Flexible data transport for the online analysis in a particle physics experiment.*

D. Klein[1,2] *and M. Al-Turany*[1]

[1]GSI, Darmstadt, Germany; [2]University of Applied Sciences, Darmstadt, Germany

Motivation

The next generation of experiments at GSI/FAIR share the common software framework *FairRoot*[1] which provides the building blocks for offline analysis. The software is well-suited for batch-processing.

Online analysis processes the data on-the-fly, filters the interesting physics data and it needs to reduce the raw data rates by three orders of a magnitude.

The FairRoot framework must be extended to support the continuous pipeline-processing scenario of the online analysis.

During prototyping it is very important to be able to often change data paths and processing elements in the pipeline. The inter-process data transport must be reliable and efficient within one node and over the network between nodes.

The Data Transport Framework

A data transport framework has been proposed and implemented[3]. Each framework component, a so-called device, runs in its own operating system process. A device has a variable number of inputs and outputs which can be flexibly interconnected to each other. The data transport of each connection is built upon a highly efficient message queuing library (ZeroMQ[2]). The message queuing technology itself is reliable by definition. For inter-node connections the transport relies on the Linux TCP/IP stack which works over Ethernet and Infiniband. Intra-node connections are realized on Unix' named pipes.

The framework provides a basic set of devices: *sampler*, *processor*, *splitter*, and *merger*.

A sampler device starts any pipeline during simulation. It feeds simulated data from root files into the pipeline. The current implementation supports control over the sending speed in events per second.

Processors are devices which operate on the contents of the messages (events) - they constitute the actual processing instance in the pipeline. The current implementation features a plugin system for processing algorithms via *ProcessorTask* similar to the *FairTask* class in FairRoot.

Splitters and mergers are devices to de/multiplex the data path. Whenever a processing instance must be distributed over serveral nodes/processes due to bandwidth and/or cpu limitations data paths can be splitted and merged before and after processor devices.

Let $n, m \in \mathbb{N}$, then a many-to-many mesh of n splitters to m mergers serves as a transposing engine (with n inputs and m outputs) of signals from n subdetector links to an over m nodes/processes distributed processing instance. Fig. 1 illustrates an example which was successfully run with the current implementation, monte-carlo data generated with PandaRoot[1] and a dummy processor task.

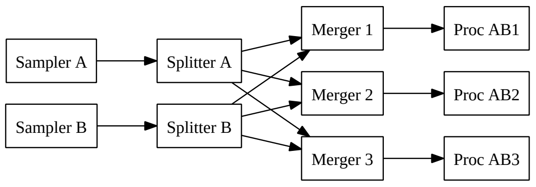

Figure 1: 3-way distributed processor fed from two sources.

TCP throughput

A bandwidth utilization of 99.7% of the theoretical maximum for TCP over GigabitEthernet has been seen. In the test a sampler was connected to a processor on two identical nodes - CPU load was 25% of one core (2.13 GHz Intel Xeon) per device.

Next Steps

- Integration into FairRoot repository,
- configuration management via directory service,
- adding support for time-series simulated data,
- improving the user interface,
- reducing latency even more by using shared memory transport of ZeroMQ between processor tasks each running in their own thread,
- collecting runtime monitoring information.

References

[1] http://fairroot.gsi.de

[2] http://www.zeromq.org

[3] D. Klein, "Flexible data transport for the online analysis in a particle physics experiment", Feb 2013, Bachelor thesis.

* thanks to the KoSI program