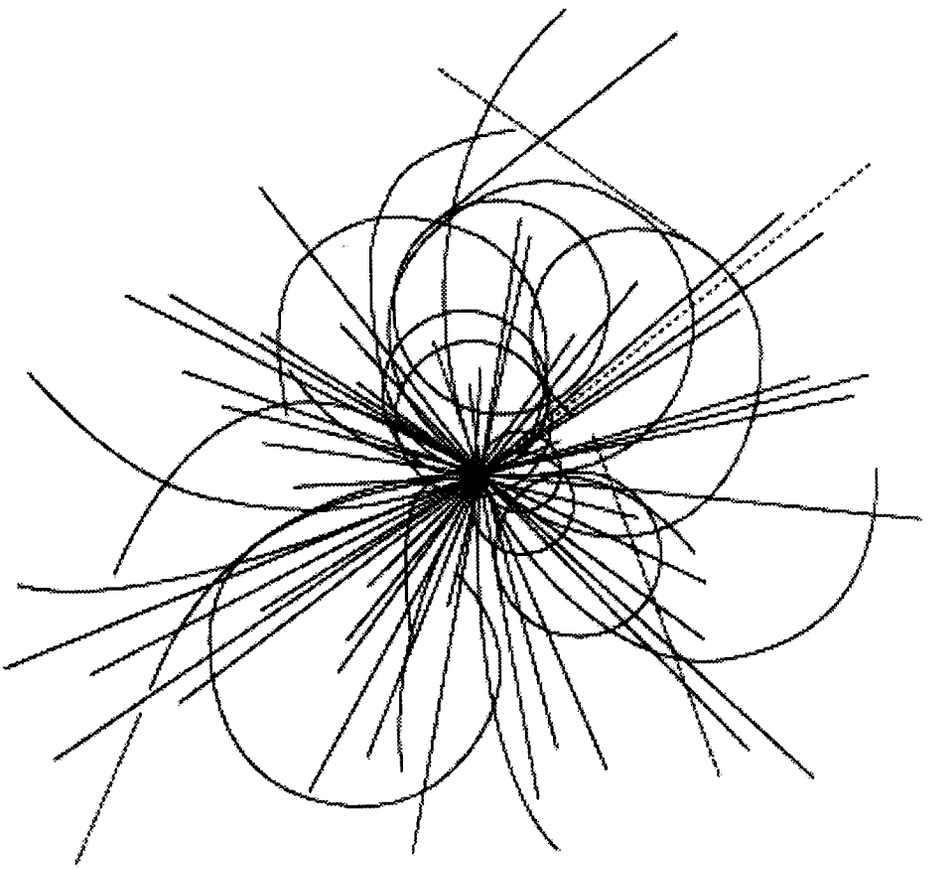
SSCL-Preprint-524
October 1993
Distribution Category: 400

C. Wang
Y. Chen
J. Dorenbosch
J. Lee
R. Sayle

A Simulation of the SDC On-line Processing Farm



Superconducting Super Collider Laboratory

SSCL-Preprint-524

A Simulation of the SDC On-line Processing Farm*

C. Wang, Y. Chen, J. Dorenbosch, and J. Lee

Superconducting Super Collider Laboratory†
2550 Beckleymeade Ave.
Dallas, TX 75237, USA

R. Sayle

University of California
Irvine, CA 92717

October 1993

---

*Presented at the 1993 *IEEE Nuclear Science Symposium and Medical Imaging Conference.*

†Operated by the Universities Research Association, Inc., for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

A Simulation of the SDC On-line Processing Farm

C. C. Wang, Y. M. Chen, J. Dorenbosch & J. A. Lee
Superconducting Super Collider Laboratory[1]

R. P. Sayle
University of California at Irvine, Irvine, CA 92717

Abstract

In the Solenoidal Detector Collaboration (SDC) data acquisition system (DAQ), an enormous amount of data flows into a processor farm for extraction of interesting physics events. To design an efficient on-line filter, the operations in the farm must be carefully modeled. We present a simulation model developed at the Superconducting Super Collider Laboratory which efficiently allocates physics events to the farm.

INTRODUCTION

In the Solenoidal Detector Collaboration (SDC) data acquisition system (DAQ), collisions ('events') are selected in real time by a two level pattern recognition system. These filtering systems, called the level 1 and level 2 triggers, classify accepted event by trigger type. The complete data set for events accepted by level 2 flow into an on-line computing subsystem for third level event selection. The on-line computing subsystem consists of approximately 1000 processors operating in parallel[2]. It is known as the processing farm. The farm's event input rate can be as high as 10 kHz. Some processors handle only a specific trigger types while others may process multiple trigger types. A scheme to balance the allocation of events to the farm is needed to increase system throughput and improve processor utilization.

SIMULATION MODEL

To investigate various control alternatives we have developed a detailed behavioral model of the DAQ system. The model is written in MODSIM, a discrete event simulation language developed by CACI[3]. The farm is modeled as a set of processors, arranged in rows. Each row may contain a different number of processors. Row managers, specialized processors connected to each row, store and forward event requests. The column of row managers connects to a farm manager which provides centralized processor control.

The farm connects to three other DAQ subsystems (Fig. 1). The event builder (EVB) takes detector data fragments from the front-end electronics crates and constructs complete events. The EVB passes the events to the farm as directed by the event data flow control subsystem (EDFC). The EDFC communicates with the second level trigger, the farm, and the EVB. The second level trigger provides the EDFC with the trigger type of all events available for third level processing. The EDFC receives requests from the farm, matches them to events available from the level 2, and instructs the EVB to forward event data to specific processors. The farm is also connected with the archival storage so that the events satisfying the third level selection criteria can be written to tape.

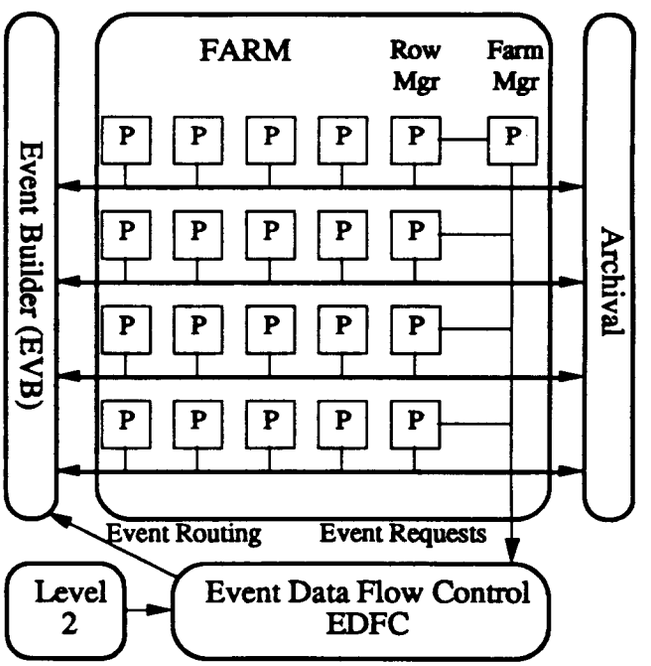


Fig. 1. The on-line processor farm architecture, and its connections to other SDC DAQ subsystems.

EVENT DATA FLOW CONTROL

The data flow is controlled by the EDFC. A set of two Processor-Trigger Type tables (PTTs) is used to bring events with a certain trigger type to the pro-

cessors that request such data (Fig. 2). The first table, called the processor table (ProcTab), stores the trigger types requested by each processor. The ProcTab is an array of queues indexed by processor identification. The second table is essentially the inverse of the first one. Known as the trigger type table (TrigTab), it is an array of queues, indexed by trigger type. Each TrigTab array hold the set of processors interested in the trigger type given by its index.

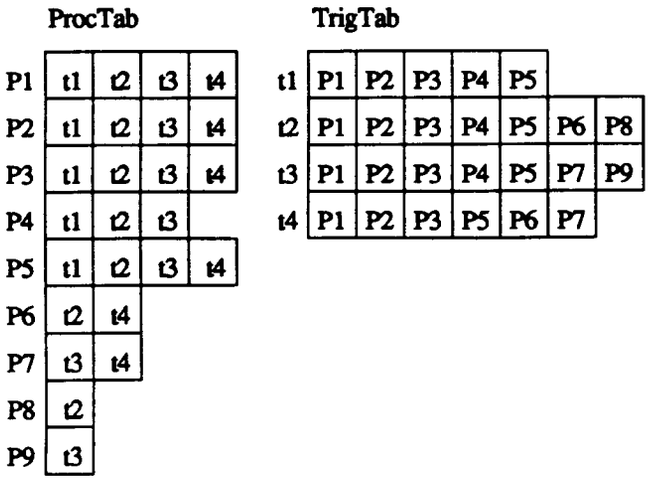


Fig. 2. Processor-Trigger Type tables. The ProcTab holds the trigger types each processor is interested in. The TrigTab stores processor numbers by trigger type.

The PTTs are used to balance the distribution of events with the same trigger type across all processors interested in the trigger type. When the EDFC receives an event request from a processor, it uses the ProcTab to provide a trigger type assignment to the request. If the ProcTab only has is single entry for this processor, it only handles one event type, and the EDFC assigns that trigger type to the request. If a processor can handle more than one trigger type, the ProcTab array for this processor holds multiple entries.

Next, the EDFC attempts to match the request to a pending second level event of the required trigger type. If a match is successful, the EDFC sends routing information to the EVB. Included in the routing message is the event number provided by the accept message form level 2, the processor identification given by the request, and the trigger type. If the EDFC fails to find a match, it stores the request in until an appropriate event arrives from the level 2 trigger.

Matching requests from processors that handle multiple types requires use of the PTTs. When the EDFC receives such requests, it uses the ProcTab to provide a match. Beginning with the first trigger type listed in ProcTab, the EDFC checks if an event is available. If not, the EDFC iterates through all relevant trigger types in the ProcTab until a match is found or the search fails. A successful search causes the EDFC to send a routing message to the EVB. After providing a successful match, the PTT round-robins ProcTab[p]'s entries by placing the type used at the end of the queue. This process ensures that the next assignment to processor p is likely to be different; which helps achieve a balance of types presented to each processor. A failed match is queued until a proper event arrives.

In order to service buffered requests, the EDFC waits for accepts to arrive from the second level trigger. Each time it receives an accept, the EDFC once again employs the PTT. The event's trigger type (t) in the accept message causes the EDFC to use the queue TrigTab[t]. TrigTab[t]'s entries in turn point the EDFC to queues in a Request Buffer. The EDFC searches through each queue for a request that includes trigger type t. If the EDFC finds a matching request, it sends the corresponding routing information to the EVB. The PTT also round-robins TrigTab[t]'s entries similarly to the procedure used to shuffle ProcTab[p]. The action ensures that the next search attempting to match a pending request with an accept uses the same processor only after all other applicable processors have been checked. This is the second load-balancing strategy the model utilizes. The EDFC queues unmatched events until a corresponding request arrives.

CONCLUSION

The model presents an efficient mode of communication between the SDC farm and EDFC. It enables users to solve data acquisition challenges by optimizing the processing farm architecture given the distribution of physics events.

---

[1] Operated by the Universities Research Association, Inc., for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

[2] SDC Technical Design Report, SDC-92-201, SSCL-SR-1215; April, 1992, SSC Laboratory, Dallas, TX.

[3] CACI product Company, 3333 North Torrey Pines Court, La Jolla, CA 92037. • (619) 457-9681.