Streaming data processing with FairMQ

A. Rybalchenko[1] *and M. Al-Turany*[1]
[1]GSI, Darmstadt, Germany

The FairRoot framework [1] is a framework for simulation, reconstruction and data analysis of particle experiments. Currently the framework is being extended to provide support for simulation, reconstruction and analysis of free streaming data. A number of components were introduced to the framework, under the common name of FairMQ, which allow flexible construction of topologies to distribute and process free streaming data. FairMQ is based on a prototype framework [3], developed in 2012/2013 at GSI, which relies on the ZeroMQ library for the transport and organization of data between system processes and/or network nodes. An example topology is presented in Figure 1. It includes two separate nodes (blue), each containing a number of processes (green and grey). Arrows indicate the data flow.

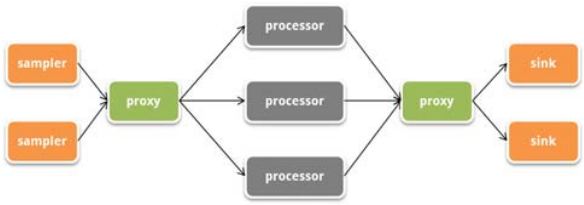

Figure 1: FairMQ example topology.

An example is implemented in FairRoot tutorials, which shows the workflow within the new scheme [7]. The example workflow demonstrates how to configure and run a topology, including components to read, process, distribute, collect and write data. The topology can be configured to run on a single node or on any number of nodes, allowing the user to monitor the system load more granularly and to add additional workers dynamically when necessary. A performance comparison with the traditional workflow was presented in [2], demonstrating significant performance benefits. On a 2 x 2.4 GHz Intel Xeon quad core node, the traditional execution setup with 8 parallel tasks reached a throughput of 2660 events/s. With the same data, FairMQ streaming system reached 7320 events/s, utilizing 9 system processes - 3 for task execution, 4 for I/O and 2 for data distribution.

Among the improvements which were introduced to FairMQ is a new component called the proxy. The proxy replaces splitter and merger components, which were used for merging/splitting of the data stream. The Proxy is more flexible than splitter/merger, because new nodes/processes can be added to it dynamically, without changing the configuration or even stopping the process. Using the proxy also simplifies the configuration, since it only occupies one network port for input and one for output, while splitter/merger would occupy a port for every new connecting node.

Moreover, FairMQ has a new layer that abstracts the specific transport implementation. By keeping framework and transport code independent, the framework can be easily adapted to emerging technologies in the future.

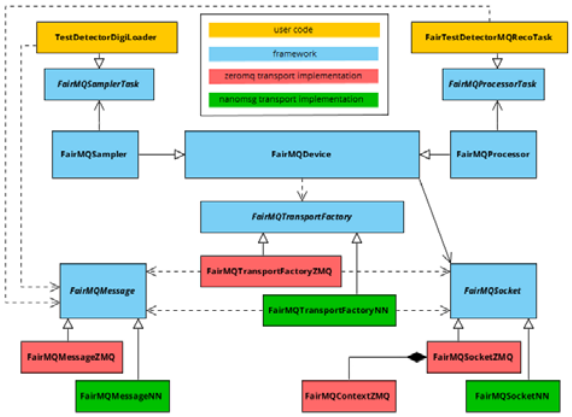

Figure 2: Transport layer abstraction.

At the moment two transport implementations can be used: ZeroMQ [4] and nanomsg [5]. Nanomsg is an emerging communication library, developed by the authors of ZeroMQ. While still in an early alpha stage, nanomsg aims to offer a number of advantages in comparison to ZeroMQ [6], which could to be useful for FairRoot. Most notable among these are better Zero-copy support (with shmem and RDMA) and a formal API for adding new transports such as Infiniband and Websocket.

References

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

[2] M. Al-Turany, D. Klein, A. Manafov, A. Rybalchenko, F. Uhlig: Extending the FairRoot framework to allow for simulation and reconstruction of free streaming data. accepted for publication by, Journal of Physics: Conference Series (2013).

[3] D. Klein: Flexible data transport for the online analysis in a particle physics experiment. Bachelor thesis, Hochschule Darmstadt (2013).

[4] ZeroMQ: http://zeromq.org/.

[5] nanomsg: http://nanomsg.org/

[6] Differences between nanomsg and ZeroMQ: http://nanomsg.org/documentation-zeromq.html

[7] FairRoot Tutorial 3: https://github.com/FairRootGroup/FairRoot/tree/master/example/Tutorial3