



**Dataflow analysis in the L1 Pixel Trigger Processor  
Farm**

**BTeV-doc-1178, version 1**

**19 September 2002**

Gustavo Cancelo for the BTeV Trigger Group

## Table of Contents

1	Dataflow analysis in the L1 Pixel Trigger Processor Farm	3
2	Input data characterization	4
2.1	The event execution time distribution	4
2.2	The Farmlet input distribution	5
3	Elementary Dataflow Analysis	5
4	Processor Failure Analysis	6
4.1	Node Failures	7
4.2	Transient Analysis	7
4.3	Orthogonal Least-Squared Approximations to the Transient Analysis of the M/M/1 queue	8
4.4	Simulations of the M/M/1 transient behavior	11

# 1 Dataflow analysis in the L1 Pixel Trigger Processor Farm

The architecture of the L1 Track and Vertex (T&V) is simplified by the fact that each event is processed independently. There is neither time-correlation nor need of data sharing among the computing nodes. Each processor works on the events it has been assigned to, without need of communicating or exchanging information with other processors. There is not need for shared pools of memory either. In consequence, the dataflow analysis is simplified. However, a critical issue in such a massive parallel processing system is reliability. It is important to study the dynamics of the dataflow when a computing node fails (i.e. a processor or associated electronics). Another important issue is determining the optimum number of nodes in the basic package that we have called the Farmlet. Even when the nodes work fairly independently, the nodes on the same Farmlet can share resources and services. It is easy to demonstrate that incrementing the number of processors per Farmlet is advantageous in terms of dataflow. Of course, this faces obvious technological problem of board size and density. The number of processing nodes is likely to be between 4 and 8.

The L1 Trigger architecture shows that events of “Triplet Data” are built by the L1 Trigger Switch. Every Switch output is connected to a Farmlet input as shown in Figure 1. A centralized event buffer and scheduler in the Farmlet (i.e the Buffer Manager), distributes events to the processing nodes.

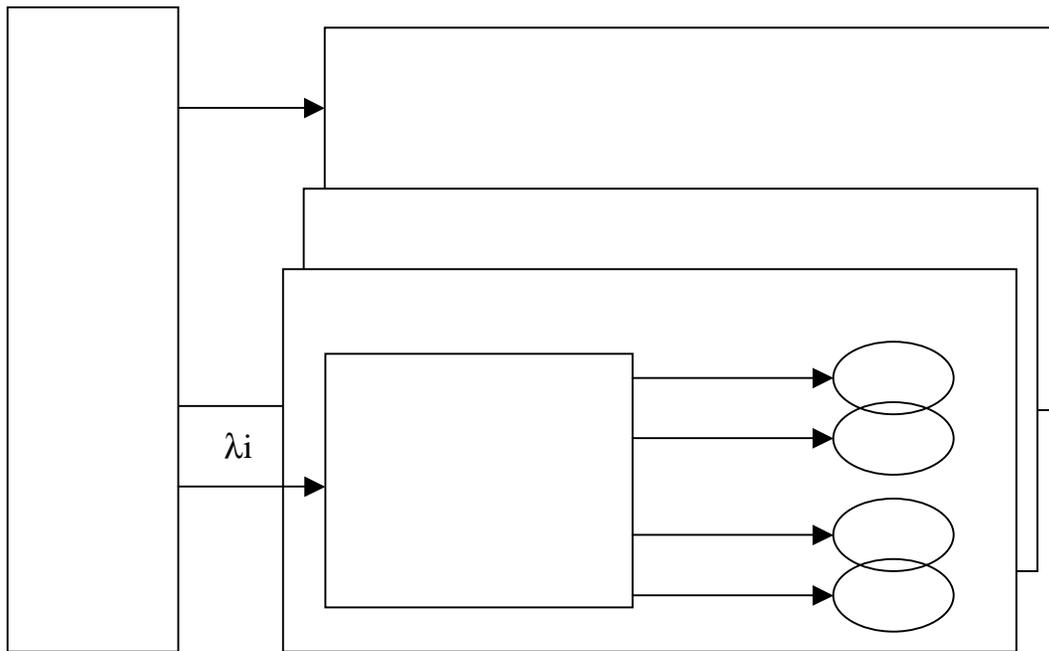


Figure 1

The following analysis uses as example a Farmlet with 4 Processing nodes. The data bandwidths in the Farmlet were calculated in [1]. The event processing times used in the data flow analysis simulations were taken from simulations of the L1 Track and Vertex algorithm using available hardware (e.g. Intel P3-M 1.13GHz).

The following numbers are used in the calculations:

Pixel Detector Output Bandwidth: 47.3 MB/s

Number of L1 Trigger Highways: 8

Number of Processing Nodes per Farmlet: 4

A crude cost analysis shows that the cost of the L1 Trigger is driven by the cost of the processors and associated circuitry more than the cost of the memory needed or the cost of the data links.

## 2 Input data characterization

The input data has segments of trajectories that represent “triplets” of pixel hits. In fact each “triplet” includes 6 points, corresponding to 3 bend-view and 3 non bend-view hits of consecutive Pixel Detector stations. However, in the current analysis we follow the convention of representing a “triplet” by 2 entries; one containing the 3 bend-view hits and a second one with the 3 non bend-view hits. Hence, the number of “triplets” doubles. The Triplet Data distribution is as shown in Figure 2. The average number of Triplets is 88.9 three-hit triplets. If a triplet can be represented by 16 bytes, and average event is about 1.5KB.

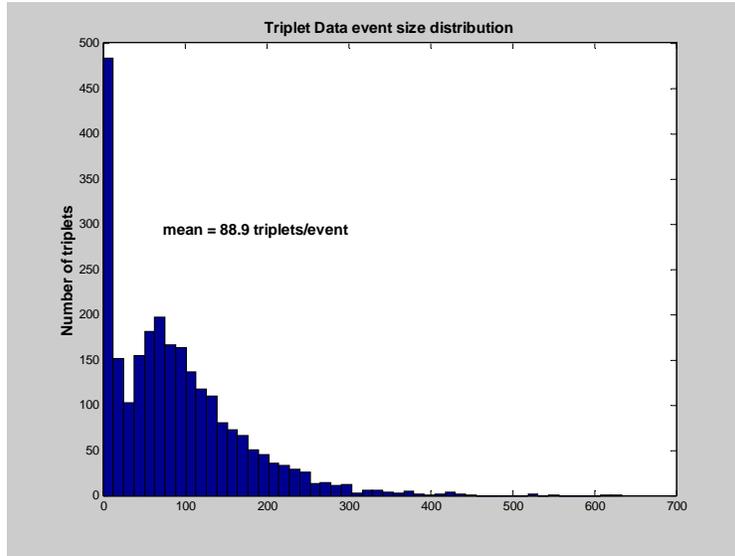


Figure 2

The 2500 event statistics shows that about 12.7% of the events are empty of triplets. If we only average over non-empty events, the mean event size is 102 triplets/event.

### 2.1 The event execution time distribution

Figure 3 shows the event execution time distribution on an Intel P3-M 1.13GHz. The average execution time is 90.91  $\mu$ s. The distribution is exponential with parameter  $\mu=1/\text{average execution time}$ , but it also has few events in a long tail.

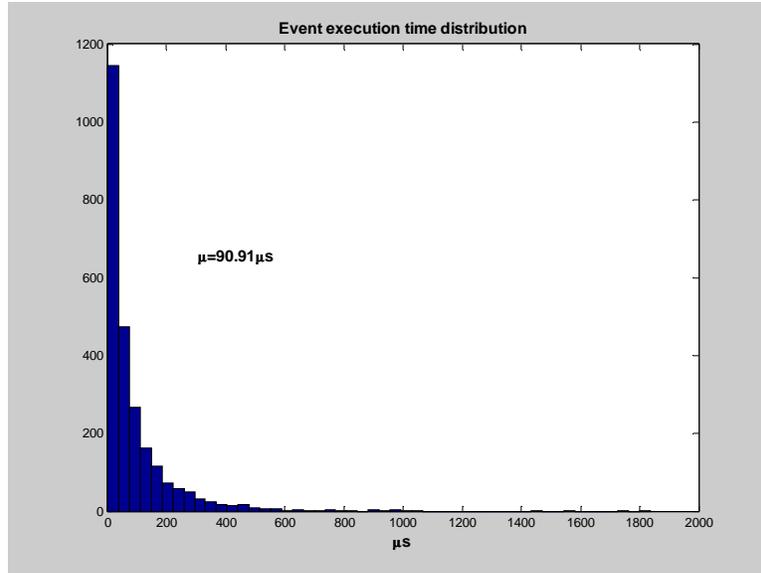


Figure 3

## 2.2 The Farmlet input distribution

It has been shown in [1] that the Triplet Data distribution is exponential. The occurrence of new events is exponentially distributed. The event size distribution convolutes with the arrival distribution. The mean event-interarrival time of the input distribution can be used as a design variable. In fact, the mean event-interarrival time depends on the number of Farmlets or L1 Switch outputs enabled. The calculation of the number of Farmlets based on dataflow issues is one of the main results of the current analysis.

## 3 Elementary Dataflow Analysis

The simplest model for the Farmlet is the M/M/1 queue. We can model the Farmlet's Buffer Manager (BM) (Figure 4) as such queue. The input to the BM is the Triplet Data and it is stored in an input buffer (FIFO). The service distribution is composed by the sum of the individual service distribution of the processors in the Farmlet. Since the sum of few Poisson streams is also Poisson, the queue can be modeled as M/M/1. In fact, this reasoning works both ways. We can either think that the BM is an M/M/1 process whose mean interarrival time is  $\lambda$  and mean service time is  $N\mu$  or we can define an M/M/1 process for the individual computing nodes defining the mean interarrival time as  $\lambda/N$  and the mean service time as  $\mu$ .  $N$  is the number of computing nodes in the Farmlet. See Figure 5.

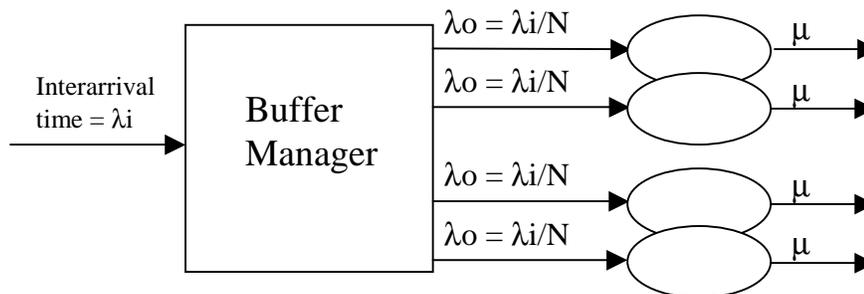


Figure 4

The steady state distribution of the M/M/1 process is given by:

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

$\rho = \lambda/\mu$  represents the utilization of the processing node. For this simple process the Idle Time is equal to  $p_o$  or  $1-\rho$ . The problem is that both the average queue size and the latency in processing an event (i.e. the queuing time) approach infinite as  $\rho$  approaches 1.

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

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

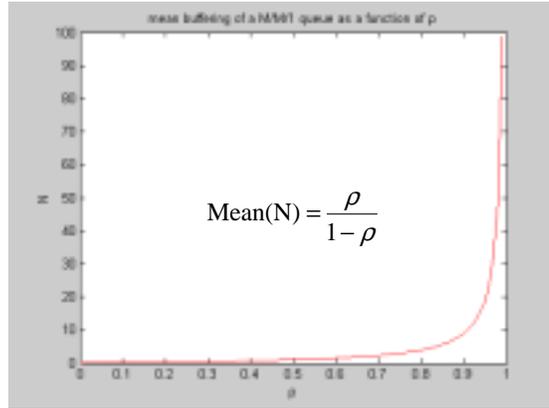


Figure 5

For instance, a utilization of 90% of the server implies an average queuing time equal to 10 times the average processing time. That is, 0.9 ms for the average processing time shown above. The average queue size for 90% utilization is 9 events, which is about 13.5 KB for an 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 (i.e. 70KB,  $\sigma=9.487$ ).

## 4 Processor Failure Analysis

This is very important topic to analyze because in a 2500 processing node system reliability is a big issue. The current architecture is based on Farmlets with a fix number of nodes per Farmlet. If a node fails, the Buffer Manager in that Farmlet can reroute the buffered events to the nodes still working 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. Here, we define stability as the ability of the system to keep operating at full capacity, keeping the event queue away from overflowing and event execution times finite 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 Farm before a failure and be able to keep the Farm 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 Farm such as PTSM 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 larges 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.

## 4.1 Node Failures

There are several modules in the Farm whose failure will cause the entire Farm to fail. If the failure is permanent (e.g. a hardware failure) the Farm must be switched off and replaced. On the other hand, the failure of a single node in a Farm may be tolerated at least for a short period of time. A failure can be considered transitory (or recoverable) as opposed to permanent (or a long time failure) if the failure time is short or the processing node can be restarted in a short time compared to the system's dynamics. Section 4 analyzed Farm stability after permanent failures. However, many processing node failures can be recoverable (e.g. software runtime failures). The following section analyzes the transient behavior of a Farm after a node's failure and the possibility of maintaining stability when the node's failure is recoverable.

## 4.2 Transient Analysis

The M/M/1 transient analysis is far 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 given by

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

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

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

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

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

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 (Figure 6).

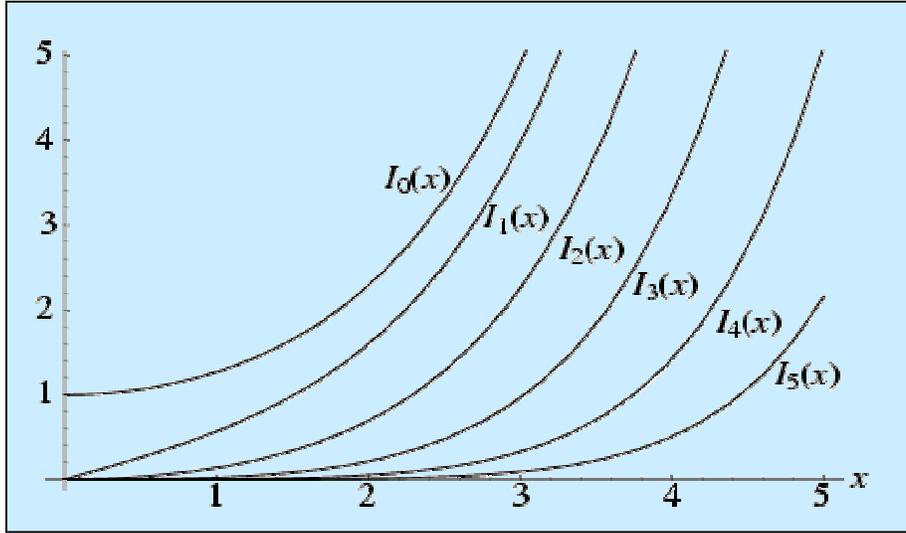


Figure 6

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 I have chosen to follow the Orthogonal Least-Squared Approximation method suggested by Bolot [REF].

### 4.3 Orthogonal Least-Squared Approximations 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 the instantaneous value. We define the transient mean queue size as

$$Q(t) = \sum_{j=0}^{\infty} j P_{0_j}(t). \text{ It is obvious that the mean queue size for } t \rightarrow \infty \text{ must be } Q(\infty) = \rho / (1 - \rho) \text{ as defined}$$

in Section 3.

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

$$q_n(t) = a_0 + \sum_{i=1}^n a_i e^{-b_i t} \quad b_i > 0$$

to approximate  $Q(t)$ . The measure of approximation is the L2 norm

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

Of course, we can get rid of the square root and minimize the square of the L2 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, which is a better demonstrator and will

be used later on. Let  $\tau = \frac{1}{b_1} = \frac{1}{\mu(1 - \rho)^2(1 + 0.2539\rho)}$  (5)

The first order approximation works quite well for small  $\rho$  but it is not so accurate for  $\rho$  closer to 1. Figure 7 shows the real  $Q(t)$  and the 1<sup>st</sup> and 2<sup>nd</sup> order approximations for  $\rho=0.9$ .

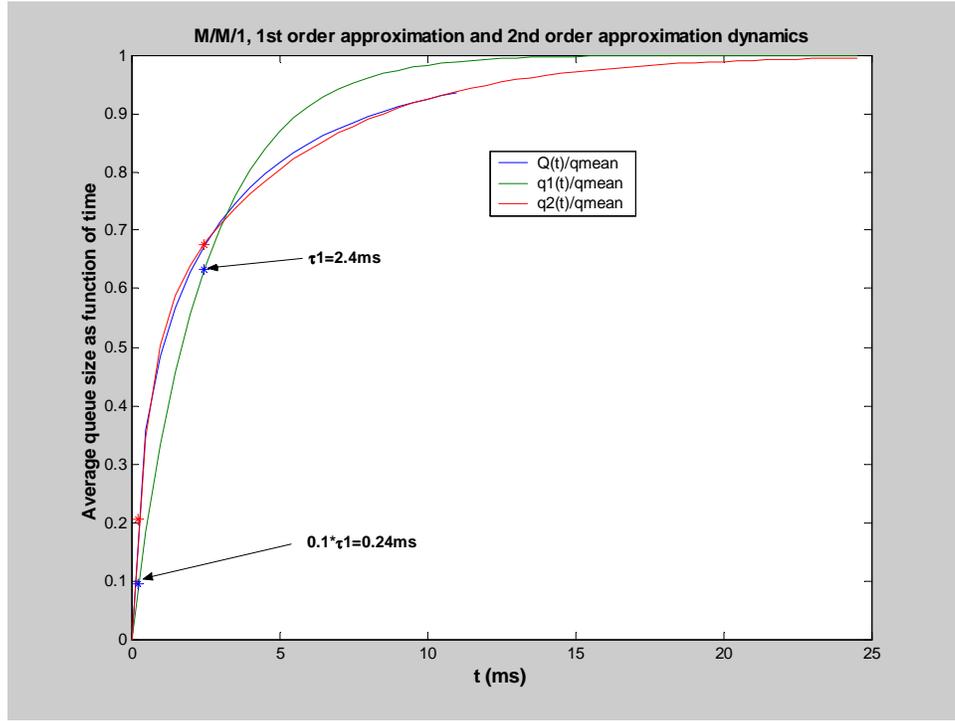


Figure 7

The second order approximation uses the model

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

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 7 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 we can 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 8 shows the Time Constant  $\tau$  as a function of  $\mu$  (i.e. the processor's average service time), parameterized by  $\rho$ .

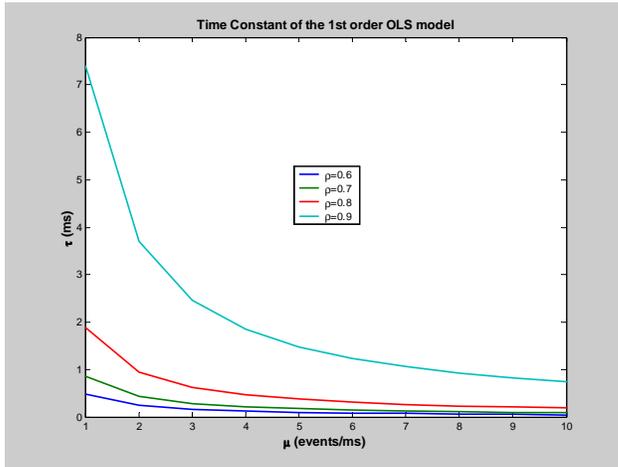


Figure 8a

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

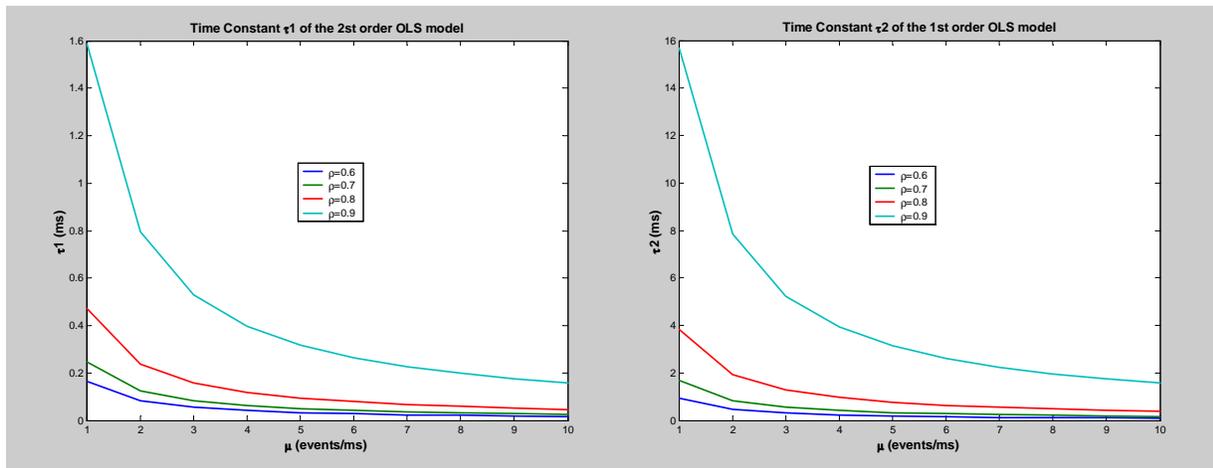


Figure 9b,c

As both plots show, the model's dynamics speeds up with  $\mu$ . That is, faster processors or a larger number of them per Farmllet 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 we are only talking about Kbytes. On the other hand,  $\rho$ , which is a design parameter, will be chosen based on the allowed Farmllet'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 Farmllet 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.

## 4.4 Simulations of the M/M/1 transient behavior

The simulation of the transient M/M/1 shows the behavior of the buffer queue and the processing nodes. The input data to the transient M/M/1 simulation comes from Geant simulations of the BTeV's detector. The raw pixel data has been run through a Pixel Processor and Segment Tracker simulator, which finds the Triplet Data as defined in Section 2. Please, see reference [1] for details. The simulation of the transient M/M/1 does not include the entire Track and Vertex finding code. Instead, the processing time values corresponding to each event were taken from other tests where the Track and Vertex code was actually run on hardware [1].

The first simulation shows the turn-on of the Farmlet. The Farmlet has 4 processing nodes. The event's execution time pdf is as shown in Figure 3, which has a mean of  $90.91\mu s$ . As said in Section 2, the input data rate is a design parameter. The L1 Trigger can control the mean event-interarrival time to a Farmlet by controlling the number of Farmlets hooked to the L1 Switch outputs. In the current simulations we set this value to a value that provides the desired  $\rho$ . Figure XXX shows the evolution of the Buffer Manager's queue for  $\rho=0.688$ .

(to be complete)

### References

[1] [Gustavo I. E. Cancelo, Dataflow analysis in the L1 Pixel Trigger Processor Farm, BTeV-doc-1178-v1](#)

