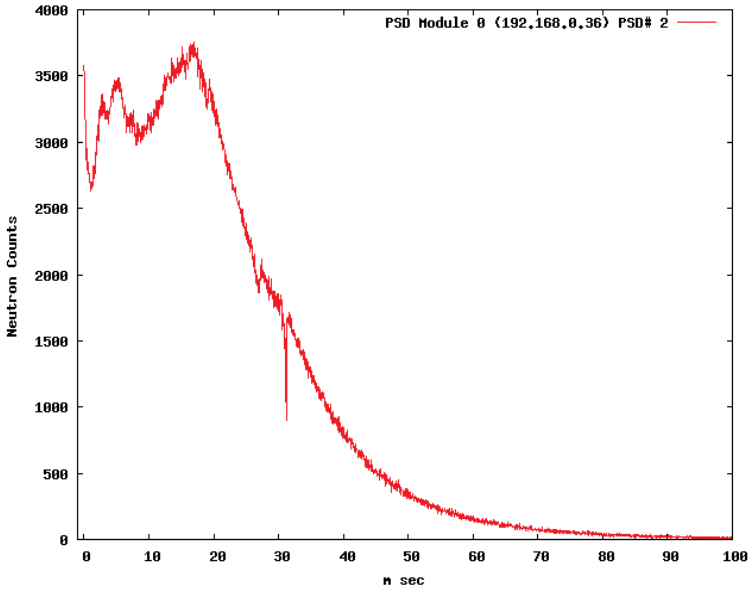

図 13　J-PARC/MLF DAY1 (2008 年 5 月)に茨城県材料構造解析装置（BL20）で取得したデータを簡単にオフライン解析した図。横軸は TOF(ミリ秒)、縦軸は中性子カウント数である。30 ミリ秒付近のカウント数のドロップは中性子がアルミに吸着するため起こる現象であり、これにより DAQ-Middleware で正常にデータ取得ができたことが確認された。

7　今後の予定、展開

2010 年 1 月現在で DAQ-Middleware は以下の実験での使用が検討されようとしている:

- KEK PF BL16A
- JAEA　JRR3 中性子小散乱装置(SANS-J)

DAQ-Middleware 自身の開発予定としては、まずベースとして使用している OpenRTM-aist が近々バージョンアップ(1.0.0) が予定されているのでそれを使用したテスト等がある。

またデータ収集を行う計算機環境に目を向けると、最近の傾向として CPU のクロックをあげての高性能化は頭打ちになってきて、CPU コア数を増やして高性能化をめざす傾向にある。一方その CPU 上で稼動するソフトウェアに目をむけるとマルチコア化によって増大した CPU の能力を有効利用するために VMWare、Linux 上の Xen あるいは KVM に代表されるような OS の仮想化が進んでいる。これらの技術を DAQ-Middleware に適用することにより以下の理由でデータ収集能力を向上させることができると考えられる。 DAQ ミドルウェアにおけるデータ収集は、それぞれ単機能な、DAQ コンポーネント(図 3 中の gatherer、dispatcher 等)と呼ばれるプログラムがネットワーク通信しながら行う。ある DAQ コンポーネントの負荷が高い場合は、その DAQ コンポーネントだけ他の計算機で稼働させるように変更することが容易にできる(図５)。この場合、現状では、隔離された DAQ コンポーネントは他の DAQ コンポーネントとイーサネットを通じて通信を行う。イーサネットのスピードは最高 10Gbps であるが、これを 1 台のたくさんの CPU コアをもつ計算機で他の DAQ コンポーネントと一緒に稼働させれば、ネットワーク通信の部分はメモリー上のコピーで済むのでイーサネットの速度速度に制限されることはなくなることによりデータ収集能力を向上させることができる。今後仮想化技術の適用により実際にデータ収集性能が向上するのかどうかテストする予定である。