EPJ Web of Conferences **295**, 10002 (2024) https://doi.org/10.1051/epjconf/202429510002
CHEP 2023

Control Simulation for an ESnet-JLab FPGA Accelerated Transport Load Balancer

Derek Howard[2,*], *Michael* Goodrich[1,**], *Carl* Timmer[1,***], *Yatish* Kumar[2,****], *Graham* Heyes[1,†], *David* Lawrence[1,‡], *Stacey* Sheldon[2,§], and *Vardan* Gyurjyan[1,¶]

[1]Thomas Jefferson National Accelerator Facility
[2]Energy Sciences Network

Abstract. The Thomas Jefferson National Accelerator Facility collaborates with Lawrence Berkeley National Lab to implement a dynamic UDP load balancer (LB) for high-throughput scientific data processing. This study employs a simulation to compare the efficacy of Proportional, Integrative, Derivative (PID) controllers and Q-Learning based controllers for configuring the load balancer. Two cluster configurations, homogeneous and heterogeneous, were examined. The simulation results indicate that PID control is superior in both configurations. In homogeneous clusters, PID achieved a 50% reduction in aggregate queue levels and maintained an even distribution across computational nodes (CNs). In contrast, Q-Learning was less effective in heterogeneous environments, exacerbating queue levels compared to the no-control case and failing to achieve balance across the cluster. Our findings suggest that PID control should be used for the ESnet-JLab FPGA Accelerated Transport (EJFAT) system.

1 Introduction

In operation since 1995, the Thomas Jefferson National Accelerator Facility's (JLab) primary mission is to do nuclear physics research. To handle increasing data rates and changing detector needs, JLab has embraced distributed computing. Current instruments are capable of producing data at rates far too great for local compute infrastructure to process or store. JLab has partnered with ESnet to build a UDP load balancer (LB) that allows near-realtime processing of high-throughput scientific data on shared compute infrastructures such as NERSC's Perlmutter supercomputer. Since cluster schedulers allocate nodes dynamically, the LB must also dynamically adjust where this scientific data is sent. This requires a control mechanism so data rates can be adapted to available resources. Here, we use a simulation to investigate which control mechanisms for this load balancer are most effective at maintaining required throughput and latency.

*e-mail: dhoward@es.net
**e-mail: goodrich@jlab.org
***e-mail: timmer@jlab.org
****e-mail: yak@es.net
†e-mail: heyes@jlab.org
‡e-mail: davidl@jlab.org
§e-mail: stac@es.net
¶e-mail: gurjyan@jlab.org

2 EJFAT System Architecture

The EJFAT architecture consists of a data producer, an FPGA-accelerated dynamic load balancer, and a dynamic cluster of data consumers.

The EJFAT LB defines a custom UDP-based protocol for dynamically directing *events*, where we define an *event* as data that is to be processed as a meaningful whole by a single CN, in toto to selectable CNs and logical streams within these events to individual ports on the selected CN.

Data producers send a stream of events by fragmenting them into multiple UDP packets each with metadata headers that mark each packet with event and logical stream tags, where the network destination is the LB Data Plane (DP) address. In addition, a *Reassembly Engine (RE)* header follows the LB metadata and provides for event reassembly downstream. The RE metadata is not defined by EJFAT and is a use-case-dependent agreement between sender and receiver.

The LB DP redirects each tagged packet to dynamically selectable CNs in real-time with micro-second level latency by rewriting UDP packet headers.

Data consumers register with the LB Control Plane (CP) as to their dynamic capacity to process events then receive, reassemble, and process these events.

2.1 Data Plane

Arriving packets are processed using the metadata and CN mappings defined by the control plane (CP). Packets are validated as in the correct format, then the event number is used as a look-up key ascertain the mapping *epoch* and destination CN.

The Epoch advancement mechanism is how the LB can map events to CNs while ensuring that all packets for a given event are delivered to the same host (Figure 1). An Epoch is defined as a sequential block of event numbers for which the nodes that events will be sent to are fixed. CNs are selected by taking the event number modulo a number of *slots*, which are a cyclical mapping of event numbers to CNs. The number of slots assigned to each node determines the proportion of total events it will receive. The CP defines independent mappings for each epoch, which become effective when the event numbers sent by data producers advance into its range. In our implementation 3 epochs are defined at one time - current, previous and future. The previous epoch is kept in order to direct late-arriving packets from the previous epoch to its correct destination. The future epoch represented allows adjustments to be made to the number of slots occupied by a node without disruption to the existing configuration.

After mapping events to their epoch and slot, the LB rewrites the UDP packet header so the destination is the mapped CN's address and port, removes the metadata from the packet, and egresses the packet to the correct CN for that epoch.

Data consumers receive and enqueue UDP packets from the LB which must be reassembled into events by use of the accompanying metadata. This can be done in parallel for a single event by using one thread per data source within an event. Reassembled events are subsequently enqueued for downstream processing.

For more technical details on the EJFAT system, refer to previous papers [1],[2].

2.2 Control Plane

The EJFAT control plane is responsible for dynamically configuring the P4 tables on the FPGA, handling CN registrations, adjusting the priority of each node according to feedback, and predicting epoch boundaries. It is implemented as a Golang application that runs on the LB CP. The architecture of the CP is divided between Presentation, Abstraction, and Control

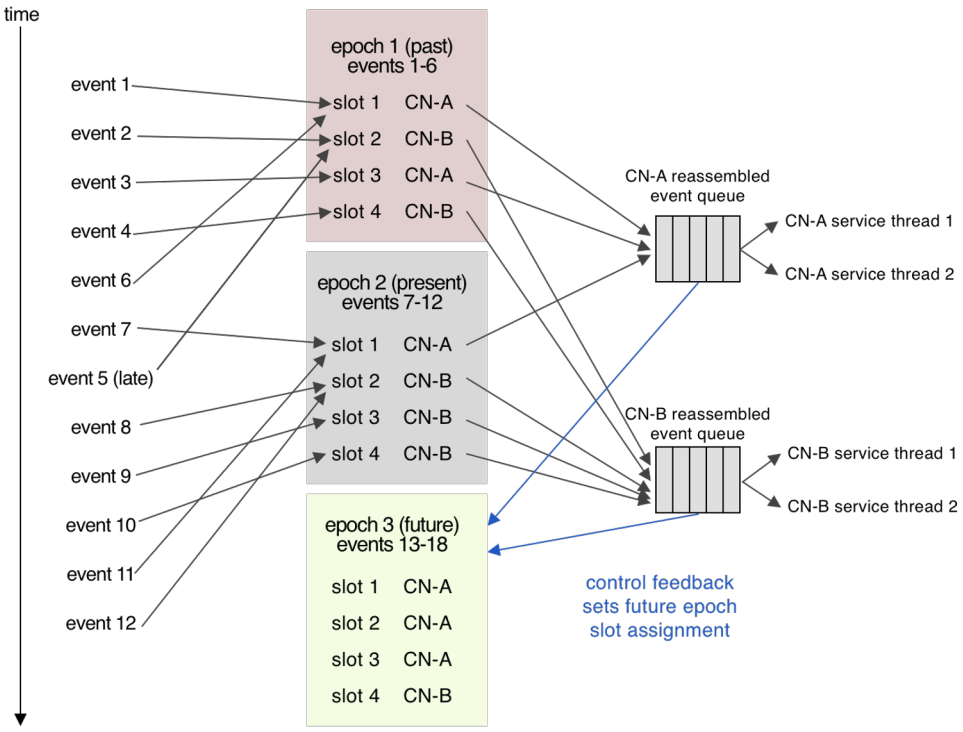

Figure 1. Conceptual model of the EJFAT system

(PAC). The abstract components model epochs, sessions, and configuration in ways that are intuitive to manipulate, similar to how ORMs model database schemas. The presentation components accept the abstract models and render them into P4 table entries or log output. The control components listen for feedback and event numbers and update the abstract models. This separation in combination with the comparison of previous to next states allows developers to implement methods of assigning slots to CNs by simply writing a function that returns an array of integers. The presentation components will automatically determine the correct updates to make to the FPGA.

The distribution of slots to CNs is important in modelling the behavior of the overall system since it determines the distribution of inter-arrival times. In dynamic mode, feedback is required from all registered CNs. Each node n begins with equal weight (weight(node, 0) = 1 for all nodes), and the control feedback (u(node, epoch)) is used to adjust their number of slots (slots(node, epoch))

$$\text{weight(node, epoch)} = \text{weight(node, epoch} - 1) + u\text{(node, epoch)} \tag{1}$$

$$\text{slots(node, epoch)} = \left\lfloor \frac{\text{weight(node, epoch)(max slots)}}{\sum_{n \in \text{nodes}} \text{weight(n, epoch)}} \right\rfloor \tag{2}$$

For example, if in Epoch 1 (the second epoch), CN-A sent a control signal of 0.1, CN-B sent a control signal of 0.0, and CN-C sent a control signal of -0.1 and there are 512 slots, Epoch 2 would be:

$$\text{slots(CN-A, 1)} = \left\lfloor \frac{(1.1)(512\,\text{slots})}{(1.1)+(1)+(0.9)} \right\rceil = 188\,\text{slots}$$
$$\text{slots(CN-B, 1)} = \left\lfloor \frac{(1.0)(512\,\text{slots})}{(1.1)+(1)+(0.9)} \right\rceil = 171\,\text{slots}$$
$$\text{slots(CN-C, 1)} = \left\lfloor \frac{(0.9)(512\,\text{slots})}{(1.1)+(1)+(0.9)} \right\rceil = 154\,\text{slots}$$

Since 188 + 171 + 154 = 513 > 512, we need to remove one of the slots so that exactly 512 are assigned. The slot will be removed from the last node. In the case that the rounded slots sum to less than 512 and need to be added, slots are added to the node that asked for the most new work, in this example CN-A. This method of slot assignment allows the LB to adapt to heterogeneous cluster environments as long as there is enough overall capacity.

3 Simulation Overview

The simulation models a computational cluster servicing a noisy data stream of events from an LB instance using previously measured processing latencies used as a tabulated processing rate probability distribution for latency noise. For a cluster size of 4 CNs, this determined a sending event rate match of 456Hz for the cluster. To ensure EJFAT could work in both homogeneous and heterogeneous environments, we choose both symmetric and asymmetric sets of CNs for simulated clusters with the node event service rates of (114Hz, 114Hz, 114Hz, 114Hz) and (91Hz, 106Hz, 122Hz, 137Hz) respectively, both giving an aggregate rate of the required 456Hz.

Mean sending rates of events between 420-456 Hz were simulated to measure the responses of the cluster for both configurations under varying load conditions. The simulation models noisy event disbursements from the LB to the cluster, Event Service Rates for each CN, Dynamic Scheduling Policies for Proportional, Integrative, Derivative (PID), and Q-Learning based controllers.

Measures of performance for the simulation in this study are the evolving FIFO levels that each CN experiences during the course of the experiment and the evenness of queue levels among CNs in the cluster.

3.1 Event Disbursement Modeling

Noisy streams of events from the LB to each CN were modeled as a Poisson process using a dynamically adjusted rate parameter as dictated by the CPs current scheduling policy for the cluster, where the scheduling policy is adjusted each new epoch represented as each iteration of the simulation. The CP maintains the aggregate delivery rates for each experiment as given in Section (3).

3.2 Event Processing Rate Modeling

Previous reconstruction studies (unpublished) suggest the optimal number of cores as 32 for a single reconstruction service on a CN. For a CN with 128 cores, this indicated use of 4 independent reconstruction services, where each service instance has a nominal service rate of $\approx$ 28.5Hz and is the rate used for each service thread (4 total for each CN) in the symmetric CN cluster. For the asymmetric cluster, each service thread rate is re-scaled on each node for all threads to yield the individuated aggregate CN service rates as indicated by Section (3).

3.3 LB Control Plane Modeling

The primary challenge for the CP, both real and simulated, is to dynamically adjust its scheduling policy so as to apportion events to CNs based on indications that nodes are under stress. This stress is measured as RE FIFO backlogs induced either by prior misallocations of events or random walk behavior induced by *process noise* both in the RE and due to the dynamics of previous noisy disbursements.

An important consideration of CP modeling is that a new scheduling policy determined in iteration i is not available until iteration $i + 2$ due to processing latencies incurred within the real CP.

3.3.1 PID Control

One dynamic form of control is modeled as a PID controller with appropriate control constants determined by previous experimentation for this particular simulated EJFAT system. The PID control effected is conventional using the the *process value* input from each node as:

$$p_n = F_n \div C_n \tag{3}$$

where F is the current fill count and C is the queue capacity of the CN, and the *control value* for each CN returned by PID uses the PID values of $K_p = 10^{-4}$, $K_d = 10^{-1}$, and $K_i = 10^{-4}$ with a *leaky integrator* that loses 10% of its integrative *accumulated error* state value each iteration. These control values c_n for each CN are then used to model the new scheduling policy for the next epoch, s.t.:

$$SP_n^{new} = SP_n^{old} + c_n \tag{4}$$

and the new SP_n must be re-normalized as

$$SP_n = SP_n^{new} \div \sum_n^N SP_n^{new} \tag{5}$$

where SP_n is the percentage of the aggregate event rate from the LB that will be disbursed to CN_n in the next available future epoch.

3.3.2 Q-Learning Control

A second form of control was modeled as a Q-Learning (QL) variant of *Reinforcement Learning* (RL) where the reward was modeled as the inverse normalized percent queue fill level on each CN or

$$R_n = 1 - F_n \div C_n \tag{6}$$

where F is the current fill count and C is the queue capacity of the CN. The QL algorithm then seeks to maximize the reward where the maximum reward is unity. The resulting values of the Q value can then be used directly to model the new scheduling policy for the next epoch as

$$SP_n = Q_n \div \sum_n^N Q_n \tag{7}$$

Note that the scheduling or disbursement policy for each CN is a *global* function of all Q values across the cluster. A learning rate value of $\alpha = 0.1$ gave best performance with an

associated exponential decay of the learning rate across the simulated number of epochs (as is customary) of the form:

$$\alpha = \alpha_0 \times exp(-5 \times i \div NI) \tag{8}$$

where $\alpha_0 = 0.1$, i is the iteration number and NI is the total number of iterations equal to 10,000. It was determined experimentally that the QL *discount rate* was not useful in this application.

4 EJFAT Simulation Execution

The simulation for each candidate event send rate (queuing theory *utilization*) in Section 3 is effected as follows:

- Lacking any prior calibration, set initial Schedule Policy for all cluster CNs as uniform.
- For all CNs set queue lengths of 1,000
- For all CNs for each iteration
  - For all control policies use policy to determine new CP schedule policy (disbursements)
  - For all CNs determine queue level from disbursement, current queue level, and service rate using M/M/4 queue service model.

5 EJFAT Simulation Results

A baseline simulation was conducted at each rate level in Section (3) using no control policy with the scheduling policy fixed at the known proper rates based on cluster design (symmetric, asymmetric). All results are assessed against the baseline. The Figures (2) and (3) graphically summarize these results and are color coded such that blue = no control, green = PID, purple = QL.

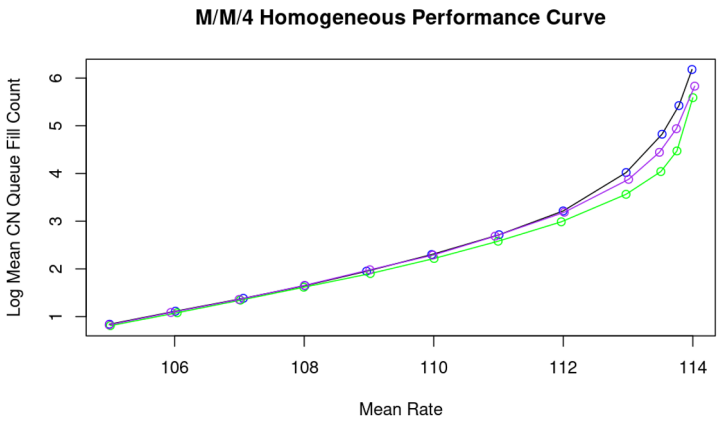

Figure 2. Symmetric Mean Queue Sizes By Rate

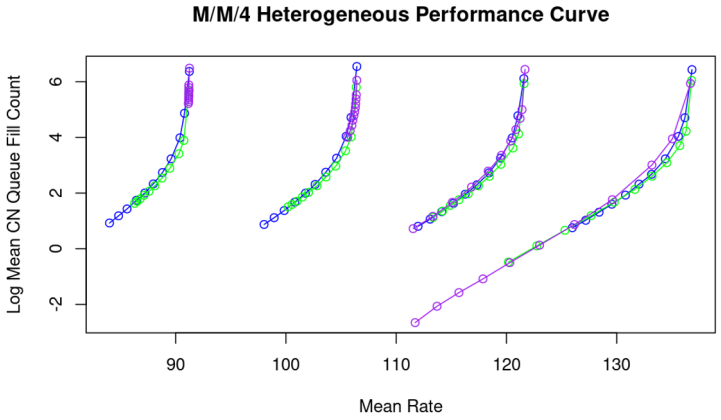

Figure 3. Asymmetric Mean Queue Sizes By Rate

6 Comparison with the real control plane

The use of simulation allows much faster iteration and testing of control algorithms and parameters compared to making changes to the four programs that make up the real system, primarily because the simulation can run much faster than real time. In order to ensure simulated results were valid, we continuously compared the simulation results to results from the real control plane when it was possible to do so. We set up tests for the real control plane by using a Docker environment that simulated a small end-to-end system. One container was responsible for generating events, another container for the control plane, and four containers representing the cluster nodes. The event generator container and cluster node containers were configured to match the parameters of the simulations.

We found correctable differences between the simulation and the real system related to event rate calibration and PID controller implementation. We implemented changes to both the simulation and real system to bring these into agreement. These changes both improved the accuracy of the simulation and the performance of the real system.

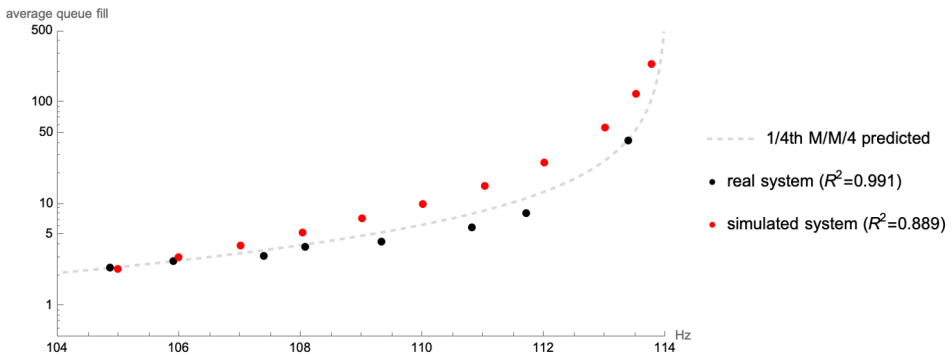

Figure 4. Average queue fills of real and simulated system by arrival rate

As a baseline comparison, we ran a series of tests at different utilization levels with no dynamic control. We then compared the average FIFO queue lengths per-CN of the real system and simulated system.

We found that both the simulation and the real system are reasonably well described by the curve for an M/M/4 queue with the same rate parameters when that curve is scaled to $\frac{1}{4}$ its original values. Both the simulation results and the real system results show a slow build up follow by rapid takeoff at utilization levels close to 100%, which is characteristic of queuing systems. Since at high utilization the simulation has higher queue levels than the real system, the simulation can be considered pessimistic, and we can be encouraged that if a particular tuning works well for the simulation at these high rates, the real system is likely to perform better than the simulation would suggest.

7 Conclusions

From the data for the charts in the Simulation Results section we make the following conclusions:

- For symmetric clusters (a.k.a homogeneous), QL control reduces aggregate queue levels about 40% from the no-control case, while for PID it is 50% with PID achieving a remarkable evenness across CNs while for QL the unevenness is about the same as the no-control case.
- For asymmetric clusters (a.k.a heterogeneous), QL functions as an ineffective control policy generally inflating mean queue levels for CNs above the no-control case, and not achieving a balance across the cluster. This we believe is due to the passive nature of control from QL since it is formally a *learning* system that learns the correct rate for each CN but does not provide *active* control. This can be seen in the charts by the placement of purple data points compared to PID and no-control for most tested rates. Also we note the overall unevenness of queue levels not present for PID control and that show a notable skew especially for lower utilization rates.
- For asymmetric clusters, the PID control functions best by determining proper mean rates for all CNs, reducing queue levels from the no-control baseline by 40% and also balancing the queue levels across the cluster to a high degree. We believe this is attributable to the *active* nature of PID as a control policy. This can be seen in the charts by the placement of green data points below levels for both QL and no-control for each tested rate with the exception of the two highest rate CNs due to the previously mentioned uneven queue levels for QL control.

In summary, we found that PID control was superior to QL and static configuration in both the heterogeneous and homogeneous cases. Therefore, future development work for EJFAT will focus on optimizing the performance of PID control.

References

[1] Stacey Sheldon, Yatish Kumar (ESnet), Michael Goodrich, Graham Heyes (Jefferson Lab), "EJ-FAT Joint ESnet JLab FPGA Accelerated Transport Load Balancer", 2022 INDIS Workshop, https://doi.org/10.48550/arXiv.2303.16351, Mar 28 2023.

[2] Michael Goodrich, Carl Timmer, Vardan Gyurjyan, David Lawrence, Graham Heyes (Jefferson Lab), Yatish Kumar, Stacey Sheldon (ESnet), "ESnet/JLab FPGA Accelerated Transport," in IEEE Transactions on Nuclear Science, vol. 70, no. 6, pp. 1096-1101, June 2023, doi: 10.1109/TNS.2023.3243871.