FairMQ

Multiple transports and shared memory support

Alexey Rybalchenko
(GSI Darmstadt, FairRoot group)

ALICE Offline Week
CERN, March 31, 2017

Shared Memory

motivation

- **Provide faster IPC transport for large messages.**
  ZeroMQ/nanomsg involves at least one copy.

- **Share messages between processes on one node without copying.**

- **Enable combination of different transports within same device -> each data channel can have it's own transport.**
  E.g.: device receives data over network and sends it further over shared memory channel.

- **Any device/channel should be able to switch transport only via configuration, without modifying device/user code.**

Shared Memory

concept

- **Use boost::interprocess library for the management and allocation of shared memory.**
- **Transfer meta information about allocated messages (handle + size) via ZeroMQ.**
- **Keep the message passing ownership concept: sender transfers ownership of the message to the receiver. If the same message is needed by multiple receivers, it is copied.**
- **No support for unreliable communication patterns (PUB-SUB).**

Shared Memory

allocation+transfer performance

Node

2 x Intel Xeon E5-2660 v3 @ 2.60GHz
(20 Cores, 40 Threads)
128 GB RAM,
60 GB shared memory segment

```
$ bsampler --id bsampler1 --mq-config config/benchmark.json --transport shmem --same-msg false --msg-size <n>
```

```
$ sink --id sink1 --mq-config config/benchmark.json --transport shmem
```

Shared Memory

CPU usage

Limit throughput to ~2.5 GB/s, measure average CPU usage (one core)
(1MB message size)

	ZeroMQ:TCP	shmem
sender	~ 68.5%	~ 1.1%
receiver	~ 89.1%	~ 0.9%

Node

2 x Intel Xeon E5-2660 v3 @ 2.60GHz
(20 Cores, 40 Threads)
128 GB RAM,
60 GB shared memory segment

```
$ bsampler --id bsampler1 --mq-config config/benchmark.json --transport shmem(/zeromq) --same-msg false --msg-rate 2500
```

```
$ sink --id sink1 --mq-config config/benchmark.json --transport shmem(/zeromq)
```

Shared Memory

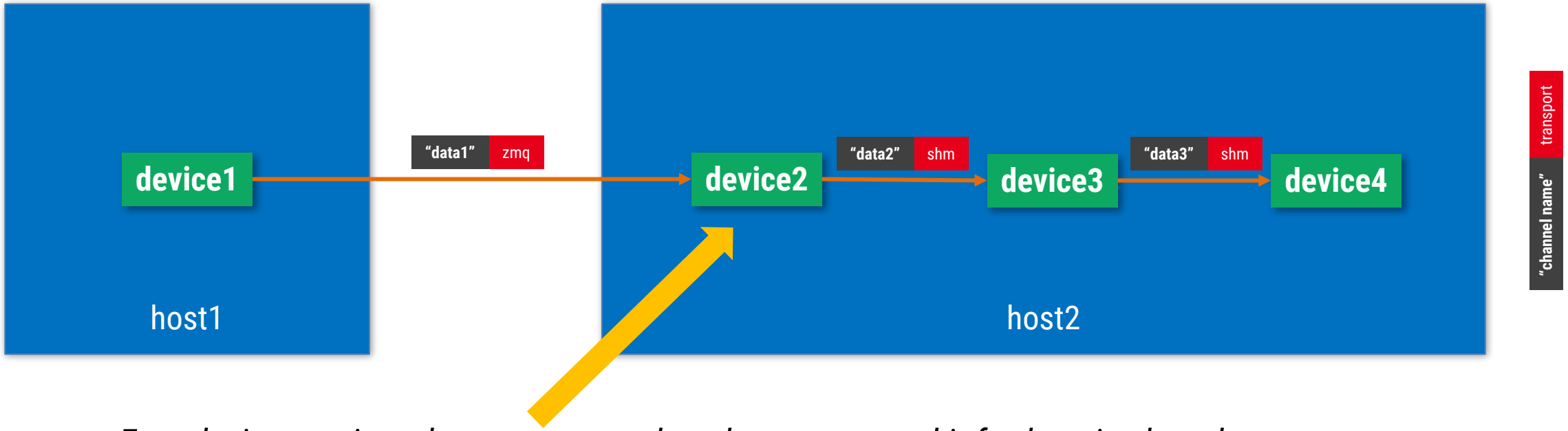
ongoing work

- **Improving shmem transport stability (in case of device crashes, restarts), watchdog via DDS task triggers.**
- **Extending the shmem transport configuration (segment size, …).**
- **Ensuring smooth combination with future RDMA transport (allow RDMA to read/write directly from/to shared memory area).**
- **Possibility to allocate a memory region, and transfer parts of it.**

Multiple Transports

motivation

Introduction of the shared memory transport created a need for having more than one implementation of the FairMQ transport interface within one running device:



E.g.: device receives data over network and wants to send it further via shared memory.

→ **Allow channels to use different transports.**

The device code should remain the same independent of the used transport or their combination!

Multiple Transports

transport per channel

Each device has default transport given to it via:

```
--transport <transport-name> (default: zeromq)
```

In addition to this, each channel can override the default to another transport (`zeromq`**,** `nanomsg`**,** `shmem`**).**

Multiple Transports

usage (1/2)

With multiple transports the basic functionality of the device remains unchanged.

The two basic functionalities that the transports provide are message creation and transfer:

```
NewMessage(...); // Creates a message with the default transport of the device.

Send(msg, “channelA“); // Sends message ‘msg‘ over ‘channelA‘, either with the transport
that this channel has configured. If the channel has no transport configured, sends with
the default transport.
```

Multiple Transports

usage (2/2)

In some cases a message allocated with the default transport is not compatible with the transport of a channel which must transfer it.

By default in such cases the message will be copied behind the scenes.

To avoid this copy, one can use following method to create a message for a specific channel:

```
NewMessageFor("channelA", ...); // Creates a message with the transport of 'channelA'.
```

Transports that are able to efficiently use already allocated memory, will be able to avoid this copy (RDMA+shmem).

multiple transports example	https://github.com/FairRootGroup/FairRoot/tree/dev/examples/MQ/multiple-transports

Thank you for your attention!

FairRoot	http://fairroot.gsi.de
FairMQ	https://github.com/FairRootGroup/FairRoot/tree/master/fairmq
FairMQ Examples	https://github.com/FairRootGroup/FairRoot/tree/master/examples/MQ