

# Failure Analysis in a highly parallel processor for L1 Triggering

G. Cancelo\*, E. Gottschalk\*, V. Pavlicek\*, M. Wang\*, J. Wu\*

\* Fermi National Accelerator Laboratory

## Abstract

The current paper studies how processor failures affect the dataflow of the Level 1 Trigger in the BTeV experiment proposed to run at Fermilab's Tevatron. The failure analysis is crucial for a system with over 2500 processing nodes and a number of storage units and communication links of the same order of magnitude. The failure analysis is based on models of the L1 Trigger architecture and shows the dynamics of the architecture's dataflow. The failure analysis provides insight in how system variables are affected by single component failures and provides key information to the implementation of error recovery strategies. The analysis includes both short term failures from which the system can be recovered quickly and long term failures which imply a more drastic error recovery strategy. The modeling results are supported by behavioral simulations of the L1 Trigger running BTeV's Geant Montecarlo data.

## I. SUMMARY

The BTeV experiment, proposed for running at Fermilab, includes a sophisticated Trigger system in three levels [1]. The Level 1 Trigger uses Pixel and Muon Detector data. Both the Pixel and the Muon triggers work fairly independent until the last stage where their outputs are combined by the Global Level 1 Trigger, as shown in Figure 1. The Level 1 Trigger is a complex highly parallel event processor of the order of 2500 processing nodes. The Trigger's downtime is required to be less than 5%. Hence, the system must be robust, fault tolerant and run even in the event of component failures. Furthermore, the Trigger's performance must gracefully decrease in the occurrence of an increasing number of single component failures. The BTeV Trigger's fault tolerance problem has originated an independent project in real time embedded systems (RTES) [2] to develop the appropriate hardware and software framework.

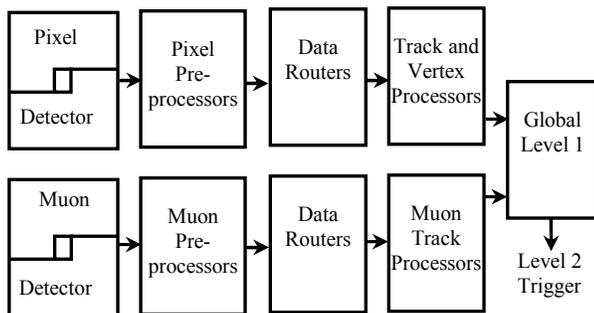


Figure 1

Figure 1 shows the L1 Trigger building blocks. The data preprocessors have two main functions, the Pixel Preprocessor and the Segment Tracker. The Pixel Preprocessors format and sort the raw data coming from the Pixel Detector. The Segment Trackers generate triplets of points that describe the beginning and the end of all tracks in each event. Each Pixel Preprocessor and Segment Tracker processes a small geographic portion of the pixel detector. The data generated every bunch crossing (BCO) of the accelerator is stamped with a distinctive temporal label called Time Stamp (TS). The Data Router or Switch routes all data that share a same Time Stamp to the same Track and Vertex processor. Each data event is assigned to a single processing node because trigger decisions are made on event by event basis. The Track and Vertex processors are grouped in larger units of hardware called Farmlets (Figure 2). Processors in a Farmlet share some resources such as data I/O path, main buffering and network connections and some interfaces.

The Trigger's architecture has been designed based on parallel computing models. A dataflow analysis through the models allowed us to optimize design parameters such as number of processors per Farmlet, processor workload, buffering, and latency. It has, also, been crucial to eliminate bottlenecks and compute link bandwidths.

As said, a main concern in the trigger design is component and system failures. It is obvious that dataflow and failure analysis are intimately linked since failures will tend to break the system balance. The current document analyses the Level 1 Trigger Farmlet's dataflow in the event of failures. The analysis of the Pixel Processor and Segment Tracker has been described in paper N29-7 at this conference.

## II. PROBLEM STATEMENT

The Farmlet's dataflow analysis was carried out resorting to statistical queuing models. All such models and results are supported by dataflow simulations.

A steady state analysis of the Farmlet's model reveals all the parameters mentioned above when the system works in its steady state. However, hardware or software failures unbalance the system's load. This problem has been analyzed looking at the dynamics of the Farmlet's queuing model. To avoid bottlenecks or data loss it is important to understand the time constants that dominate the transient system dynamics upon a failure.

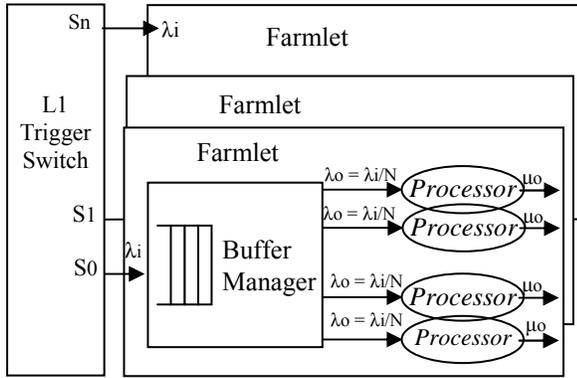


Figure 2

The input data flow to the Trigger processor is not time uniform. Hence the Trigger must allow for some idle capacity to be able to cope with the dataflow statistical fluctuations. It is crucial to keep the idle capacity as low as possible during normal operation. However, failures produce a relative increase in the net dataflow in a localized section of the architecture. If the increase is substantial or cannot be mitigated soon enough, it will result in system's instability and data loss. This paper provides a stability analysis and reports on how different variables affect the dataflow. The dataflow dynamics allow us to classify failures as transitory (i.e. short term, recoverable) or permanent (i.e. not recoverable before the system reaches a new dataflow steady state). One of the important results of this analysis is the indication that the idle capacity needed to cope with permanent sporadic processor failures is too expensive. To address this problem the transitory behavior of the system upon failures has been analyzed. When a failure occurs, the data storage queues near the failure start increasing size, the processors near the failure increase their workload and some internal communication channels, also, see a steady increase in their bandwidth requirements. However, the dynamics of these processes may allow a fault recovery system. A key result of this analysis is the estimation of the time allowed for failure recovering without creating a significant effect on the system's steady state. It is also of importance to understand which design variables can be used to control this time.

The Trigger system models were implemented using a behavioral simulator. The simulations confirmed the model's outputs. The importance of the simulation exercise resides in the ability of using input data that comes from Geant simulations of the BTeV detector. The Geant data is an accurate prediction of the real data that the Trigger will see at run time. The Geant data was used to test the Trigger and failure models at different luminosity levels of Fermilab's Tevatron.

### III. FARMLET MODELS

As shown in Figure 2 the main constituents in a Farmlet are the Buffer Manager (BM) and the processing nodes. The

incoming data are stored into a queue from which the BM retrieves and assigns events to the processing nodes. The Track and Vertex algorithm running on a Pentium III-M at 1.13GHz shows that the event service time is exponentially distributed with a mean equal to 90.91μs. The event interarrival time at the Farmlet's input is a design variable constrained by the number of Farmlets connected at the output of the Switch (Figure 2). This number can be controlled to obtain the desired processor utilization, mean service time or input buffer size. In this context the event arrival-delivery at the input queue can be modeled as an M/M/1 queue. The event data (i.e. triplets) have a modulation effect on input queue size. The analysis of this problem can be found at [4].

Defining the input queue interarrival time as  $\lambda = \lambda_i$  and the queue service time as  $\mu = \sum(\mu_o)$  the Buffer Manager's input queue can be modeled as a M/M/1 queue (Figure 3).

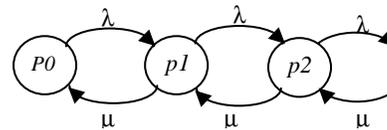


Figure 3

The steady state distribution of the number of events in the queue is given by

$$p_k = \rho^k p_o \quad \text{where} \quad p_o = 1 - \rho,$$

and  $\rho = \lambda/\mu$  represents the utilization of the processing nodes. The processor's idle time is equal to  $p_o = 1 - \rho$ . As it is well known, the average queue size and average queueing time of events approach infinite as the utilization goes to 1.

$$Avg\_queue\_size = \frac{\rho}{1 - \rho}$$

$$Avg\_queueing\_time = \frac{1/\mu}{1 - \rho}$$

For instance, a utilization of 90% of the server implies an average queueing time equal to 10 times the average processing time. This adds a latency of 0.9ms to the event pipeline. The average queue size for 90% utilization is 9 events, which represent about 13.5 KB for a typical average of 1.5KB/event.

The variance in the queue size is given by

$$Variance\_of\_queue\_size = \frac{\rho}{(1 - \rho)^2}$$

It means that if we want to operate the server at 90% utilization and be able to store 99.99% of the events ( $\mu + 4\sigma$ ) we need to be able to store at least 47 events in the input queue of the buffer manager (Figure 2)(i.e. 70KB,  $\sigma = 9.487$ ).

### IV. PROCESSOR FAILURE ANALYSIS

If a node fails, the Buffer Manager in the Farmlet can reroute the buffered events to the operating nodes until the failing node comes back to operation or the whole Farmlet is replaced. It is obvious that to keep the Farmlet in a stable state the utilization factor  $\rho$  after the failure must be smaller than 1. In other words,

$$\rho_{AF} = \frac{N}{N-1} \rho_{BF} \Rightarrow \rho_{BF} < \frac{N-1}{N} \quad (1)$$

where  $N$  is the number of processing nodes in the Farmlet. Equation (1) tells us that if we want to have a high utilization of the processing Farmlet before a failure and be able to keep the Farmlet stable after a node's failure, we must increase the number of processing nodes to distribute the workload of the failing node among more processors. Increasing the number of processing nodes per Farmlet lowers the cost per node of shared resources in the Farmlet such as interface links, network, Buffer Manager, and I/Os; but there are some hard limitations to it. The data I/O and Buffer Manager's bandwidths increase linearly with the number of processors. Implementing larger Buffer Managers is more complicated and may increase the cost. Large PC board sizes are usually not recommended for many reasons. The Farm's reliability decreases with the number of processing nodes.

There are modules in the Farmlet whose failure will cause the entire Farmlet to fail. If the failure is permanent (e.g. a hardware failure) the Farmlet must be switched off and replaced. On the other hand, the failure of a single node may be tolerated at least for a short period of time. A failure can be considered transitory (or recoverable) as opposed to permanent (or non-recoverable) when the processing node can be restarted in a short time compared to the system's dynamics. The following section analyzes the transient behavior of a Farmlet after a node's failure (e.g. software runtime failure) and the possibility of maintaining stability when the node's failure is recoverable.

## V. TRANSIENT ANALYSIS

The M/M/1 transient analysis is more complicated than the one made for the equilibrium point. In the equilibrium analysis we get rid of the time variable, in the transient analysis we must work with the full differential-difference equations of the M/M/1 model (Figure 3) given by [4]

$$\frac{dP_k(t)}{dt} = -(\lambda + \mu)P_k(t) + \lambda P_{k-1}(t) + \mu P_{k+1}(t) \quad k > 1 \quad (2a)$$

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t) \quad k = 0 \quad (2b)$$

To solve this set of equations analytically, the easiest way is to resort to transform methods. Since we have a continuous variable (i.e. time) and a discrete variable (i.e. queue state probability  $k$ ) we need to use the Laplace transform and the Z-transform respectively. Applying both to equation 2a and using 2b to reduce the number of unknowns we obtain

$$P^*(z, s) = \frac{z^{i+1} - \mu(1-z)P_o^*(s)}{sz - (1-z)(\mu - \lambda z)} \quad (3)$$

where  $P_o^*(s)$  is the Laplace transform of the distribution of the idle state  $p_o$ .  $P_o^*(s)$  can be determined using the property of analyticity of the transformations. The solution to equation (3) is

$$P_k(t) = e^{-(\lambda+\mu)t} \left( \rho^{(k-i)/2} I_{k-i}(at) + \rho^{(k-i-1)/2} I_{k+i+1}(at) + (1-\rho)\rho^k \sum_{j=k+i+2}^{\infty} \rho^{-j/2} I_j(at) \right) \quad (4)$$

$$\text{where } \rho = \frac{\lambda}{\mu}, \quad a = 2\mu\rho^{1/2}$$

and  $I_k(x) = \sum_{m=0}^{\infty} \frac{(x/2)^{k+2m}}{(k+m)!m!}$   $k \geq -1$  is the modified Bessel function of the 1<sup>st</sup> kind (4).

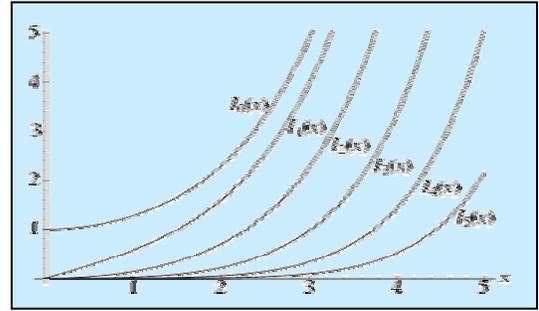


Figure 3

Equation (4) is ill suit for calculation using numerical methods. It not only multiplies increasing and decreasing exponentials but an infinite sum of them. Instead, for the present analysis we have chosen to follow the Orthogonal Least-Squared Approximation method suggested by Bolot [3].

## VI. OLS APPROXIMATION TO THE TRANSIENT ANALYSIS OF THE M/M/1 QUEUE

Equation (4) represents the instantaneous state probability distribution of the M/M/1 system. That is a whole distribution for every instant of time. However, we are more interested in how the average queue size evolves with time rather than its instantaneous value. We define the transient mean queue size as

$$Q(t) = \sum_{j=0}^{\infty} j P_{o_j}(t)$$

It is obvious that the mean queue size for  $t \rightarrow \infty$  must be  $Q(\infty) = \rho/(1-\rho)$  as defined in Section III.

It can be shown that  $Q(t)$  is monotonically increasing with exponential behavior [5]. The Orthogonal Least-Squared Approximation (OLS) uses the following model

$$q_n(t) = a_o + \sum_{i=1}^n a_i e^{-b_i t} \quad b_i > 0$$

to approximate  $Q(t)$ . The measure of approximation is the  $L_2$  norm

$$L_2(Q(t) - q_n(t)) = \sqrt{\int_0^{\infty} |Q(t) - q_n(t)|^2 dt}$$

Of course, we can eliminate the square root minimizing the square of the  $L_2$  norm with respect to the  $a_i$  and  $b_i$  coefficients. In any case, this is not an easy task, which becomes harder as we raise the order of our approximation model.

A first order model is quite simple and can be expressed in closed form. Let the model be

$$q_n(t) = \bar{q}(1 - e^{-b_1 t}) \quad \text{where} \quad \bar{q} = \frac{\rho}{1 - \rho}$$

then, minimizing

$$L_2^2(Q(t) - q_1(t)) = \int_0^{\infty} |Q(t) - \bar{q}(1 - e^{-b_1 t})|^2 dt$$

we get  $b_1 \approx \mu(1 - \rho)^2(1 + 0.2539\rho)$ .

$b_1$  is the reciprocal of the *time-constant* of the exponential function. The time constant  $\tau$  is a better demonstrator of the system's dynamics

$$\tau = \frac{1}{b_1} = \frac{1}{\mu(1 - \rho)^2(1 + 0.2539\rho)} \quad (5)$$

The first order approximation works quite well for small  $\rho$  but it is not so accurate for  $\rho$  closer to 1. Figure 5 shows the real  $Q(t)$  and the 1<sup>st</sup> and 2<sup>nd</sup> order approximations for  $\rho=0.9$ .

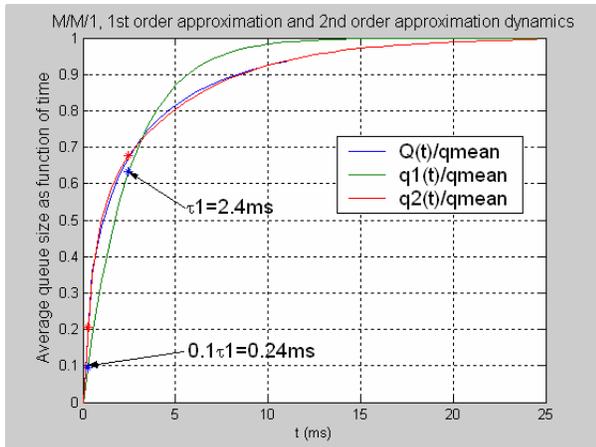


Figure 5

The second order approximation uses the model

$$q_2(t) = \bar{q} + a_1 e^{-b_1 t} - (\bar{q} + a_1) e^{-b_2 t} \quad \text{where} \quad b_1 > 0, b_2 > 0$$

The minimization procedure gets more cumbersome and the optimal coefficients must be found by numerical methods.

For instance for  $\rho=0.9$ ,  $a_1=-4.401$ ,  $b_1=0.0572\mu$ , and  $b_2=0.0058\mu$ . Figure 5 shows that  $q_2(t)$  approaches  $Q(t)$  very closely even for high  $\rho$ . The approximation errors can, also, be calculated numerically.

The beauty of the OLS analysis is its simplicity, and the fact that allow us to characterize the M/M/1 dynamics with very simple parameters. It is customary to characterize a signal's exponential behavior by its time constant (also called relaxation parameter). The 1<sup>st</sup> order approximation model's time constant is expressed by Equation (5) as a function of  $\rho$  and  $\mu$ . Figure 6 shows the Time Constant  $\tau$  as a function of  $\mu$  (i.e. the processor's average service time), parameterized by  $\rho$ .

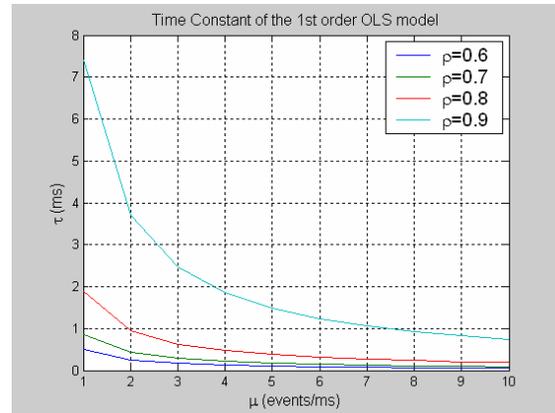


Figure 6a

The 2<sup>nd</sup> order model has two time constants one for each exponential

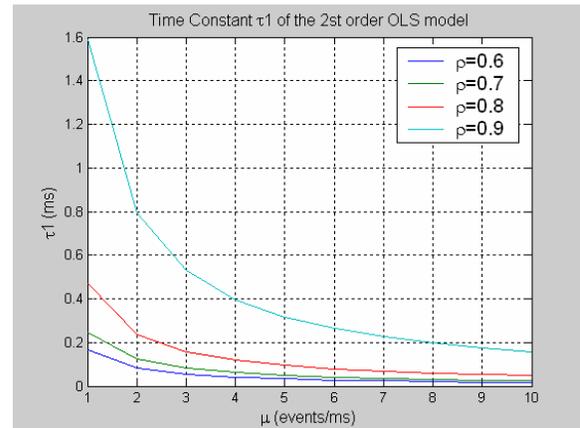


Figure 6b

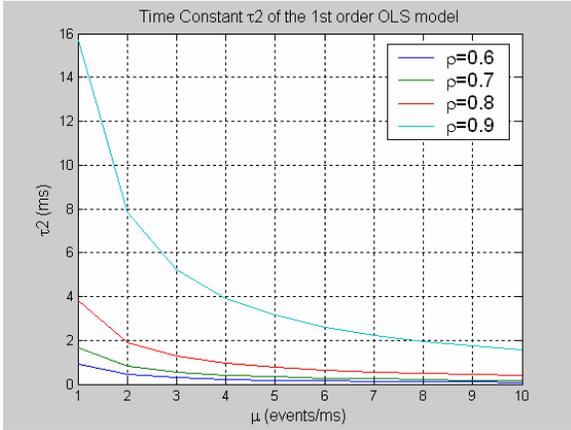


Figure 6c

As both plots show, the model’s dynamics speeds up with  $\mu$ . That is, faster processors or a larger number of them per Farmlet will lower the time constants of the exponentials and the system approaches faster to its steady state. Secondly, it can be observed that higher  $\rho$  “slow down” the system’s dynamics. Of course, the price to pay for this slower dynamics is a larger average queue size but this should not be an issue in our case because the average event size is small ( $\sim 1.5$  Kbytes). On the other hand,  $\rho$ , which is a design parameter, will be chosen based on the allowed Farmlet’s “idle time”, which is the expensive commodity, and not based on memory buffer size. Indeed, the memory buffer size is a dependent variable.

The 1<sup>st</sup> and 2<sup>nd</sup> order approximation time constants allow us to calculate how much time we allow to restore the failed processor in the Farmlet as a function of how much excess buffering and processing latency we can afford. Furthermore, as we increase  $\rho$ , the system can enter quickly into the unstable mode and all new events will be queued up, forcing the Buffer Manager to start throttling out data.

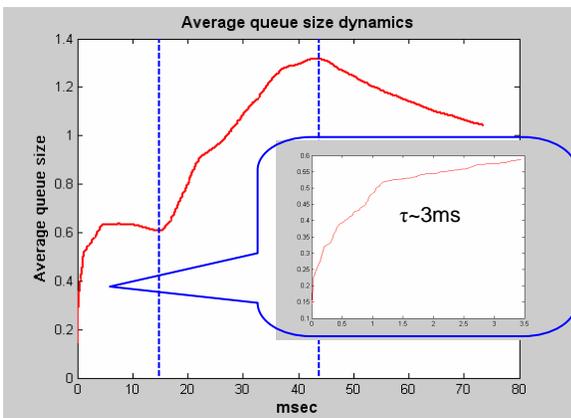


Figure 7

A behavioral simulation of the Farmlet’s transient response with about 50 thousand events shows similar results (Figure 7). The simulator models a Farmlet with four processors. At startup the input buffer is empty and the time constant measured is close to  $3\mu s$ . One of the processors simulates a failure at 15ms and is restored at 43 ms. The time constant of the change of state is longer because the accumulated data smooth out the transition.

### Data Purging

The L1 Trigger is a data-push architecture. There is not feedback mechanism between stages that can stop the dataflow. Hence, each stage must implement a way to cope with instantaneously high flow rates and eventually with overflows.

The strategy implemented in the L1 Trigger avoids all buffer overflows using a technique that purges data when the buffers are close to overflow. This is a controlled way of handling data that cannot be processed. Those events can be reported to other trigger stages, including the TS number of the data that could not be processed. In this way, unprocessed data in L1 Trigger is not thrown away. Instead L1 Trigger reports to L2/3 that events with certain TS could not be processed and should be processed by L2/3.

### VII. CONCLUSIONS

The analysis of the Farmlet’s dataflow models the dynamic behavior of the data as it goes through stages of processing. The study of the queue dynamics in an event of failure is critical in the Farmlet design. The transient analysis of the Farmlet behavior estimates the average time allowed to intent a processor recovery upon failure. This time is a function of the average queue size increase and average workload increase in the functioning processors. The queue dynamics’ time constant allows us to design recovery techniques of temporary processor failures.

### REFERENCES

- [1] E. Gottschalk, “BTeV detached vertex trigger”, Nucl. Instrum. Meth. A 473 (2001) 167.
- [2] BTeV-RTES group, “RTES-ITR proposal”, BTeV-doc-1002-v1, July 29 2002.
- [3] Bolot, J.-C., Shankar, A., “Optimal least-squares approximations to the transient behavior of the stable M/M/1 queue”, Communications, IEEE Transactions on , Volume: 43 Issue: 234, Feb./March/April 1995 Page(s): 1293 -1298
- [4] G. Cancelo, “Level 1 Pixel Trigger Data Flow Analysis”, BTeV-doc-1177-v1, Sept. 2002.
- [5] G. Cancelo, “Dataflow analysis in the L1 Pixel Trigger Processor Farm”, BTeV-doc-1178-v1, Sept. 2002.