RDMA-accelerated data transport in ALFA

Dennis Klein, Mohammad Al-Turany, Thorsten Kollegger, Alexey Rybalchenko
GSI Helmholtz Centre for Heavy Ion Research GmbH, Darmstadt, Germany

DRAFT DRAFT DRAFT

CHEP 2018 Conference, Sofia, Bulgaria

Motivation

- FairMQ lacked support of an efficient RDMA-accelerated inter-node transport
- Fine in the beginning of the project, because it had production quality TCP/IP based backends

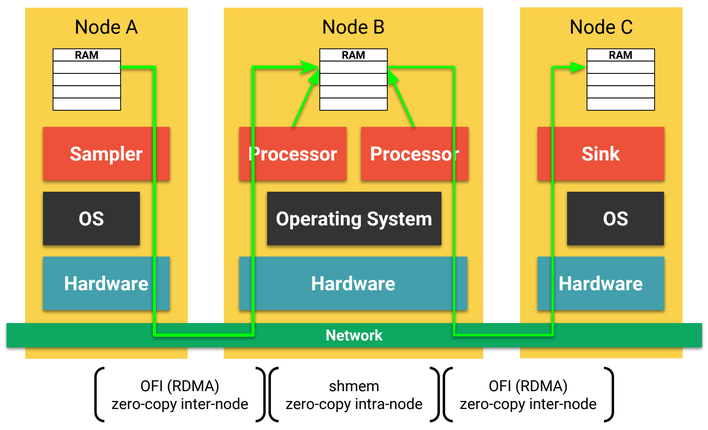
- To become a viable choice also for HPC supercomputers, the **new "OFI" transport** is developed
- Targets primarily **Infiniband fabrics**

node	process	address format	zeromq	nanomsg	shmem	ofi (NEW)
intra	intra	`inproc://endpoint`	✔	✔	n/a	n/a
intra	inter	`ipc://endpoint`	✔	✔	✔	✘
		`tcp://host:port`	✔	✔	✔	✔
inter	inter	`tcp://host:port`	✔	✔	n/a	✔
		`verbs://host:port`	✘	✘	n/a	✔

✔ not zero-copy ✔ zero-copy ✘ no support planned

Use case

- True **zero-copy send and receive**
- Build efficient distributed processing pipeline:

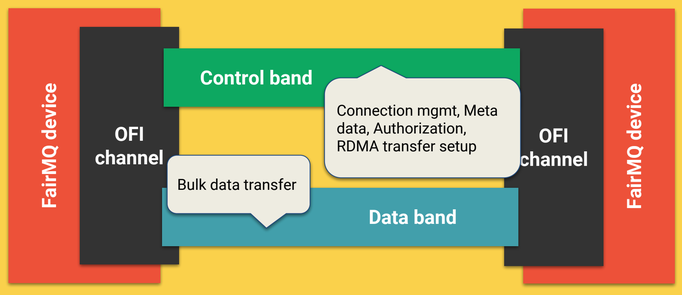


- User-space processes (in red, so-called "devices") use the exact same FairMQ message queuing API
- Change transport per channel without recompilation, just by tweaking the runtime configuration

OpenFabrics Interfaces (OFI) Transport

- Based on the OFI `libfabric` technology
- Maps a subset of `libfabric` API to the message queuing user API of FairMQ.
- Hides low level details of `libfabric`
- Supports **zero-copy send and receive** operations
- Optimizes for **high bandwidth utilization**
- Usable **with existing FairMQ user API**
- Targets Infiniband fabrics now, but can support practically every major HPC interconnect technology (supported by OFI) in the future with little extra development
- Integrates with Boost.Asio event loop, see `asiofi`[3] (implementation detail).
- Specialized local memory allocation strategies are implemented (allocate physical pages eagerly, support hugepages)

FairMQ OFI "PAIR" Connection
is realized with a TCP/IP based control band implemented with ZeroMQ, and an RDMA based data band implemented with OFI libfabric.



Ongoing Work:

- Vectored I/O support
- Scalability Protocols (PUB/SUB, PUSH/PULL, REQ/REP)
- Performance profiling, tuning
- Memory allocator improvements

Performance

TBD

Boost.Asio Integration

- `Libfabric`'s API is inherently **asynchronous** and written in C.
- ⇒ Write **safer and more easy to use C++ API**.

Boost[8] is a collection of expertly-designed, free, peer-reviewed, portable C++ open source libraries.

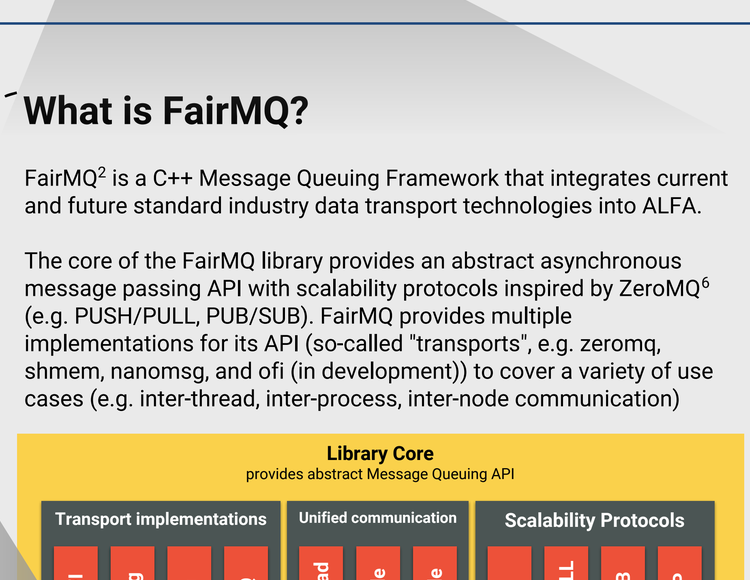
Boost.Asio offers
- a mature asynchronous programming model,
- and a portable integration with the I/O multiplexing service of the operating system.

`asiofi`[3] is the result of writing C++ Boost.Asio language bindings for `libfabric`, example:

```
boost::asio::io_context io_context;
// skip some initialization code
asiofi::endpoint endpoint(io_context, domain);
endpoint.connect([]{
  // gets called by asio event loop
  // on connection
});
io_context.run();
```

What is ALFA?

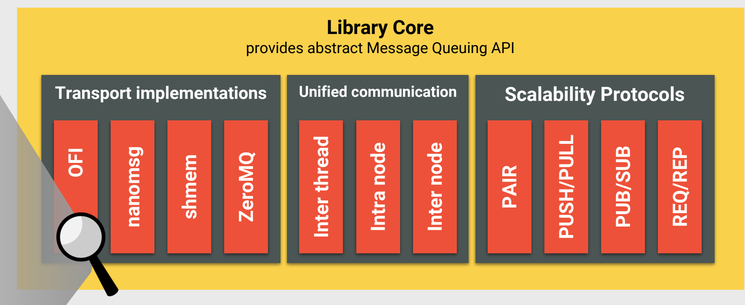
ALFA[9] is a modern software framework for simulation, reconstruction and analysis of particle physics experiments. ALFA extends FairRoot[1] to provide building blocks for highly parallelized and data flow driven processing pipelines required by the next generation of experiments, e.g. the upgraded ALICE detector or the FAIR experiments.



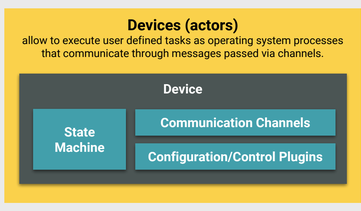
What is FairMQ?

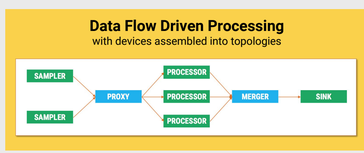
FairMQ[2] is a C++ Message Queuing Framework that integrates current and future standard industry data transport technologies into ALFA.

The core of the FairMQ library provides an abstract asynchronous message passing API with scalability protocols inspired by ZeroMQ[6] (e.g. PUSH/PULL, PUB/SUB). FairMQ provides multiple implementations for its API (so-called "transports", e.g. zeromq, shmem, nanomsg, and ofi (in development)) to cover a variety of use cases (e.g. inter-thread, inter-process, inter-node communication)



In addition to this core functionality FairMQ provides a framework for creating "devices" - actors which are communicating through message passing. Device execution is modelled as a simple state machine that shapes the integration points for the user task.





Devices also incorporate a plugin system for runtime configuration and control. The user can develop her own plugins for seamless integration with external services.

What is RDMA?

With **R**emote **D**irect **M**emory **A**ccess (RDMA) a program can read from or write to the memory of another program running on a different computer. RDMA transfers are hardware-accelerated, minimize CPU load and bypass the operating system (no extra memory copies). This enables high-throughput, low-latency networking.

What is OFI / `libfabric`?

Open**F**abrics **I**nterfaces (OFI) is a framework focused on exporting HPC fabric communication services to applications. It is developed by the OFI Working Group (OFIWG), a subgroup of the OpenFabrics Alliance[5] (OFA).

`Libfabric`[4] is a core component of OFI. It is the library that defines and exports the user-space API of OFI. `Libfabric` is open source, focuses on the needs of HPC users, and supports a variety of high-performance fabrics and networking hardware (e.g. Omni-Path, InfiniBand, Cray GNI, Blue Gene, iWarp RDMA Ethernet, RoCE and more).

References

[1] FairRoot - https://github.com/FairRootGroup/FairRoot
[2] FairMQ - https://github.com/FairRootGroup/FairMQ
[3] asiofi - https://github.com/FairRootGroup/asiofi
[4] OpenFabrics Interfaces `libfabric` - https://ofiwg.github.io/libfabric/
[5] OpenFabrics Alliance - https://www.openfabrics.org/
[6] ZeroMQ - http://zeromq.org/
[7] Alexey Rybalchenko - "Shared Memory Transport in ALFA" - CHEP'18 Poster #305
[8] Boost - https://www.boost.org
[9] M. Al-Turany et al. - "ALFA: The new ALICE-FAIR software framework" - 2015, Journal of Physics: Conference Series, Volume 664

All URLs have been last visited on 9th July 2018.