Fermi National Accelerator Laboratory

FERMILAB-Conf-99/180

Data Acquisition Systems at Fermilab

M. Votava
On Behalf of the CDF Online, D0 Online, DART Collaboration and
ODS and ESE Departments in the Computing Division

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

July 1999

Published Proceedings of the *11th IEEE NPSS Real Time Conference*,
Santa Fe, New Mexico, June 14-18, 1999

Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CH03000 with the United States Department of Energy

Data Acquisition Systems at Fermilab [1]

M. Votava on behalf of the CDF Online, DØ Online, DART Collaboration, and ODS and ESE Departments in the Computing Division
Fermilab, P.O. Box 500, Batavia, Illinois 60510

Abstract

Experiments at Fermilab require an ongoing program of development for high speed, distributed data acquisition systems. The physics program at the lab has recently started the operation of a Fixed Target run in which experiments are running the DART[1] data acquisition system. The CDF and DØ experiments are preparing for the start of the next Collider run in mid 2000. Each will read out on the order of 1 million detector channels. In parallel, future experiments such as BTeV R&D and Minos have already started prototype and test beam work. BTeV in particular has challenging data acquisition system requirements with an input rate of 1500 Gbytes/sec into Level 1 buffers and a logging rate of 200 Mbytes/sec.

This paper will present a general overview of these data acquisition systems on three fronts – those currently in use, those to be deployed for the Collider Run in 2000, and those proposed for future experiments. It will primarily focus on the CDF and DØ architectures and tools.

I. Introduction

This paper will provide a summary of data acquisition systems at Fermilab – where we are now and where we are headed. It represents work done across the laboratory and collaborating institutions, not just a single department or experiment. We will start with DART, the data acquisition system of the present Fixed Target experiments, noting the hardware architecture and software designs that lead into the design for the next Collider (a.k.a. Run II) systems. The Collider data acquisition systems are being designed and implemented now. Experiences gained with these Collider systems are, in turn, feeding into the faster, more complicated systems of the future. These newer systems are on the scale of the future LHC experiments at CERN [2]. Table 1 lists system requirements for the various data acquisition systems.

Table 1
System Requirements

	KTeV	Run II	BTeV
Level 1 input	100KHz	7.6 MHz	7.6 MHz
Level 3 output	500Hz	30 –75 Hz	4 kHz
Event Size, kbytes	8-10	250	150
Logging Rate, Mbytes/sec	4	10	200
# of Front End Processors	30	100	4000
# of Level 3 Processors	30	200	4000

II. DART

DART has been described in much detail at previous conferences, but will be reviewed here to provide a basis of comparison. It is a scalable, flexible, VME-based system used by 10 fixed target and test beam experiments in the 1996-1997 run. During the subsequent eighteen months when the accelerator was not running to allow for construction of the Main Injector, DART was deployed at test stands for the upcoming Collider run. This deployment was relatively smooth - a credit to the flexibility of the DART design.

This May, the Tevatron was reactivated for the final Fixed Target run, to be completed at the end of 1999. Three of the original experiments (KTeV, HyperCP, and E835) will continue data taking during this period using the original DART system with only small modifications needed to upgrade to new operating system versions and tools.

At the high end, for KTeV, DART supports a 100 Mbytes/sec transfer rate into Level 3 over a 4-plane system with an aggregate logging rate of 20 Mbytes/sec. DART works successfully on small to medium sized experiments as well.

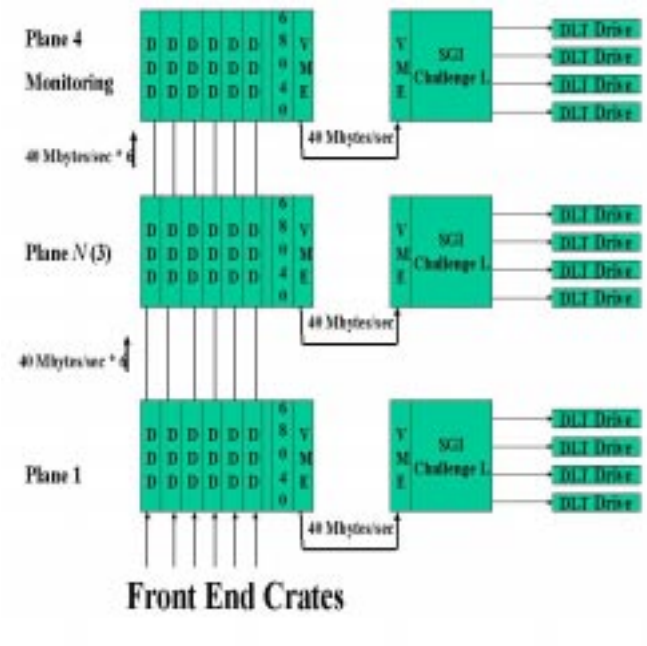

Figure 1: KTeV DART Configuration

The VME crates each have a Motorola 68040 based MVME16x processor running VxWorks 5.1 [3], which is used for monitoring, and control. Data readout is under the control

[1] This work is sponsored by DOE contract NO. DE-AC02-76CH03000

of the UNIX [3] host. On the UNIX side, DART has been ported to IRIX (V6.2) [3] and OSF1 (V4.0) [3] flavors, but higher bandwidths require a host with a VME bus (SGI Challenge L). Data are fed in parallel streams from RS485 cables to Dual Ported Memories (DPMs) through the VSB backplane, are read out across the VME backplane and sent to the host via a commercial VME to VME interconnect board from PTI [4].This configuration can deliver data to the SGI Challenge L host at sustained rates of 40 Mbytes/sec with a DMA transfer size of 64 kbytes. Other experiments using part of DART, including the Collider Experiment test stands, had a low enough event rate that data was simply sent over thin wire ethernet from the front end VME crates to the SGI host. For these systems, cheaper SGI machines (e.g., Indy) were used.

Software filters analyze events on the UNIX host(s) and deliver events to data loggers and or consumer processes. Before being logged to tape, data can be staged to disk, but in all cases, the tape drives are located in the counting rooms of the experimental halls.

The DART software architecture is based on a client/server TCP/IP based model. Three main servers run on a host machine: run control message passing, error logging, and database management. The latter is actually three processes: one each for run configuration, run history, and status monitoring. The location of the data is handled and passed through a buffer management subroutine package - the data itself may or may not be copied. The above software has been written by Fermilab but relies on certain freeware utilities. The underlying databases are based on GDBM [5], and run control messages are parsed using TCL[6].

With the exception of the data logger and main run control program, experimenters, using subroutine libraries provided as part of DART, wrote the remaining code themselves which included filters, data readout, event builders, and consumers. Every process has a unique socket link with each server that it needs to communicate with. All of the user code is written in C. The data logger and servers themselves are written in C++. Most backend GUIs are written using TCL/TK including the main run control panel and data acquisition monitoring displays. Physics monitoring software is based on the Fermilab Histoscope [7] package or PAW [8].

DART ran very well during the 1996-1997 Fixed Target Run. An IRIX 6.5 upgrade of the device drivers is in progress for the Sloan Digital Sky Survey (SDSS). This run has so far started relatively seamlessly and the projected support load from DART experts is anticipated to be about 0.1 FTE.

III. Collider Data Acquisition Systems

The current focus of HEP data acquisition activity at Fermilab is in preparation for the start of the 2000-2005 Collider run of the CDF and DØ experiments. Both experiments have been undergoing a significant upgrade of their detectors during the last 5 years, and the software and hardware architectures have been redesigned to reflect the hardware upgrades.

The collider experiments are high rate, large event experiments. Each year, both expect to record 150–250 Terabytes of raw data. The system requirements for both experiments are listed in Table 2.

Table 2
System Requirements for Collider Experiments

	CDF	DØ
No. of Channels	900,000	1,000,000
Level I input	7,600,000 Hz	7,600,000 Hz
Level I output	40,000 Hz	10,000 Hz
Level II output	300 Hz	1,000 Hz
Level III output	30 – 75 Hz	50 Hz
Event Size	250 kbytes	250 kbytes
Logging Rate, Mbytes/sec	7.5 – 18.75	12.5

Both experiments will implement a three tiered trigger system. Levels 1 and 2 operate on partial events in the VME crates. In contrast, a Level 3 decision is made on a full event that has been assembled on a higher level processing farm. Events are sent directly from Level 3 to the data loggers and consumers. The data acquisition hardware architectures of the CDF and DØ experiments are shown in Figures 2 and 3.

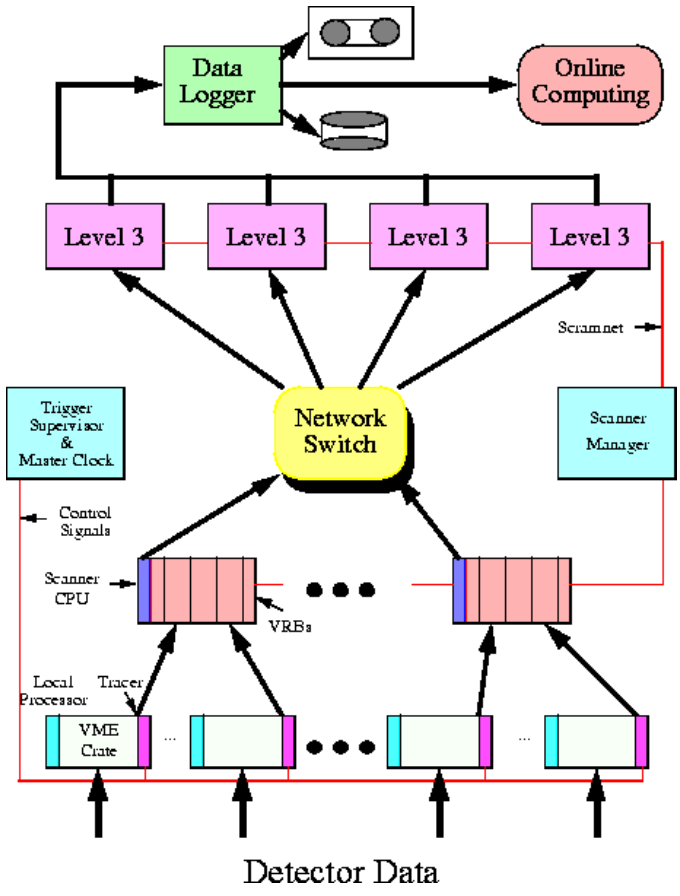

Figure 2: CDF Run II Architecture

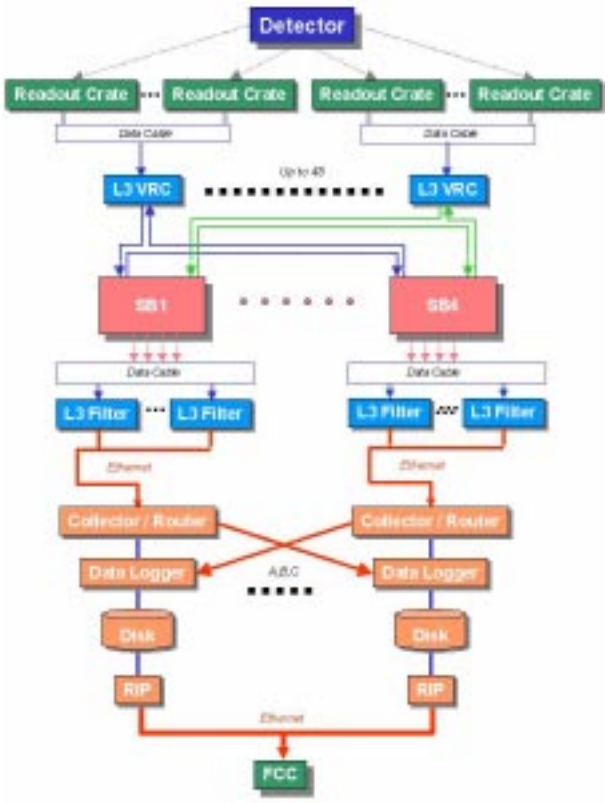

Figure 3: DØ Run II Architecture

A. Front End Crates

Clearly the detector elements of each experiment are unique and individual boards have been engineered with a particular experiment's needs in mind. However, we can see many similarities in the two systems' architectures.

Table 3
Front End Architectures

	CDF	DØ
# of digitizing VME crates	100	70
Data path to concentrators	TAXI Links GLinks	Brown data cable
# of concentrators	16 VME crates with MVME2603	8 multi-processor NT PCs
User Code Language	C	C/C++
Control Path	Ethernet, SCRAMNET	Ethernet, 1553
Data path to Level 3	ATM switch	Fiber -> Brown data cables
Communication Protocol with host	SmartSockets and CORBA	DØme and EPICS

The digitizing electronics of both systems reside in VME VIPA crates. CDF has approximately 100 VME crates feeding data into 16 additional concentrators. The 16 concentrators themselves drive an ATM switch to feed into the Level 3 processors [9].

DØ has about 70 digitizing crates that feed data over a custom data cable into 8 multi-processor PC concentrators (running NT [3]). The cable, developed at Brown University for Run I, supports a bandwidth of 48 Mbytes/sec. Each concentrator accumulates data from two such data cables and pumps them over a 100 Mb/sec fiber using the fiber channel protocol. The 8 fibers connect to 4 farm segment controllers. Based on header information in the data itself (e.g., event number and front end crate ID), each controller determines if its segment is processing the event, and, if so, will reroute the event to the correct L3 node. If the data is not destined for that farm segment or if the segment controller's buffers are full, the data will be transmitted to the next segment. The last segment will return unwanted data to the concentrator where it will be recirculated [10].

Both experiments have made extensive use of the J3 connectors in the VIPA crates to pass control information. Data transfer in CDF and DØ systems takes place over dedicated links, with the final transfer (following a Level 2 accept) across a VME backplane. CDF makes use of 64 bit VME transfers. While CDF uses a commercial CPU for the VME readout, DØ uses a custom readout controller (VBD).

Several VME crates at each experiment need an embedded processor board (to control the remaining cards, to interface to the host, to read out, etc.). Both have selected a variation of a Motorola Power PC board, with some legacy 68k boards. All processors are running VxWorks v5.3. Fermilab has invested over 1 man-year into making the kernel stable and porting the cross development environment to other platforms - IRIX 6.x and Linux 2.0 - being used by the experiments [11].

In addition to transferring data, the VME backplane is also employed to read status information and download configurations to the readout boards. This type of information is transmitted back to the host through the controller board via Ethernet. Various monitoring and message passing software also needs to run on the VME controllers. The g++ C++ compiler and the standard C++ libraries provided with the VxWorks cross compiler environment were insufficient for DØ's needs. We have expanded the tool set to include support for the Egcs C++ [12] compiler with the STL, but it has yet to be tested.

Each experiment has selected different software protocols to communicate to the front-end boards. DØ is using both EPICS [13] and an experiment specific message passing system called DØme [14]. DØme is a client/server package, which provides a common, well-defined way to exchange information and data among all DØ data acquisition applications. It is using threads to parallelize sending and receiving messages through the TCP/IP protocol. It is based on ACE [15], a multi-platform communication/transport freeware layer. On the other hand, CDF uses CORBA implementations for front-end diagnostic software (more about that later) and a commercial package called Smart Sockets [16] for status and control.

B. Diagnostic software

Engineers at Fermilab have developed several of the new VME boards for Run II. A suite of diagnostic software has been written for these boards in collaboration with CDF called CDFVME [17]. CDFVME is in use by several CDF board developers. The framework provides a user-extensible, Java-based GUI in which users can test one or more boards in one or more VME crates. It provides an easy interface in which users can run a series of tests in batch mode. The client software has been running on both Unix and NT nodes.

C. Slow controls

The two experiments have different implementations for slow controls and alarms. CDF has purchased commercial software called FIX-Dynamics [18] to monitor the high voltage, pressures, and temperatures. Each detector subgroup (muon, cot, svx, etc) will have a local FIX-Dynamics monitoring node running NT. Each subgroup has specific code to monitor that piece of the detector, whether it is via CAMAC [19] or a VME interface. These subsystems then feed into an NT server node stationed in the main control room. None of the slow control data will feed into the main data stream, and all of the electronics for slow controls live in different crates than the main data acquisition system. The data collected will be stored in the online ORACLE [3] database for archiving and later retrieval.

Conversely, DØ runs EPICS on the VME controllers in the same crates as the digitizing electronics. There are some "control only" crates which optionally have either 1553 bus interfaces or "vertical interconnect" VME bus extenders, allowing memory-mapped access to VME crates. If the bandwidth permits, monitoring information is sent over VME. If not, there are provisions for a 1394 [20] (Firewire [3]) connection between the VME board and local processor. The monitoring applications are tied to the EPICS IOC software in the VME crates, basically acting as a data source for the control path.

D. Level III Processing/Farms

Event data is transferred from the front ends to the Level 3 processing farms either over a high speed switching network (CDF) or a token ring on Brown data cable segments (DØ).

Table 4
Level 3 Architecture

	CDF	DØ
Level 3 Processor Type	Commodity PCs	Commodity PCs
Operating System	Linux	WinNT
# of Level 3 processors	150-200	48
User Code Language	C/C++	C++
Control Path	Ethernet	Ethernet
Data path to Loggers	Fast Ethernet	Fast Ethernet
Communication Protocol with host	Smart Sockets	DØme/Ace

In each case, Level 3 farms are a series of scalable PC sub-farms connected to the concentrators. Both experiments will purchase commodity PCs early next year, but it is not yet known if they will be the same. CDF is interested in those that support Linux [3], while DØ needs a configuration to attach to custom hardware via the PCI bus

CDF will have 16 Level 3 sub-farms that are connected to the Concentrators via the ATM switch. Each sub-farm consists of a single receiver node, which routes the event over Fast Ethernet to 6 or more processing nodes.

The DØ configuration has 16 PCs connected to each of 4 segment branches, but this can be scalable to 48 nodes/data cable segment In the initial configuration, DØ will have 48 nodes total.

E. Data Logging/Consumers

Once the Level 3 processing farms have analyzed the data, good events are made available to the data loggers. Events will be first staged to disk files with enough disk space available to hold more than 8 hours worth of data, but the long term storage is remote. Disk files are copied through fiber to a robotic system located in the Feynman Computing Center, up to 2 miles away from the experiment. The robotic system itself is EMASS [21] but the tape drive technology to be used for Run II has not yet been selected. We are waiting to test SONY AIT 2 and Exabyte Mammoth 2 tape drives, and in the meantime we are using AIT 1s and Mammoth 1s.

At CDF, each Level 3 sub-farm has an I/O node that is connected via (4-8 parallel) separate fast Ethernet ports to a single SGI consumer/server node [22] with a large RAID disk array attached as a data buffer. The files are read from the buffer and logged into the EMASS robot using a tape package called FTT [23].

DØ has a more parallel architecture with the advantages of scalability and reliability. The Level 3 nodes feed *N* (currently 3) OSF1 data logging nodes, which also connect to a RAID disk array. Data is copied into the EMASS robot with a higher level Fermilab software package called ENSTORE [24].

Table 5
Data Logging

	CDF	DØ
Disk type	RAID Array	RAID Array
Disk buffering capacity	8 hours	8 hours
Disk -> tape Interface	FTT to directly attached tape	ENSTORE

Both experiments are using the ROOT [8] analysis system for online monitoring of the event data on IRIX, OSF1 and Linux, and are using the shared memory and network server functions of the system [25]. Prototype consumers have been developed and are being used to test portions of the data acquisition system.

F. Run Control

The run control design and implementation of the two experiments is in progress. In this arena, the experiments are more different than alike – each has adopted different tool sets with which to develop. It starts with the run control language

itself. CDF has adopted JAVA as the primary language for run control and diagnostics, while DØ is following a Python/C++ [26] path. Since collaborators often bring in the hardware of home institutions, both experiments were interested in ease of portability between operating system platforms. It was also important to select a language that their user base would find comfortable.

A second fundamental difference is the message passing system. CDF has purchased a commercial package called SmartSockets for its message passing needs (e.g., run control messages, error messages, status messages). The SmartSockets software met the CDF requirements: operates in a publish/subscribe paradigm (publishers and subscribers can come and go without affecting the server itself), multiple servers can run in parallel to distribute the load, Java bindings are available, and client software is supported on the necessary platforms (VxWorks port is in progress).

On the other hand, DØ has opted to write their own package, DØme. DØme has been designed to be the single protocol to pass both messages and data (from Level 3 and beyond) through the DØ data acquisition system. This package is written in C++, but wrappers are provided allowing it to be used directly in the Python scripting language as well with a very similar interface.

The choice of ORACLE as their online database system is one area that both experiments have in common. Run II will mark the first large scale application of online commercial databases at Fermilab. Both experiments are currently designing their online databases and beginning to understand the requirements for the API layer. Table 6 summarizes the run control host machine

Table 6
Host Machines

	CDF	DØ
Run control language	Java	Python/C++
Message Passing System	SmartSockets	DØme
Run Control Host Machine	IRIX	OSF1
Online Databases	ORACLE	ORACLE
Monitoring Display	IRIX/Linux	WinNT/Linux
User Code	C++/Java	C++/Python

IV. Possible BTeV Architecture

The post Run II direction of HEP experiments at Fermilab has yet to be determined. A research and development project has been approved to test the feasibility of doing a *b* physics experiment, BTeV. Work is already underway to develop and test new detectors, to prototype read buffers and controllers, and to architect a fast and affordable data acquisition system [27].

Data acquisition rates for the proposed BTeV detector are comparable to those at the LHC. A distinctive feature of the BTeV system is the digitization and transmission of data at the beam crossing frequency. This is done because the first level trigger is based on tracking in the pixel system and trigger latencies are larger than can be accommodated by typical front-end pipeline buffers. Moving all buffers off the detector allows the use of parallel, asynchronous, high latency triggers. This comes at the expense of high data rates, which may exceed one Terabyte/sec into first level buffers.

Following the first level, the BTeV architecture is similar to other large scale systems. The Level 1 accept rate is expected to be 100-200 kHz at an event size of approximately 150 kbytes. This results in 20-30 Gbytes/sec at the event builder and processors. Input and output links to the data switch are ring based to facilitate load balancing. Large input buffers are used to convert the event arrival rate to a uniform distribution for high switch bandwidth utilization.

Each buffer module must support an input data rate of approximately 800 Mbytes/sec. These rates make it unlikely that conventional backplane bus standards will be used to any large extent. All data connections are made using point-to-point serial or narrow parallel links. Control connections use lower speed serial links. Event data is time stamped and transmitted asynchronously, so there is no fast control or synchronization requirement beyond the beam crossing clock at the front-end.

There are an estimated 4000 processors in the second level trigger. The current BTeV proposal uses a staged readout approach where the processors request event data in several steps. This will be analyzed to determine if the cost benefit of the reduced switch size is offset by the expanded buffer and control requirements.

Much of the functionality of current centralized processor boards will be distributed as embedded processing in the individual data acquisition modules. This includes initialization, slow control, network interface and test features. These smaller processing elements may not support all features of the traditional high level OS environment. While the software load of individual processors will be reduced, the total software requirement will increase significantly. BTeV is expected to include over 10,000 processors in various applications, ranging from simple link control to event analysis.

V. Summary

This paper has described the current data acquisition activity at Fermilab from small-scale systems to future experiments that are relying on increasing performance trends to continue in order to run at all. In terms of hardware, we have seen that the commercial market is increasingly influencing data acquisition electronics. As the Level 3 decisions become more CPU intensive, Level 3 farms are growing to hundreds of CPUs. With larger numbers, it is imperative to adopt high performance/price machines. The commercial PC market is clearly at the forefront. Some of the savings in hardware, though, are offset by additional manpower costs. Configuring, controlling, monitoring, and managing several hundred machines is a much more complex task than a small number of very powerful machines. Run II experience will help architect the much larger demands of the LHC.

DA systems are relying on the backplanes for data transfer less and less. Ethernet rates are expanding to support the slow

control and monitoring load, so that all front end crates have ethernet access. The trend is to give each individual board access as well. Data paths themselves are moving away from a bussed system like VME and toward high speed switch networks.

The sheer number of nodes needed in such data acquisition systems can also have a big impact in software license costs. Experiments are continuing to rely on freeware for operating systems (Linux) and tools.

The resources needed to develop the data acquisition systems for our Fermilab experiments are large and there tends to be more collaboration between experiments. The success and longevity of DART has proven that several experiments can share pieces of a common data acquisition system if their requirements are well thought out and accommodation is made for local customization. Historically, the collider experiments have taken completely different paths for their data acquisition systems, but they are beginning to move from complete customization to include planning for centralized support and reduction of their maintenance load .

The dividing line between experiment online and offline systems continues to fade, with the use of high-speed fiber links and remote robotic tape systems, the affordability of significant computing near the detector, the commonality in software development tools and infrastructure code, and the benefits of the full reconstruction happening in near real time being realized in previous data taking runs. As an example, BTeV is not even planning to write raw data to tape, but will archive an already reduced data summary set. In another example, offline and online software developers are taking advantage of sharing a common infrastructure – database management systems, code management schemes, analysis systems etc.

VI. Acknowledgements

This paper represents the hard work of many people across the lab and its collaborating institutions. A special thanks to all of the co-authors and/or reviewers: from Brown University: Gordon Watts; from Fermilab: Ed Barsotti, Mark Bowden, Stu Fuess, Carmenita Moore, Jim Patrick, and Ruth Pordes; from Massachusetts Institute of Technology: Christoph Paus; and from Yale University: Colin Gay. These people are a small subset of all the collaborators (too numerous to mention) who have contributed their industry and creativity to the monumental effort of engineering and deploying these data acquisition systems.

VII. References

[1] R. Pordes et al, Fermilab's DART DA System, Proceedings of CHEP94.

[2] The Large Hadron Collider. http://wwwlhc01.cern.ch

[3] UNIX™ of AT&T; VxWorks™ of Wind River Systems Inc. http://www.wrs.com; IRIX™ of Silicon Graphics, Inc.; OSF1™, Digital UNIX™ of Compaq Corp.; Windows NT™ of Microsoft Corp.; ORACLE™ of ORACLE Corp.; Linux™ of Linus Torvalds; Firewire™ of Apple Computer Inc.

[4] Performance Computer Company. 315 Science Parkway, Rochester, NY 14620 USA.

[5] GDBM: The GNU database manager. http://www.gnu.org/software/gdbm.

[6] Tcl and Tk Toolkit: J.Ousterhout, Addison Wesley Computing series.

[7] Histoscope: Interactive Histogramming Tool. http://www.fnal.gov/fermitools/abstracts/histoscope/

[8] ROOT: http://root.cern.ch; Physics Analysis Workstation: http://wwwinfo.cern.ch/asd/paw.

[9] Christoph Paus, et al. "Event Builder and Level 3 Trigger at the CDF Experiment", Abstract 104, RT99

[10] Gennady Briskin, et al. "The DZero Level 3 Trigger/Data Acquisition and its Real Time Control", Abstract 164, RT99

[11] David Berg ,"VxWorks Support for Collider Run II Data Acquisition at Fermilab", Abstract 151, RT99

[12] Egcs Compiler: http://www.cygnus.com

[13] Experimental Physics and Industrial Control System. http://www.aps.anl.gov/asd/controls/epics

[14] Carmenita Moore, et al. "Multi-threaded Message and Event Routing for the DZero Online System", Abstract 155, RT99

[15] The Adaptive Communication Environment. http://www.cs.wustl.edu/~schmidt/ACE.html

[16] SmartSockets: commercial publish-subscribe middleware by Talarian Corporation. http://www.talarian.com

[17] Yuyi Guo, et al. "CDFVME – Software Framework for Testing VME Boards", Abstract 107, RT99

[18] FIX-Dynamics: http://www.intellution.com

[19] Dave Slimmer, "CAMAC Driver Support for Windows NT4 and LINUX", Abstract 128, RT99

[20] IEEE Standard for a High Performance Serial Bus, IEEE Std 1394-1995.

[21] EMASS robot, now ADIC (AML/2). http://www.adic.com

[22] Makoto Shimojima, et al. "Consumer-Server/Logger System for the CDF Experiment", Abstract 127,RT99

[23] FTT: Fermi Tape Tools. http://www.fnal.gov/fermitools/abstracts/ftt/abstract.html

[24] ENSTORE tape handling facility. http://www-hppc.fnal.gov/enstore/design.html.

[25] Makoto Shimojima, et al. "Online Monitoring in the Upcoming Fermilab Tevatron Run II", Abstract 123, RT99

[26] Python is an interpreted, interactive, OO language. http://www.python.org

[27] Mingshen Gao, "SUMAC: A Monitor and Control Tree for Multi-FPGA Systems ". Abstract 152, RT99.