A high speed Data Acquisition Collector for Merging and Sorting Data

Clyde C. W.Robson and Christian Bohm

Abstract— Distributed data acquisition systems have a need for high throughput and a reliable delivery of data and commands in order to function properly. Connectionless oriented protocols like UDP provides a small overhead, but due to its unreliability it can be adequate to equip the application layer with additional services like flow control, data ordering and error checking. High performance buffers are important and necessary components here in order to not add further delays. When data, transmitted from a number of nodes are merged together in one node, not only serving as a collector of data, but also performing additional tasks like sorting of incoming data etc, high performance buffers and software architectures are equally important for adding as little delay as possible to the transmission chain. Involved in different projects we have faced this challenge and have managed to reach throughputs of 100 MB/s on 1Gb Ethernet with the developed architecture when implemented in Java. We will in our paper share our results when it comes to performance and ease of development, and discuss pros and cons with an equal implementation developed in C. first performance results will be presented

***Index Terms*—data collector, sorting, high speed networking**

I. Introduction

Global merging and processing of high-speed data obtained from distributed data sources is common in many detector systems. The processing is often of the type trigger processing or feature extraction where different signatures are searched. Here the processing can be separated into pre- and post processing. In many detector systems the preprocessing can be merging high granularity data into lower granularity to facilitate feature extraction and trigger decision. In digital coincidence systems the preprocessing will be global sorting of locally time-sorted event streams.

Some examples are PET cameras and distributed neutrino or cosmic ray detectors, such as Ice-cube in Antarctica. In PET cameras time sorted single events are recorded in modules and sent to a global coincidence unit for global sorting before searching for coincident events in running time windows. In Ice-cube data from Digital Optical Modules from each string are time sorted in hub processors and then sent to a global trigger processor for global sorting before applying trigger conditions. We will in the following describe a method for transferring and sorting data at large data rates.

The network topology considered is a tree where low speed (10 Mb/s) branches are merged successively into higher data rates (100Mb/s and finally 1Gb/s). The data rates are increased when approaching the trunk to avoid unnecessary bus contention. Once entered into the processor via a small number of high-speed lines the data is split into different circular buffers, one for each data source.

We assume further that data is sent according to the roller coaster principle where data frames are sent from the source when filled. However, if they are not filled after a certain time a frame transmission is forced. This will help to even the data flow, but it does mean that sometimes very sparsely populated frames are transmitted.

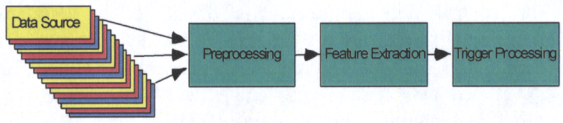

Fig. 1. The sorting engine as a block diagram.

II. The Sorting Algorithm

The sorting algorithm [1] starts from an array of time stamps. A tree structure is built by noting the outcome from pair-wise comparisons of nearby pairs keeping the least values. The result will be an array of half the length. This is process is repeated until the result contains only one item. This is the least value in original array.

When this value is removed and placed in the output stream a new value must take its place in array. All comparisons related to this value are then performed, one for each level until the root is reached. The new root value is the next least value, larger than the previous.

This repeated, producing a sequence of sorted time stamps, until one of the data buffers are found empty.

Manuscript received November 14, 2008.

C. Robson is with the. Department of Physics, University of Stockholm, Sweden (phone: +46855378694; fax: +46855378601; e-mail: clyde@physto.se).

C. Bohm is with the. Department of Physics, University of Stockholm, Sweden (e-mail: bohm@physto.se).

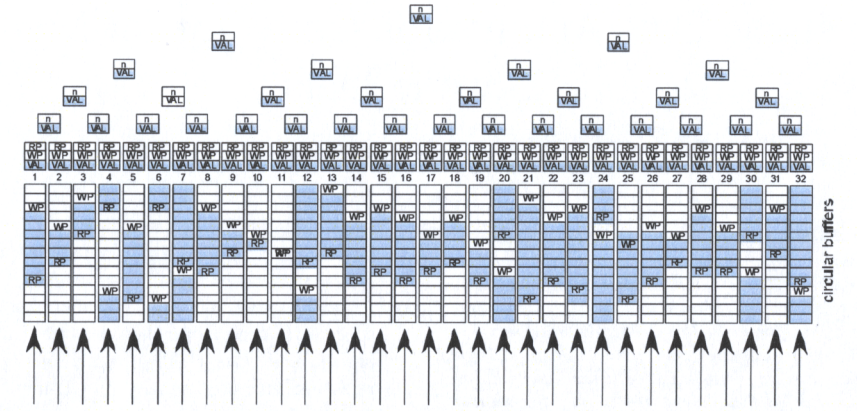

Fig. 2. The reverse binary search algorithm.

III. System Overview

Since we are talking about a network based, distributed architecture for data flow, all components in the chain has an impact on performance. The Control Server is therefore implemented as a high-speed select server with a minimum of threads for achieving maximum speed. The architecture for control and monitoring is based on the System Adaptable Application Layers (SAAL) concept [2] developed at the University of Stockholm. This is developed with ease of use and ease of development in mind. It is highly adaptable on all levels of the system, the embedded controllers are operation based [3] and can replace the set of commands it can execute even during runtime without the need for recompile and redeploy. The Control Server serves as an access point for both users and other parts of the system. AJAX-based [4] clients, highly adaptable and easily changed for new purposes can be used with great success for control and monitoring of the system.

For the Sorting Engine two major aspects for software development has been carefully studied and taken in consideration, network and system programming and efficient data structures adapted for the task and sorting algorithms.

The network components are using highly efficient event based servers, thus making it possible to decrease the number of threads in the system. Avoiding contexts switches decreases the load on the underlying operating system and speeds up the system as a whole. The use of threads and synchronization has been carefully considered with emphasize of speed in focus.

IV. Conclusion

With straight transmission without subsequent processing we have achieved close to 100 MB/s on a 1Gb/s Ethernet line. With a simple sorting mechanism operating on 86 circular buffers we have reached 20 MB/s, but with the more optimized sorting algorithm described above we expect better performance.

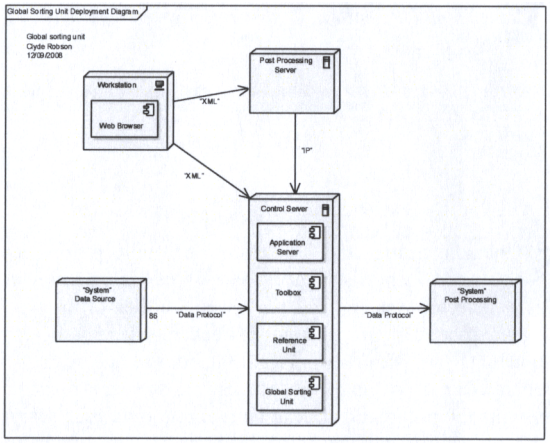

Fig. 3. System depicted as a UML deployment diagram.

References

[1] C. Bohm and C. Robson, "Reverse binary search – an efficient way to merge and sort streams of sorted time stamp data" submitted to Trans. Nucl. Sci.

[2] C. Robson, S. Silverstein and C. Bohm, "Application-Level Protocols for Network-Based Control and Data Acquisition", IEEE NSS/MIC 2007 Conference proceeding.

[3] C. Robson, S. Silverstein and C. Bohm, "An Operation-Server Based Data Acquisition System Architecture ", IEEE Real Time 2007 Conference proceeding.

[4] C. Robson, S. Silverstein and C. Bohm, "Implementing Clients for Control and Monitoring Using AJAX ", N15-165 IEEE NSS 2007, Honolulu, Hawaii, USA Conference proceeding