Libfabric dpa provider performance

Paolo Inaudi, Emilio Billi, Marco Aldinucci

October 31, 2015

A3Cube [1] is a US based company producing among other products the RONNIEE Express fabric [2]. RONNIEE NICs form an In-Memory Network, where a segment of physical memory on a node can be mapped onto the virtual memory of another node.

RONNIEE Express is built around the concept of non-coherent memory sharing paradigm. The memory distribution is transparent to the software and even to the processors, i.e. memory segments are logically shared, just as in a system with a centralized bus and shared memory. This capability represents the most important feature of RONNIEE Express. A memory access by a processor in the cluster is mediated to the target memory module by the hardware logic. The major advantage of this feature is that inter-node communication can be realized by simple load and store operations by the processor, without invocation of a software protocol stack. The instructions accessing remote memory can be issued at the user level, and the operating system (OS) need not be involved in the communication. This results in very low latency communications, typically in the nanosecond range. DPALIB is a memory manager and API collection that permit to manage in an extremely robust way all the communication across the shared memory segments. The main characteristic of RONNIEE express and DPALIB are:

a) The RONNIEE Express technology implements a remote shared memory approach in the data transfers between processors.

b) An application can map into its own address space a memory segment actually residing on another node.

c) Read and write operations from or to this memory segment are automatically and transparently converted by the hardware in remote operations.

d) Memory segments are dynamically managed by the fabric manager.

DPALIB provides quality of service and error management mechanism resulting in failover capabilities, error detection, hot insertion and removal of the clustered nodes. Thus with this technology it's easy to implement a robust shared memory ultra-low latency communication across clustered computing nodes, without using RDMA semantic and any cache coherency mechanism.

The libfabric [4] *dpa* provider [5] supports the A3Cube RONNIEE Express fabric using DPALIB. This paper presents some data about performance of the libfabric *dpa* provider versus native DPALIB performance. Tests were executed on a 2 node cluster where each node features an Intel® Core™ i7-4770 @3.40GHz CPU and is equipped with a RONNIEE Express NIC. As a note to avoid confusion, bandwidth results will be provided in megabytes per second, and we define 1 MB = 10^6 B.

1 RMA Bandwidth

Given the shared memory interface offered by the A3Cube's network, implementing the libfabric RMA data transfer interface requires just a tiny wrapper over the DPALIB interface.

Table 1: RMA bandwidth test

(a) libfabric *dpa*

Size (B)	Bandwidth (MB/s)
4	40.95
8	82.48
16	159.79
32	313.63
64	738.18
128	1 326.15
256	2 187.85
512	2 897.07
1 024	2 903.32
2 048	2 900.85
4 096	2 896.75
8 192	2 882.17
16 384	2 894.03
32 768	2 850.13
65 536	2 898.54

(b) DPALIB

Size (B)	Bandwidth (MB/s)
4	61.59
8	119.86
16	240.07
32	462.79
64	876.16
128	1 543.18
256	2 464.03
512	2 888.72
1 024	2 894.84
2 048	2 896.89
4 096	2 895.05
8 192	2 895.61
16 384	2 889.69
32 768	2 887.68
65 536	2 871.95

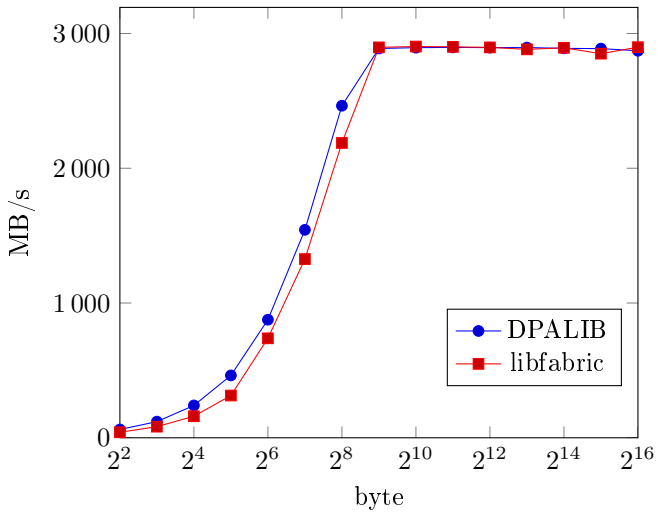
The first test is a bandwidth benchmark. For libfabric *dpa*, the results were obtained with the `fi_msg_rma` test which is part of the fabtests suite [3].

Instead, along with DPALIB A3Cube offers the `shmbench` executable, whose behavior is very close to the `fi_msg_rma` one.

In table 1 performance of the libfabric *dpa* provider is compared with DPALIB performance.

The data from table 1 is plotted in figure 1, providing graphical support to the comparison. Bandwidth figures are very similar, except for very small messages, and libfabric *dpa* tops the network bandwidth with 512 bytes messages, the same size needed for DPALIB to reach such achievement.

Figure 1: Bandwidth test



2 RMA Latency

To measure latency on RMA data transfers, A3Cube provides `shmpp`. Such test performs a remote write of data, plus it writes a counter after the data. The receiving end checks on its physical memory the counter value, and when it gets incremented performs a write of data of the same size on the original sender's memory, again plus the counter. After the write, both sides locally increment their counter to be ready for the next loop. After a number of iterations, the total loop time is divided by the number of iterations to get the round trip time for data of the size used; the round trip time is halved to get the unidirectional latency.

No similar test was found in the fabtests suite, so a new one was implemented modifying `fi_msg_rma` to reproduce the `shmpp` behavior using libfabric RMA primitives.

Table 2: RMA latency test

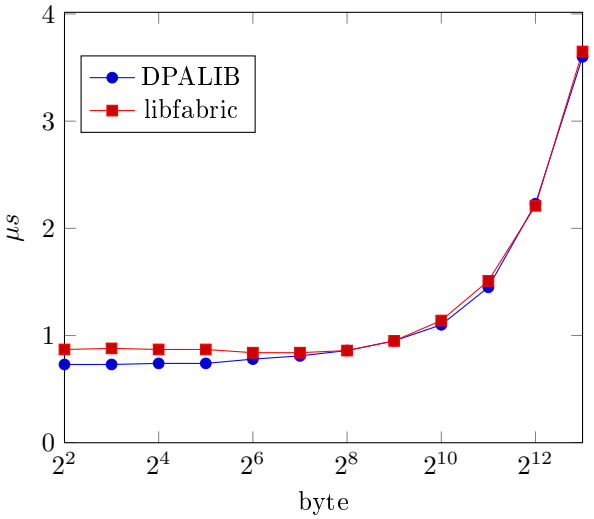
(a) libfabric *dpa*

Size (B)	Latency (μs)
4	0.87
8	0.88
16	0.87
32	0.87
64	0.84
128	0.84
256	0.86
512	0.95
1 024	1.14
2 048	1.51
4 096	2.21
8 192	3.65

(b) DPALIB

Size (B)	Latency (μs)
4	0.73
8	0.73
16	0.74
32	0.74
64	0.78
128	0.81
256	0.86
512	0.95
1 024	1.10
2 048	1.45
4 096	2.23
8 192	3.60

Figure 2: RMA latency test



Results comparing DPALIB and libfabric *dpa* are presented in table 2 and figure 2.

The benchmark shows that the greatest difference between libfabric *dpa* and DPALIB is when transferred data is very small, with a 20% latency increase (about 140 nanoseconds) using the libfabric interface to write 4 bytes remotely.

For data transfers writing more than 128 bytes, the latency difference between libfabric *dpa* and DPALIB can be measured in tens of nanoseconds or less, always

below 5%. It means that most users will not notice a performance degradation using libfabric instead of the native DPALIB interface on this network.

3 Message Queue Interface

DPALIB does not provide a native support for message passing operations. The message passing interface is implemented in the *dpa* provider using a shared memory segment as a receive buffer.

The latency of the fi_msg interface can be tested through `fi_msg_pingpong`, which is part of the fabtests suite. Results are shown in table 3 and figure 3.

Table 3: Message Queue latency test

(a) libfabric *dpa*

Size (B)	Latency (μs)
4	1.84
8	1.76
16	1.77
32	1.79
64	2.07
128	2.12
256	1.90
512	2.04
1 024	2.31
2 048	2.97
4 096	3.74
8 192	5.68

(b) DPALIB

Size (B)	Latency (μs)
4	0.73
8	0.73
16	0.74
32	0.74
64	0.78
128	0.81
256	0.86
512	0.95
1 024	1.10
2 048	1.45
4 096	2.23
8 192	3.60

Of course the message queue interface is unable to match performance of the native DPALIB interface, because it offers a different service (message-oriented data transfer maintaining message boundaries with flow control).

The `fi_msg_pingpong` code was then modified to produce a bandwidth test: to take advantage of the receive buffer, the sender wouldn't wait for a response from the receiver, but unidirectionally send as many messages as possible while the receiver limits itself to post receive buffers and read from the completion queue to ensure progress (tests were executed with `FI_PROGRESS_MANUAL`).

The results, shown in table 4 and figure 4, clearly indicate that if each message is big enough (at least 32 kB), there is no performance loss between DPALIB and the libfabric *dpa* provider. So the libfabric message queue interface is a very good choice for applications needing to stream very large amounts of data

Figure 3: Message Queue latency test

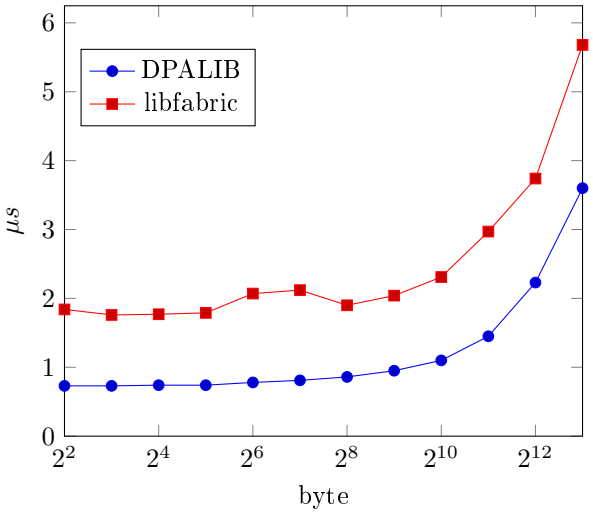


Table 4: Message Queue bandwidth test

(a) libfabric *dpa*

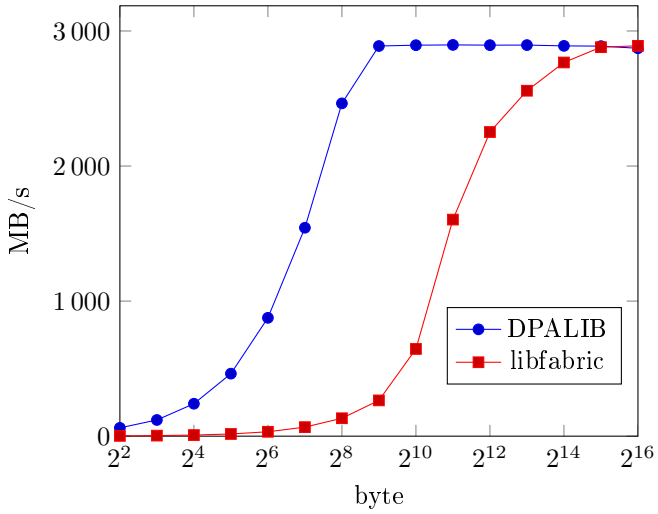
Size (B)	Bandwidth (MB/s)
4	2.03
8	4.05
16	8.11
32	16.21
64	32.43
128	66.88
256	132.60
512	264.85
1 024	645.81
2 048	1 603.51
4 096	2 251.79
8 192	2 557.60
16 384	2 766.87
32 768	2 880.10
65 536	2 890.10

(b) DPALIB

Size (B)	Bandwidth (MB/s)
4	61.59
8	119.86
16	240.07
32	462.79
64	876.16
128	1 543.18
256	2 464.03
512	2 888.72
1 024	2 894.84
2 048	2 896.89
4 096	2 895.05
8 192	2 895.61
16 384	2 889.69
32 768	2 887.68
65 536	2 871.95

between nodes on this network, since it offers the needed buffered service with virtually no overhead (unless the message is split in too small chunks, which is something an application designed for high performance would never do).

Figure 4: Message Queue bandwidth test



References

[1] A3Cube: *A3Cube website*. Available at `http://a3cube-inc.com/`.

[2] A3Cube: *Ronniee Express*. Available at `http://www.a3cube-inc.com/ronniee-express.html`.

[3] OpenFabrics Interfaces Working Group: *Fabtests*. Available at `https://github.com/ofiwg/fabtests`.

[4] OpenFabrics Interfaces Working Group: *Libfabric*. Available at `http://ofiwg.github.io/libfabric/`.

[5] Paolo Inaudi, Marco Aldinucci: *Libfabric* dpa *provider*. Available at `https://github.com/alpha-group/libfabric-provider-dpa`.