BTeV Trigger/DAQ Innovations

Margaret Votava on behalf of the BTeV Collaboration

Abstract—BTeV was a proposed high-energy physics (HEP) collider experiment designed for the study of B -physics and CP Violation at the Tevatron at Fermilab. BTeV included a large-scale, high-speed trigger and data acquisition (DAQ) system, reading data from the detector at 500 Gbytes/sec and writing data to mass storage at a rate of 200 Mbytes/sec. The design of the trigger/DAQ system was innovative while remaining realistic in terms of technical feasibility, schedule and cost. This paper will give an overview of the BTeV trigger/DAQ architecture, highlight some of the technical challenges, and describe the approach that was used to solve these challenges.

Index Terms—Data acquisition, large-scale systems, real time systems, triggering.

I. INTRODUCTION

AT the heart of the BTeV experiment [1] was an ambitious and innovative online trigger and data acquisition (DAQ) system. The design of this system resulted from nearly a decade of research and development. Three phases of R&D occurred during this time. The first phase involved extensive physics and detector simulations to establish trigger and DAQ requirements and a baseline design. This was followed by prototyping of hardware components and software algorithms to establish trigger and DAQ performance metrics, and provide a basis for cost estimates. The third phase was characterized by several design changes that resulted from optimizations to achieve reductions in cost of construction and maintenance, while at the same time improving online performance and addressing the commissioning and operational needs of the experiment.

The primary purpose of BTeV R&D was to design a dedicated B -physics experiment that would make precision measurements of beauty and charm decays to challenge the Standard Model and perform an exhaustive search for physics beyond the Standard Model. It was designed to find B events at high rate and with high efficiency while operating in a harsh hadron-collider environment. An important feature that distinguished BTeV from other experiments with similar goals was that it was designed to find these B events in the first stage of triggering (referred

to as Level 1 or L1) by taking advantage of the key property that differentiates B (and charm) particles from other types of particles, namely their characteristic long lifetime. BTeV was to have begun data taking at the end of this decade, but was cancelled due to HEP budget constraints.

II. OVERVIEW

BTeV based its trigger and DAQ design on four design principles: a consistent trigger strategy that was based on finding detached vertices at all stages of triggering, a thorough analysis of every Tevatron bunch crossing to search for evidence of B decays, use of high-resolution three-dimensional tracking data even at the first trigger stage to maximize the number of B decays for physics analyses, and deployment of fault tolerant and fault adaptive computing at all stages of the trigger and DAQ.

The trigger was designed to analyze multiple interactions in every Tevatron bunch crossing by reconstructing tracks and vertices to search for evidence of a particle-decay within a few hundred microns to a few centimeters away from a primary interaction vertex. This was done by calculating impact parameters for tracks relative to a primary vertex, and selecting those events that satisfied a “cut” requiring a minimum number of large impact-parameter tracks. Typical hadron-collider experiments use a fairly simple “first level” trigger with very low fixed latency. This is done to reduce input data rates to subsequent trigger levels to a “manageable rate,” allowing sufficient time to perform a detailed analysis of events. The drawback to this approach is that it limits the types of final states that can be selected, thereby limiting B -physics analyses. Trigger strategies that require the presence of specific final-state particles, such as muons, or demand the presence of a few high- p_t hadrons, are examples of this. By avoiding these restrictive trigger strategies at L1 and by exploiting the characteristic lifetimes of B particles at the first and all subsequent trigger levels, the BTeV trigger was able to maintain high efficiency for B events throughout the entire event selection process. To accommodate the greatly increased and variable latency at L1, BTeV used a large system of memory buffers.

The BTeV trigger was a three-level hierarchical trigger system (shown in Fig. 1). Each level was designed to contribute to the reconstruction of events, and successive levels imposed more and more refined selection criteria to select B events and reject light-quark background events.

Manuscript received June 5, 2005. This work was supported in part by Universities Research Association Inc. under Contract No. DE-AC02-76CH03000 with the United States Department of Energy.

M. Votava is with the Fermi National Accelerator Laboratory, Batavia, IL 60510 USA (phone: 630-840-2625; e-mail: Votava@fnal.gov).

The trigger was designed to run at an initial (peak) luminosity of $2 \times 10^{32} \text{ cm}^{-2} \text{ s}^{-1}$ with a 132 ns, 264 ns, or 396 ns bunch crossing interval, corresponding to an average of 2, 4, or 6 interactions per crossing, respectively. The L1 trigger operated at the Tevatron bunch-crossing rate, and was considered to be the most demanding part of the trigger system. L1 subsystems included the L1 pixel trigger, L1 muon trigger, and Global Level 1 (GL1). The primary trigger for BTeV was the pixel trigger, while the muon trigger was to be used as an independent verification of pixel trigger performance. GL1 collected results from both subsystems and imposed the L1 trigger decision.

The three-level trigger system shown in Figure 1 shows the BTeV detector, front-end electronics, L1 and Level-2/3 trigger systems, Level-1 and Level-2/3 buffers, DAQ highway switch, and data-logging system. With data read out from front-end electronics at the Tevatron bunch-crossing rate and a *variable* L1 trigger latency of order 1 ms, the buffers for detector data had to be able to accommodate a large amount of data while trigger calculations were being performed. The data rate from the detector was estimated to be $\sim 500 \text{ Gbytes/s}$ for a crossing rate of 2.5 MHz, an average of 6 interactions/crossing, and an average event size of 250 Kbytes. The concept of an eightfold “highway” architecture was introduced to reduce the full data rate to 62.5 Gbytes/s into each highway, and to enable the use of low-cost components in the DAQ, such as commercially available Ethernet switches. The Level-1 buffers, which held the detector data, were designed to buffer approximately 2×10^6 bunch crossings in total,

corresponding to 800 ms of L1 trigger decision time and corresponding to approximately three orders of magnitude more than the average L1 processing time. The DAQ highway switch was designed to route data to a combined Level-2 and Level-3 trigger system, referred to as the “L2/3 trigger”, for bunch crossings that satisfied L1 selection criteria, as determined by GL1. Data for bunch crossings that satisfied L2 and L3 selection criteria were to be logged on archival media such as disk drives.

In our baseline design, the L1 trigger reduced the data rate by a factor of 50 by rejecting background events while retaining *B* events for L2 processing. L2 improved the track and vertex reconstruction by reviewing tracking data used at L1, and by including additional tracking data if necessary. L2 reduced the data rate by rejecting at least 90% of events accepted by L1. At L3 all of the data for a particular bunch crossing was available, which would permit a complete analysis, much like an offline analysis, of the data. L3 imposed the final trigger selection criteria and reduced the data rate by 50% producing an output rate of 2.5 KHz and an estimated 200 MB/s, which was achieved by data compression and the elimination of data not needed for physics analyses, including some of the raw event data. See Table 1 for the data rates at each trigger level.

The primary trigger for BTeV was the L1 pixel trigger, also referred to as the “L1 vertex trigger.” It consisted of four stages: pattern recognition, track reconstruction, vertex reconstruction, and impact-parameter calculations to find detached vertices that were the basis for the L1 trigger decision. In the baseline design the bulk of the pattern

recognition was performed by field-programmable gate arrays (FPGAs), which are ideal for performing large numbers of rudimentary calculations in parallel. The remaining L1 calculations were performed by commodity general-purpose processors, referred to as “L1 farm processors” or “L1 worker nodes”. Subsequent trigger levels, L2 and L3, also relied on general-purpose processors to perform trigger calculations.

Due to the flexibility of the trigger and DAQ system, design changes were relatively easy to accommodate, although such changes were not approved without considerable simulation and analysis effort and detailed reviews. For example, we investigated two alternatives for pattern recognition that significantly reduced the processing time in the L1 worker nodes, thereby reducing the cost of this subsystem. The first alternative, a “hash sorter” [2] was

Figure 1. BTeV three-level trigger architecture..

implemented in an FPGA, thereby reducing the load on L1 worker nodes. The second alternative reduced the load on the worker nodes by moving calculations that completed the pattern recognition for pixel data from the worker nodes to FPGAs. The first of these alternatives was included in the baseline design, while studies for the second alternative were underway when BTeV was terminated in February, 2005.

Table 1. Event Rates

	Frequency	Event Size	Data Rate
Into L1	2.5 MHz	200 KB	500 GB/sec
Into L2/3	50 KHz	250 KB	12.5 GB/sec
Archived	2.5 KHz	80 KB	200 MB/sec

The design of the BTeV trigger and DAQ evolved over time. Many of the changes were introduced to accommodate changes that were made to the pixel detector, and in some cases a change to a trigger algorithm enabled significant improvements in the design of the pixel detector system, thereby demonstrating the importance of concurrent development of the two systems [3]. The BTeV trigger and DAQ designs were also influenced by technology changes as new hardware became available. For several trigger and DAQ subsystems we abandoned custom hardware in favor of commodity hardware, which contributed to a significant savings in material costs and labor costs that would have been required to maintain the trigger and DAQ during operations of the BTeV experiment.

The proposed BTeV detector consisted of 6 separate subdetectors: pixel, silicon strips, straw tubes, RICH, EMCAL and muon, with the pixel detector dominating the channel count (see Table 2).

Table 2. Channel Count

Subsystem	Channels	DCB Subsystems
Pixel	23M	10
Strips	118K	2
Straws	54K	6
RICH	144K	4
EMCAL	10K	2
Muon	37K	4

Several features of the DAQ are noteworthy. First, the architecture of the detector readout was unique in that all the data for every crossing was brought off the detector, digitized in subdetector-specific front-end boards, and stored in the L1 buffers. These data were sent via point-to-point copper connections to data combiner boards (DCBs) before being sent to the first level trigger over optical links. The DCB design was common across subdetectors. Two variants were designed, one to be used with pixel and strip detector front-end boards and another to be used with all other front-end boards. This architecture enhanced the trigger/DAQ design in that 1) much of the trigger could be done in software, which made it possible to use many commodity components in the system; and 2) the DCBs

provided a single type of data entry hardware into the trigger/DAQ system, making it possible to have a unified design that would ease maintenance tasks and reduce long-term support of the system.

A second advantage of the architecture was that data collected in the DCBs could be routed to several independent, parallel data paths, which were called *highways*. This reduced the control overhead on each individual highway and grouped data coming out of the L1 buffers into larger packets for better network performance.

The following sections will outline some of the features of the online system.

III. DATA FLOW

Data highways were a key feature of the BTeV trigger/DAQ architecture. The concept of routing data through parallel and independent paths, either via “round robin” or some other mechanism, originated with the desire to eliminate custom switches from the architecture, thereby reducing the cost of the BTeV DAQ. With the highway scheme, an individual highway had a lower data rate and a larger event fragment size compared to a monolithic trigger/DAQ system. These two aspects of the architecture made a commodity implementation viable.

BTeV was able to capitalize on other more subtle benefits of the highway architecture. In particular, the highway architecture was important because it allowed for a flexible construction schedule for the trigger/DAQ system. We could build the trigger and DAQ in two stages – four highways to be purchased in one fiscal year, and the remaining highways to be purchased at a later time. The highway scheme allowed for future expansion in capacity with no additional cost other than the cost of more commodity components and the effort to reconfigure the system. No redesign of the trigger/DAQ system would be necessary for future upgrades.

A. DCBs

The data combiner boards were a custom component and the sole interface between the subdetectors’ front-end electronics and the common data readout system. A given DCB was connected to 8 highways and would route data to the appropriate highway based on internal routing tables. The highways remained independent through all levels of the trigger until data was pushed out of the L3 trigger farm and into mass storage.

Fig. 2 shows the data and control flow in/out of a DCB. Each board contained a commercial CPU and fast Ethernet interface for control, and low speed data readout for diagnostics and commissioning. The DCBs received precise clock information from the timing system. A possible feature that was being considered for the DCBs was to use the timing information to dynamically reset and reconfigure all DCBs in response to a certain condition. When a change to the DCB configuration was needed, the dynamic reset/reconfiguration could be enacted for a specified point in time, such as a specific bunch crossing number. For

example, if a catastrophic highway failure occurred, a command could be sent to the DCBs telling them to route to 7 highways instead of 8 starting at crossing number “N.”

We were investigating the possibility of routing data to a different highway for each “Tevatron turn” (approximately 20 microseconds of data), as opposed to routing each bunch crossing to a different highway. This approach would simplify the routing, but increase the length of time the data would remain in a DCB, exposing it to a higher single-event-upset rate. It would also mean that data from pixel detector sources would need to be time ordered, a functionality that was being designed into the pixel L1 trigger. Turn routing also had the advantage of load balancing, since the data carried over a single highway was the same on average for each of the highways. This was true no matter how many highways were in service. If a highway dropped out, the remaining highways would absorb the load evenly.

B. L1 Trigger

The BTeV L1 trigger was a key feature of the experiment, and was considered crucial to the success of BTeV. Detector data from the DCBs were sent to two locations, the L1 trigger and Level-1 buffers. Data from the pixel and muon detectors were sent to the L1 trigger, since these were the data that were used at L1 to select *B* events for subsequent trigger processing in L2 and L3. Since all data had to be buffered in Level-1 buffers for subsequent trigger processing, the L1 trigger also wrote the pixel and muon data, as well as intermediate and final L1 trigger results, to Level-1 buffers.

Before being sent to Level-1 buffers, the pixel detector data were sent to FPGA pixel preprocessors that performed several operations on the data such as time-stamp expansion (adding additional timing data to individual pixel hits); pixel hit clustering; and *x*, *y*, and *z* coordinate

determination for pixel hits. The processed data were then sent to Level-1 buffers and to an FPGA-based segment tracker that executed the track-segment finding stage of the L1 trigger algorithm [3]. Track segments that were found at this stage were routed by a network switch to a single worker node in the L1 trigger farm, where the track and vertex finding stage of the trigger algorithm was performed. Results from this stage of the calculation were written to Level-1 buffers, and summarized results were sent to a GL1 (Global Level 1) processor that was responsible for making the L1 trigger decision. Individual trigger decisions were stored as a list of accepted bunch crossing numbers in the Information Transfer Control Hardware (ITCH), which broadcasted “accept messages” to all Level-1 buffers so that data could be sent to the L2 trigger.

The L1 muon trigger was going to be built with the same hardware that was going to be used for the L1 pixel trigger. The GL1 trigger would then receive the summarized trigger results for both the pixel trigger and the muon trigger to produce the L1 trigger decision. For the L1 pixel trigger, the worker nodes that were to perform the track and vertex finding stage of the L1 trigger were estimated to require 200 microseconds on average per bunch crossing for a processor purchased in 2008. For an interaction rate of 2.5 MHz a total of approximately 500 CPUs (or 250 dual-CPU worker nodes) would be required, assuming a 50% duty time on each CPU.

We measured the execution time of the L1 trigger algorithm on several general-purpose processors that are commonly used in commercial desktop and server PCs. The processors we considered were the following:

- 3.2 GHz Intel Pentium 4 with 800 MHz front-side bus
- 2.4 GHz Intel Xeon with 400 MHz front-side bus
- 2.2 GHz AMD Opteron on a Sun Sunfire V20z
- 2.0 GHz IBM970 PowerPC on an Apple PowerMac G5

The L1 code was compiled with the Intel C/C++ compiler version 8.0 on Linux for the x86 platforms, and the IBM XL C/C++ compiler on Mac OS X for the PowerPC platform. Our baseline design consisted of a total of 528 8.0 GHz IBM 970 PowerPC CPU's, or equivalent. For example, a dual-core 4.0 GHz processor was considered equivalent to an 8.0 GHz single-core CPU. Compared to a 2.0 GHz Apple PowerMac G5 (available in 2005), we were assuming a factor of four increase in processing power by the time we needed to purchase the L1 worker nodes in 2008.

C. Event Building

Details of the event building design were under development when BTeV was terminated in 2005. We were investigating an implementation in which the L2 trigger would receive complete events by sending all data for a particular bunch crossing to a single L2/3 worker node. It was estimated that 40% of the data were needed to make an L2 decision. We concluded that it would be more cost effective to add the extra network capacity to send all data

for a bunch crossing at once, and reduce the complexity of the handshaking that would be involved in sending the data fragments over several discrete intervals.

D. L2/3 Trigger

From the L1 buffers, data were sent over Gigabit Ethernet links to the L2/3 processor farm, which had worker nodes that were used to execute both the L2 and L3 trigger algorithms. The L2/3 farm consisted of commodity processors that were subdivided into highways. Highways were self-contained, with lower bandwidth communications channels connecting the highways. Each highway consisted of 96 worker nodes, each with dual CPUs.

To reduce the control overhead and complexity of the DAQ software, we designed the event-building switch with enough capacity to send a complete event to each L2 worker node. Worker nodes would notify the ITCH when they were ready for data. The ITCH would assign them a particular crossing number. All data from that crossing would be sent to that worker node.

Worker nodes themselves were grouped into manageable units in a highway. Each group was controlled by a regional manager consisting of 12 worker nodes. A regional manager was responsible for configuring its associated worker nodes, fanning out control commands, collecting status information, caching database information such as versions of trigger algorithms used in the trigger, and handling regional faults.

E. Mass Storage

One of our goals for mass storage of data was to operate a “tapeless” archiving system. However, the choice of storage media that would be used by BTeV was deferred due to the ever-changing price/capacity/performance matrix of disk storage versus tape storage.

In the baseline design, the online system would write to disk and not to tape. Fermilab is already successfully running large disk caching farms as front-ends to the mass storage tape farms using a dCache [4] framework. “Backup” copies of the original data would be kept on university disk farms as part of the analysis process. We believed this to be a viable model as the capability of handling multiple persistent copies of data was being implemented in the dCache project.

IV. FAULT TOLERANCE AND MITIGATION

The BTeV Real Time Embedded Systems (RTES) project was born from a need to address concerns that arose from a project review that was conducted in 2000:

“Given the very complex nature of this system where thousands of events are simultaneously and asynchronously cooking, issues of data integrity, robustness, and monitoring are critically important and have the capacity to cripple a design if not dealt with at the outset... BTeV [needs to] supply the necessary level of “self-awareness” in the trigger system.”

The RTES project group was formed and funded through a

Figure 3. Design Time and Runtime aspects of RTES. Multiple domain-specific languages are used at the Design and Analysis level, to represent system organization and behavior. Synthesis produces runtime artifacts (codes, scripts). Runtime middleware (VLAs, ARMORs) provides detection and mitigation of faults.

5-year NSF ITR grant. This research group is a collaborative effort between electrical engineers, computer scientists and high energy physicists. The group is researching ways to increase the reliability of high-performance, heterogeneous, real-time systems.

The BTeV trigger was used as a model system for RTES. In order to satisfy the requirements of this trigger and address the problems associated with it, RTES was assembling a fault handling subsystem to be used by all components in the trigger and DAQ. This subsystem would accurately identify problems and compensate for them, including application related activities such as changing algorithm thresholds and overall system activities such as load shifting. As many recovery procedures as possible were to be automated. Operators and system developers must be able to easily incorporate new procedures or policies into the system. The operators must be able to easily select error handling policies. A detailed record of observations and actions must be kept to facilitate reproduction of analysis results and to identify long-term trends.

The RTES approach employs both design-time modeling and run-time capabilities. Fig. 3 shows the overall project perspective. The technologies introduced by RTES and discussed below are ARMORs for L2/3 nodes and overall management nodes, VLAs for the embedded processors and specific monitoring tasks at L2/3, and GME for system modeling and configuration. Each of these apply to different aspects of the trigger and all of them must work together.

A graphical modeling tool called the Generic Modeling Environment (GME) was used to apply model integrated computing methods to the specification and analysis of the

system. GME supports domain-specific languages for representing differing dimensions of the system organization and behavior, as well as the meta-modeling capabilities for defining new domain-specific languages

Adaptive Reconfigurable Mobile Objects for Reliability (ARMORs) are multithreaded processes internally structured around objects ("elements") which provide functions or services. Every ARMOR process contains a basic set of elements that provide core functionality, *e.g.*, reliable point-to-point inter-ARMOR messaging, and ARMOR-state checkpointing. A modular, event-driven architecture permits developers to customize an ARMOR process's functionality and fault-tolerance services (detection and recovery) according to the application's needs. The self-checking ARMOR runtime environment includes: one fault-tolerance manager (FTM) to initialize the ARMOR-based system configuration, to maintain registration information on all ARMORs and applications, and to initiate recovery from ARMOR and node failures; one heartbeat ARMOR (HB) to detect failures in the FTM; one daemon ARMOR per node, acting as a gateway for ARMOR-to-ARMOR communication; and any number of execution ARMORs, which launch and monitor application processes.

Very lightweight agents (VLAs) are responsible for providing a lightweight, adaptive layer of fault detection and mitigation. Agents consist of a relatively few lines of code embedded within applications, or acting as independent processes, which monitor hardware and software integrity. VLAs can be proactive or reactive, depending on their scope.

As a first exercise in demonstrating their methodologies, the RTES group developed a prototype for the BTeV L1 trigger processing farm, using digital signal processors (DSPs) of the type being studied by BTeV at that time. This prototype was demonstrated at the Super Computing 2003 conference [5].

A formal review of the project software was conducted after the conference. In response to this review, it was recognized that GME would need to serve a large number of domains and submodels: system description, message modeling, fault mitigation behavior, run control behavior, user interface definition, *etc.* For the ARMOR software, it was recognized that custom (application-specific) ARMOR elements needed to be easy to create, and that package organization and version control were vital. These recommendations were incorporated prior to our second demonstration project.

As a next effort, the RTES project undertook to prototype the L2/3 trigger commodity processor farm. Hardware for this farm was accumulated by BTeV, recycled from other computing farms at Fermilab. The farm was heterogeneous (dual-CPU P3's and P4's; at least 4 different speeds), and several exhibited hardware problems. It was an excellent setting for demonstrating reliable software infrastructures.

Several different test configurations were developed, employing variously 3, 12, and 54 worker nodes (performing L2/3 processing), with additional regional and global control nodes. As each node was a dual-CPU machine, these configurations allowed the testing of ARMORs, VLAs, and GME-derived communications and control to be applied to over 120 processors. The "16 node" (12 workers) and "65 node" (54 workers) systems were demonstrated at the Real Time and Embedded Technology and Applications Symposium, March 2005 [6].

Developing and supporting both "16 node" and "65 node" configurations had several beneficial effects. Effective software engineering was vital to minimize the number of instances where the "same change" needed to be applied. And rapid remapping between configurations exposed scale-dependent behaviors and system bugs that might not otherwise have been detected only by the testing of one configuration.

V. PARTITIONING

Partitioning of the BTeV detector consisted of running multiple independent data acquisition systems in parallel. This is not to be confused with the highway concept which was the physical implementation of parallel data streams. A partition is a logical concept and would have spanned multiple highways.

The value of partitioning, and the decision when one should partition the detector are different depending on the phase of the project. For example, the partitioning needs during commissioning (testing the subdetectors in parallel) were expected to be different than the needs for testing new L3 trigger algorithms while taking physics quality data. Partitioning allowed spare cycles on the online trigger farm to be used for offline processing when the beam would be off or luminosity low enough so as not to require all of the computational power in the trigger farm. The BTeV L2/3 farm contained significant processing power, and partitioning could provide a means to increase the utilization of this resource.

Partitioning was strictly a logical concept which needed to be mapped onto the physical implementation of the online system. The online trigger/DAQ was to be constructed in two stages. The first stage consisted of four highways, and the second stage consisted of the remaining four highways, but installed one year later. Even within a stage, individual highways were commissioned one at a time. The logical concept of partitioning needed to support running multiple partitions on a single highway (when only 1 was constructed) as well as the final system with eight highways

The parallel highway architecture and dynamic reloading of DCB routing tables allowed for much flexibility in configuring partitions, and rules were being established to limit the scope, function, and definition of partitioning. We had not reached a consensus on many details regarding

partitioning at the time of BTeV's termination. What we present here are some of the ideas and the directions in which we were headed.

Running a partition involved: 1) selecting and reserving a subset of electronics hardware to be read out, 2) defining how much L2/3 trigger processing power was needed and reserving those resources, 3) initializing the hardware, 4) collecting the data, and 5) freeing the resources.

One of our original ideas for a possible scenario was for a user to select specific front-end crates, request 50 Mflops of L2/3 nodes for processing, and then let the partitioning software map the request onto a physical implementation. Depending on how many nodes might be needed, the layout may require a single highway, or route data to n highways. We rejected this idea because it could have been confusing to the user to understand where data was flowing in the system. We ultimately developed a proposal that struck a balance between flexibility and ease of use. This approach imposed the following constraints: 1) the hardware for the L1 trigger on a particular highway could not be partitioned, but could hold trigger tables for multiple partitions; 2) the smallest source unit that could be reserved was a single L1 buffer which corresponded to as many as 24 DCBs; 3) a worker node could belong to one and only one partition; and 4) L2 worker nodes connected to the same regional manager could not span multiple partitions.

Because of the large number of electronics modules in the trigger/DAQ, we were developing a concept in which the L1 trigger and all active highways could be available as a shared resource. A run coordinator could then establish the overall online configuration for a period of time (days or weeks) and coordinate the data taking runs during this period. This person would have the understanding of which configurations would support multiple overlapping runs, and a stable configuration period would have a fixed and predefined set of allowable highways. Subdetector groups would be able to select specific electronics to read out, and front-end electronics could be reserved for read/write or read-only access. It was the responsibility of the run coordinator to schedule the detector so that users could get write access as needed, and partitions could be created or deleted as necessary.

For example, the run coordinator could make four highways available for the next two days. The pixel group could reserve the pixel front-end electronics and associated L1 buffers for read/write, and load the pixel trigger table. Bunch crossings would then be distributed to all four highways. The online software would assign specific L2/3 nodes to this partition as specified by the run coordinator. The silicon strip group could request and reserve silicon electronics for read/write and pixel electronics for read only access, and load a second set of trigger tables. Again, the software would assign L2/3 worker nodes specifically to this partition. If a given crossing passed the L1 trigger for both partitions, it could be routed to worker nodes in both partitions or split data between two partitions based on a

prescale value.

Partitioning became an obvious solution when discussing the problem of how to utilize spare online cycles for offline analysis. The computer scientists involved in RTES promoted real time scheduling on the worker nodes to maximize CPU utilization, but this was countered by the opinion that a particular worker node should only perform a single task to make operation of the system easier to understand and more manageable. Nodes could manually be moved between online and offline partitions, but would be automatically shifted to offline partitions as the luminosity in the Tevatron decreased during the course of a run. The automatic decision to migrate nodes between partitions would be influenced by metrics measured by RTES as well as an overall luminosity profile that would be loaded at the start of a run.

VI. CONCLUSION

When BTeV was first proposed, the online system was considered to be a high risk component of the experiment. Thanks to the efforts of many talented people in the collaboration and at the lab as well as extremely helpful comments from the many external project reviews, the DAQ and trigger groups in BTeV developed a low risk online architecture that was well understood, and was feasible in terms of cost and schedule.

At the time of termination in early 2005, the experiment had just passed its baseline review. Although not fully implemented, many of the architecture choices, design, and prototype work for the online system (both trigger and DAQ) were well on their way to completion. Other large, high-speed online systems may have interest in some of the BTeV design choices, including (a) a commodity-based tracking trigger running asynchronously at full rate with variable latency even at L1 (b) hierarchical control and fault tolerance in a large real time environment, (c) a partitioning model that supports offline processing on the online farms during idle periods with plans for dynamic load balancing, and (d) an independent parallel highway architecture.

REFERENCES

- [1] BTeV Collaboration, BTeV Proposal and TDR. Available: <http://www-btev.fnal.gov/public/hep/general/proposal/index.shtml>
- [2] J. Wu, "The Application of Tiny Triplet Finder (TTF) in BTeV Pixel Trigger", Proceedings of the the IEEE Real Time Conference 2005, Stockholm, Sweden, June 2005, to be published.
- [3] E.E. Gottschalk, "BTeV Detached Vertex Trigger", Nucl. Instrum. Meth. A473, 2001, p. 167.
- [4] Available: <http://www.dcache.org>
- [5] D. Messie, M. Jung, J. C. Oh, S. Shetty, S. Nordstrom, M. Haney, "Prototype of Fault Adaptive Embedded Software for Large-Scale Real-Time Systems", *Proceedings of the 12th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 4-7 April 2005, pp 498-505.
- [6] S. Ahuja, et al., "RTES Demo System 2004", *ACM SIGBED Review*, Special Issue on High Performance, Fault Adaptive, Large Scale Embedded Real-Time Systems, July 2005, Volume 2, Number 3.