Data Acquisition Backbone Core DABC

J. Adamczewski, H.G. Essel, N. Kurz, S. Linev

***Abstract*— For the new experiments at FAIR new concepts of data acquisition systems have to be developed like the distribution of self-triggered, time stamped data streams over high performance networks for event building. The Data Acquisition Backbone Core (DABC) is a software package designed for FAIR detector tests, readout components test, and data flow investigations. All kinds of data channels (front-end systems) are connected by program plug-ins into functional components of DABC like data input, combiner, scheduler, event builder, analysis and storage components.**
After detailed simulations real tests of event building over a switched network (InfiniBand clusters with up to 23 nodes) have been performed. With the DABC software more than 900 MByte/s input and output per node can be achieved meeting the most demanding requirements. The software is ready for the implementation of various test beds needed for the final design of data acquisition systems at FAIR.
The development of key components is supported by the FutureDAQ project of the European Union (FP6 I3HP JRA1).

I. Introduction

A. The New Facilities at GSI

The upcoming new GSI facility FAIR [1] will provide unprecedented accelerator facilities to investigate physics cases in the fields of nuclear structure physics and nuclear astrophysics, hadron physics, physics of nuclear matter, plasma physics, atomic physics, and applied physics.

B. Data Acquisition Concepts

One of the most challenging experiments at FAIR is the Compressed Baryonic Matter experiment CBM [2] because it will produce hundreds of GByte/s primary data rate and needs thousands of CPUs on-line for event filtering.

The main concepts of a data acquisition for such an experiment have been presented in [3]. The key idea was not to use traditional trigger mechanisms but rather process time stamped data streams from self triggered front-end components and combine them through a high speed network. The algorithms for the event definition and event filtering could then run on arbitrary CPUs beyond this building network.

The topology of such a system as sketched in Figure 1 is relatively straight forward. No trigger decisions have to be evaluated between readout and dispatcher levels. Data flow latencies are not critical because the data flow goes only in one direction. Complex and flexible event selection codes can be implemented more conveniently using commodity hardware. Typical high performance data acquisition systems like the ALICE experiment at LHC [4] utilize trigger systems before the Builder network making the traffic between readout and dispatcher levels much more complex. Two key problems must be solved, however: firstly the timing system and secondly the high data rates. Because of the enormous progress in networks, nominal bi-directional data rates in the order of GByte/s per node can be achieved already today. Can such rates be achieved also in data acquisition systems?

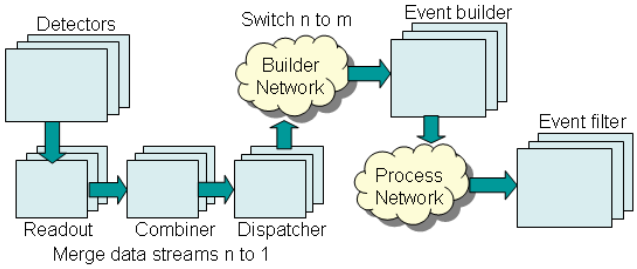

Fig.1. Schematic view of data flow.

The main components of such a data acquisition system are outlined schematically in Figure 1. Data signals from the detectors are digitized in readout modules. A time stamp is added (~ns resolution, not shown). Special time stamps mark time slices between 0.1 and 1 ms long keeping 0.1 to 1 MB of data from some thousands of event fragments. These data blocks can be used later for event building. Combiner boards merge the data streams of several readout modules and send them to dispatcher modules. These modules send data packets defined by time slices through a switched builder network to the event builder nodes (CPU farm) in a way that the data of all detectors of one time slice arrive at the same event builder. To achieve an optimal network utilization the dispatchers must be synchronized and must send their data following a schedule. The schedule is made up and distributed by a master module based on information received from all dispatchers. All scheduler traffic runs over the same builder network. Detailed simulations of such builder networks have been presented in [3].

C. Motivation for an Acquisition Backbone

A data acquisition system as sketched above and meeting performance requirements like the ones of CBM has to be developed in several phases. In the first phase the feasibility of the concept has to be proven, especially the data flow performance and the event building. The design of a final data acquisition production system will start after such demonstration. The selection of the computing hardware for the event filter calculations depends strongly on the

Manuscript received May 11, 2007; revised November 29, 2007.
Authors are with Gesellschaft für Schwerionenforschung, GSI 64291 Darmstadt, Planckstr. 1, Germany. (e-mail: H.Essel@gsi.de, J.Adamczewski@gsi.de, N.Kurz@gsi.de, S.Linev@gsi.de).

developments on the market and will be finalized in a later phase.

For the first phase a test bed for FAIR detectors, readout components, data flow investigations (switched event building) and, last but not least, controls is needed. In this first phase the dispatcher boards are built as PCIexpress cards plugged into standard PCs which perform the event building. The software for that is designated as a data acquisition backbone core (DABC). It is a backbone because it does not include the front-end systems. It is a core which provides interfaces to plug in custom code for the handling of nearly arbitrary front-end systems.

Because of the required flexibility the DABC could be also useful as general purpose data acquisition package.

II. Development of DABC

A. Hardware Setup Use Case

An example of a hardware setup to be handled by DABC is shown in Figure 2. It represents the full data chain as shown in Figure 1 at small scale. Front-end boards typically keep sampling digitizers, whereas combiner and dispatcher boards have fast data links controlled by FPGAs. Prototypes of these boards are under test as they are a first generation of modules needed for the CBM or other experiments. InfiniBand as event building network has been chosen because of its high performance which is a factor of 10 better than Gigabit Ethernet. DABC must prove that it can really operate with such a high throughput. Other networks like Ethernet are supported as well.

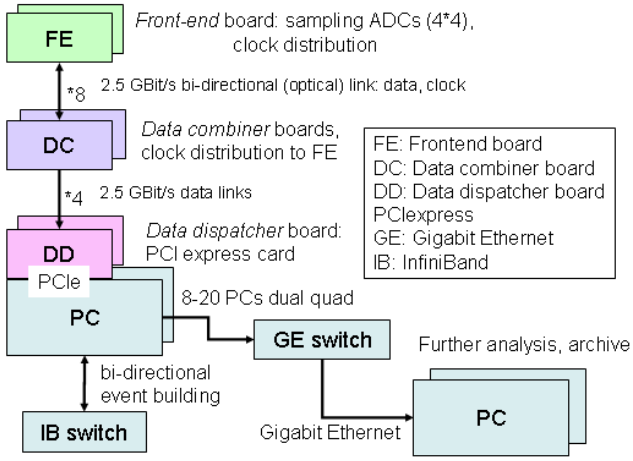

Fig.2. Hardware setup with data chain from front-ends to PCIexpress cards.

B. DABC Design

The topology of software components of DABC are sketched in Figure 3 in a coarse grain. Depending on the performance requirements (data rates of the front-end channels) one may connect one or more front-ends to one DABC node. From a minimum of one PC with Ethernet up to medium sized clusters all kinds of configuration are possible. One may separate input and processing nodes or combine both tasks in each machine using bi-directional links depending on CPU requirements.

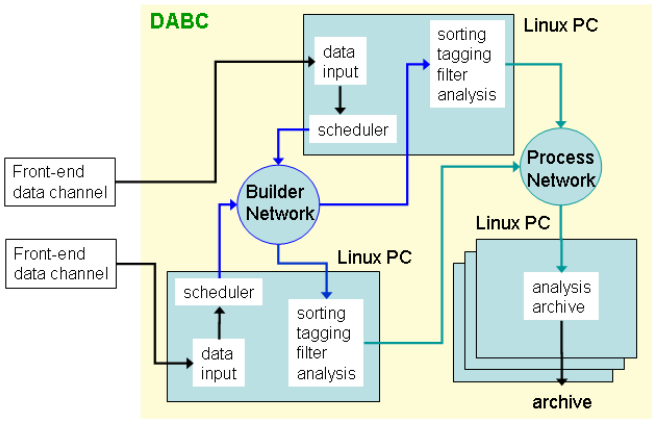

Fig.3. Logical structure of DABC.

The data of one event or time slice from the input channels of each PC, like Ethernet or PCI modules, is sent by a scheduler to one receiver node where it is composed, sorted, tagged and so on. Receivers then may send composed data to further analysis or archive nodes. Using a separate network improves the data throughput.

As mentioned above, the DABC should not be very specific to front-end systems. As a consequence it must provide convenient mechanisms to integrate application specific code, for example to handle front-end hardware or data formats. DABC is written in C++. Plug-ins are implementations of interfaces defined by DABC or subclasses of DABC classes.

Figure 4 shows in more detail a setup of components together with application plug-ins. Threads run on the input nodes to get data (input), combine/transform/check it locally (combiner thread), and send it over network to the event building nodes (scheduler/sender threads) where it is received (receiver threads); events are built (builder thread) and analyzed (analysis thread). At any step of this data flow one may want to optionally store data on disk, mainly for debugging purposes.

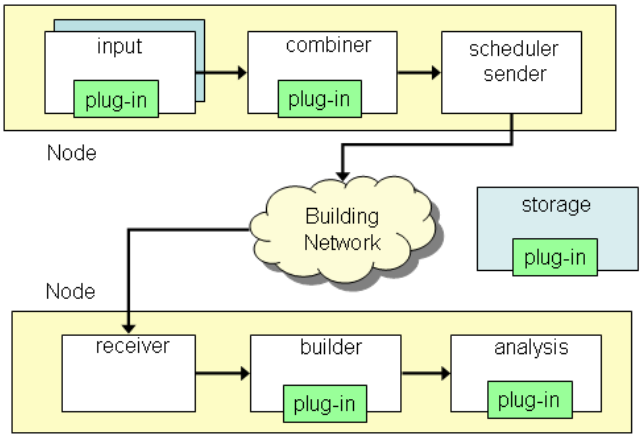

Fig.4. Data flow engine with application plug-ins.

C. Implementation

The basic component is a *module* which gets access to data buffer queues through *ports*, processes them, and outputs them to output ports as shown in Figure 5. The data queues are controlled by a central data manager which allows processing buffers by modules running on the same node without the need of copying. The transport component in this case propagates only references.

If a port is connected to a remote port, the transport and device components do the transfers. Transport and device components are implemented for Ethernet and InfiniBand. Others could be added, like Myrinet. On one node more than one input data stream can be merged into new buffers because modules can have several input ports.

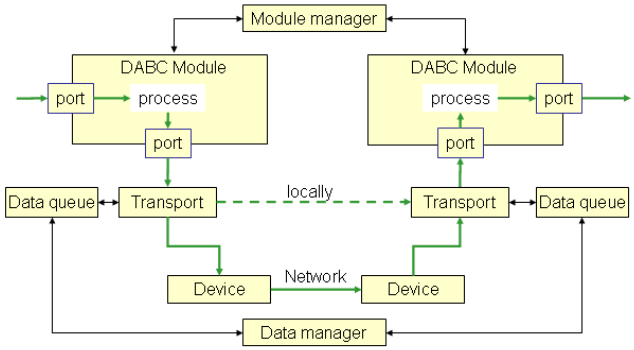

Fig.5. Module communication through ports.

Module components can run in separate threads to utilize multiple CPUs or in one thread, which is faster because no synchronization is needed. Data processing functions of the module are called by an action event manager shown in Figure 6.

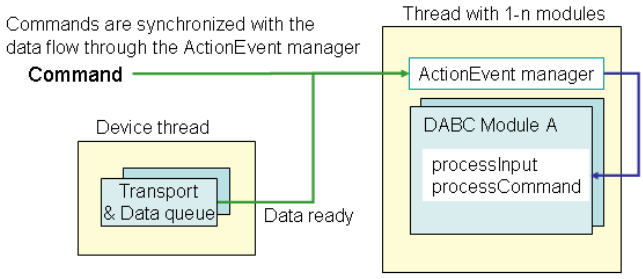

Fig.6. Thread synchronization and module control.

The data flow is controlled by buffer resources (queues) handled through the transports. A device, like a socket device, can control several transports. Once a queue buffer is filled, the transport sends a signal to the *ActionEvent manager*, which in turn calls the *processInput* function of the module associated through a port. Similarly, commands can be sent to the *ActionEvent manager*, which calls the associated module function.

The last thread in the chain, normally the network sender, needs an additional back pressure mechanism to avoid resource exhaustion at the receiver side. This mechanism is implemented in the data flow engine.

D. Data Flow Engine

As shown by the InfiniBand performance evaluations (see next section) this network is very powerful in optimizing non-synchronized data traffic. In these test measurements, the receiver nodes always have enough queue buffers. However, if in real operation the receiver nodes are busy with other calculations like event building, it is necessary to hold the senders. This is achieved by the back pressure mechanism shown in Figure 7.

When an output port is connected to a network device it can no longer be synchronized by signals. Instead the sending node must get feedback information from the receiver. When the consumer thread requests a buffer from its input port the port implicitly - after a number of requests - sends a feedback message to inform the senders of the receive queue status. The producer thread is blocked by the output port if there are not enough resources on the receiver side, or continues if the availability of resources is signalled by a feedback message.

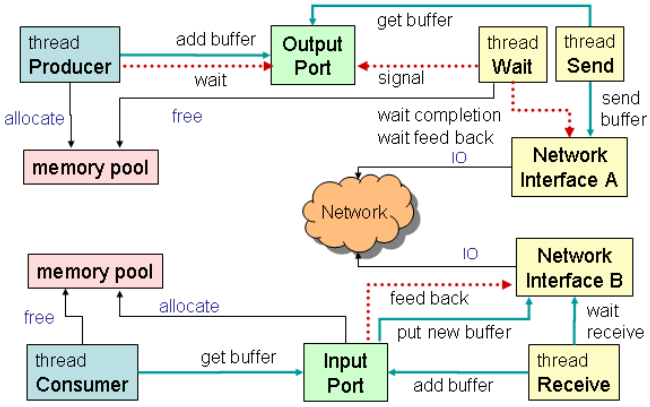

Fig.7. Data flow control by back pressure.

The Wait, Send, and Receive components are completely provided by DABC. Separating the pure send/receive tasks from producers and consumers allows running these components in real-time priority tasks. This is necessary when the senders must follow an exact time schedule with an accuracy of ~10 µs. In this case the new PREEMPT_RT real-time patches including a high precision timer for the Linux kernel (standard in kernel 2.6.21) must be used.

In scheduled transfer mode the feedback may not be needed under normal operation because the schedule avoids conflicts at the receivers. The resource status of the receivers is part of the information sent by all nodes to the master controller calculating and distributing the schedule. Therefore the feedback mechanism can be disabled.

III. Measurements

Before the start of the design of DABC very detailed measurements and tests of hardware and software components were performed.

A. InfiniBand Hardware and Software Setup

We evaluated InfiniBand (IB) as an excellent candidate for event building over network in a first prototype of DABC. InfiniBand provides high data rates with low CPU utilization

(few %), guaranteed delivery, and low latency (few µsec). A small test cluster of four nodes (dual Opteron) was installed at GSI in November 2005. Each node of that cluster is equipped with a Mellanox MHES18-XT InfiniBand Host Channel Adapter (HCA). The nominal data rate of such adapters is 10 Gbit/s in full duplex mode.

All software required to configure and operate InfiniBand networks is collected in the OpenFabric Enterprise Distribution OFED (version used: 1.2.pre), developed by the OpenFabrics Alliance [5]. Several user-level programming interfaces (API) were investigated and tested.

The low level *InfiniBand Verbs* programming interface included in the OFED package provides direct access to the HCA functionality from user space (so-called kernel bypass). Its most important functionalities are non-blocking zero-copy data transfer, remote direct memory access (RDMA) and (unreliable) hardware multicast.

The *User Direct Access Programming Library* (uDAPL), developed by the DAT collaborative [6] was inspired by IB functionality. Therefore it has many similarities with *Verbs* API. Since uDAPL uses a peer-to-peer communication paradigm, multicast is not supported.

The *Message Passing Interface* (MPI) is widely used in the field of parallel computing. It defines an API for fast exchange of data between computing nodes. The MPI over InfiniBand project MVAPICH [7] provides non-blocking zero-copy data transfer and hardware IB multicast.

B. *InfiniBand Benchmarking*

A test application was written to evaluate InfiniBand performance with all mentioned APIs. This test application is capable of generating different kinds of traffic patterns over InfiniBand, in both synchronized and non synchronized mode. In synchronized mode the clocks of the nodes are synchronized. The senders perform time scheduled data transfers. This avoids congestions at the receiver nodes. Figure 8 shows results of an all-to-all traffic pattern, where each node transfers data to all other nodes including itself according to a round-robin schedule with synchronization. The dependency of achieved data rates per node on the packet size for the three APIs is presented.

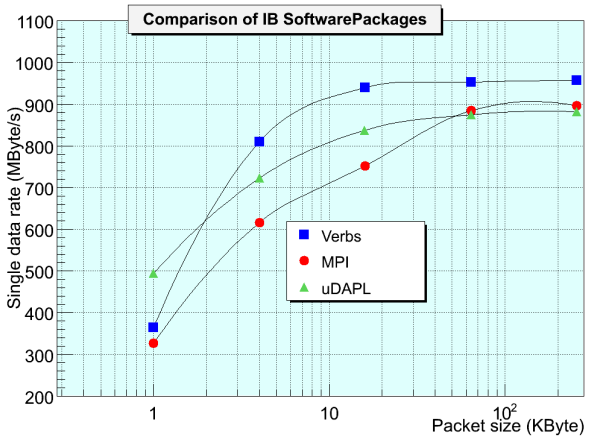

Fig 8. Single node rate performance comparison between different InfiniBand APIs (4 to 4 nodes).

All APIs provide good performance and reach 900 MByte/s for packet sizes greater than 64K. Since *Verbs* has less API overhead, it reaches such data rate already at 8K packet size and a further 950 MByte/s with 16K packets.

C. *Scaling on Larger Cluster*

To extend the measurements to larger clusters, a cooperation with the Forschungszentrum Karlsruhe has been established. Twenty-three double dual-core Opteron machines were available connected by an InfiniBand SilverStorm (QLogic) switch.

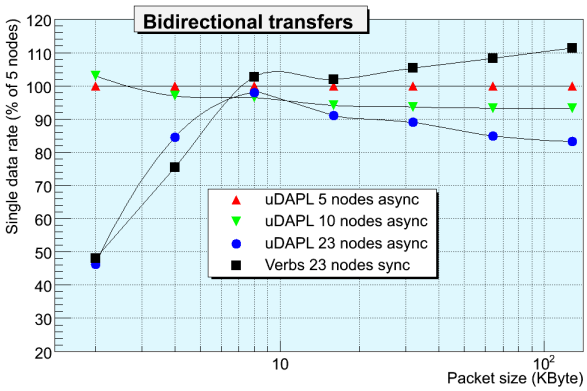

Fig. 9. Network utilization with different number of nodes.

Figure 9 shows the scaling of non synchronized all-to-all transfers from 5 to 10 and 23 nodes, relative to 5 nodes (100%). Remarkably the throughput with 6-10 KB buffers scales with nearly no loss indicating the InfiniBand network is able to optimize such traffic. The throughput falls to 84% at large packet sizes indicating non-resolvable collisions.

The effect of scheduling (synchronizing the senders) is shown as squares in Figure 9 (Verbs 23 nodes sync). With synchronization the throughput can be enhanced for large buffers because now collisions are avoided. Using *Verbs* the synchronized throughput on 23 nodes is even slightly larger than the non synchronized throughput with uDAPL on 5 nodes. On the four nodes at GSI we see no effect of synchronization. One can expect that the benefit of scheduling increases with the number of nodes. Up to 24 nodes can be connected through one single switch chip. Therefore a more challenging test has to be performed on clusters with considerably more nodes.

D. *DABC Event Building Performance*

A data flow chain as shown in Figure 4 with four data generators as data input per node, one combiner per node and one event builder per node has been implemented using the DABC core and plug-in mechanism. The result for InfiniBand and Gigabit Ethernet (GE) is shown in Figure 10 in comparison to the test program (Verbs test, same as in Figure 8). With GE only 70 MByte/s can be achieved, while 900 MByte/s can be achieved with InfiniBand. The CPU utilization is only 50% (dual CPU!). This result proves that the core does not slow down the data throughput as derived with plain test programs (Verbs Test). However, it takes more CPU time, and the packet size must be above 32 KByte.

The buffer size is not a critical parameter because one can always combine or split the data of a time slice in the schedule table. DABC supports the InfiniBand gather DMA mechanism avoiding the need to copy data to combine.

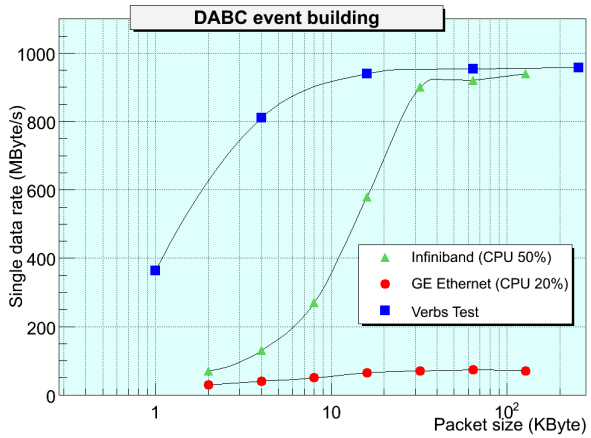

Fig. 10. Event building with four nodes.

Another test proves that the back pressure of the data flow engine as shown in Figure 7 works smoothly (see Figure 11). For this test the consumer threads increase a sleep time from zero to 500 µs. One can see that the transfer time per buffer is exactly the sleeping time with a minimum of 70 µs. That means that no overhead is introduced by the feedback messages. The CPU utilization was about 20%.

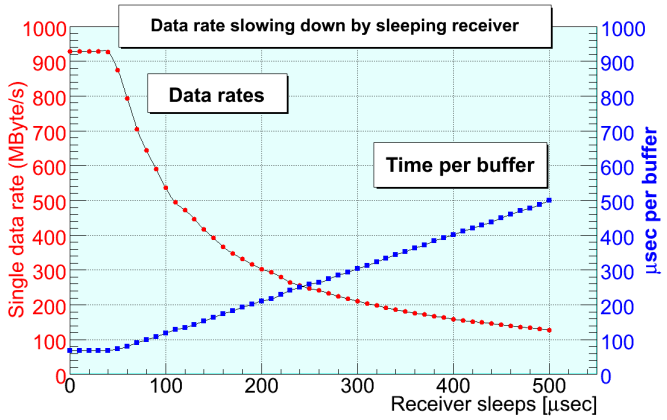

Fig. 11. Smoothly degrading throughput when consumers get busy.

IV. Conclusion

The first version of DABC is able to build events over switched networks at very high data rates as required for an event building network in future FAIR experiments. It provides a test bed for proving various technologies. The choice of the network and its protocol APIs depends on the development of the market. There is no clear preference by performance or functionality.

The software provides interfaces to plug-in custom code for custom data sources. The next task is to build up a full data flow chain from detectors over readout boards, combiner boards, dispatcher boards into DABC. These boards are under construction. We expect first data taking by the end of 2007. Only with this full data chain the goal of phase one can be achieved.

Scaling of performance up to 23 nodes has been proven. To avoid performance drops the data transfers should be synchronized. This can be done within DABC if real-time features of Linux are activated. The next very important milestone is running the software on a cluster with more than one hundred nodes. This will be a real challenge to the scheduler mechanisms. Besides InfiniBand other fast networks like 10 Gigabit Ethernet are supported, but up to now no test beds are available.

For controls the DABC provides the publication of global parameters to DIM (Distributed Information Management) [8] servers. DABC also defines and executes DIM commands. Currently a simple generic Java based graphical user interface is available.

Because of its flexibility the DABC will replace the event builder of the GSI standard data acquisition system MBS [9]. It can also be used as general purpose base for the implementation of dedicated data acquisition systems in the next years.

The DABC web site is
http://www-linux.gsi.de/~dabc/dabc/DABC.php.

V. Acknowledgements

We acknowledge the support of the European Community-Research Infrastructure Activity under the FP6 "Structuring the European Research Area" programme (HadronPhysics, contract number RII3-CT-2004-506078).

We thank Frank Schmitz, Ivan Kondov and Project CampusGrid (www.campusgrid.de) at the Institute for Scientific Computing, Forschungszentrum Karlsruhe for providing resources and support for the large-scale measurements.

References

[1] H.H. Gutbrod (Editor in Chief), *FAIR Baseline Technical Report*, ISBN 3-9811298-0-6, EAN 978-3-9811298-0-9
[2] "CBM technical status report", GSI, January 2005, pp.235
[3] H.G.Essel, "FutureDAQ for CBM: On-line event selection", *IEEE Trans. Nucl. Sci.* Vol.53, No.3, June 2006, pp 677-681
[4] F.Carena et al., *Architecture and Implementation of the ALICE Data-Acquisition System*, CHEP-2006, Vol. 1, ISBN 10: 0230-63016-2, ISBN 13: 978-0230-63016-1, pp 142-145
[5] OpenFabric Alliance OFED [Online].
Available: http:// www.openfabrics.org
[6] uDAPL [Online]. Available: http://www.datcollaborative.org
[7] MVAPITCH [Online].
Available: http://nowlab.cse.ohio-state.edu/projects/mpi-iba
[8] C. Gaspar, "Distribution Information Management system DIM" [Online]. Available: http://dim.web.cern.ch/dim
[9] H.G. Essel et al., "The general purpose data acquisition system MBS", *IEEE Trans. Nucl. Sci.* Vol.47, No.2, April 2000, pp 337-339