Benchmarking message queue libraries and network technologies to transport large data volume in the ALICE O^2 system

V. Chibante Barroso, U. Fuchs, A. Wegrzynek for the ALICE Collaboration

Abstract–ALICE (A Large Ion Collider Experiment) is the heavy-ion detector designed to study the physics of strongly interacting matter and the quark-gluon plasma at the CERN LHC (Large Hadron Collider).

ALICE has been successfully collecting physics data of Run 2 since spring 2015. In parallel, preparations for a major upgrade, called O2 (Online-Offline) and scheduled for the Long Shutdown 2 in 2019-2020, are being made. One of the major requirements is the capacity to transport data between so-called FLPs (First Level Processors), equipped with readout cards, and the EPNs (Event Processing Node), performing data aggregation, frame building and partial reconstruction. It is foreseen to have 268 FLPs dispatching data to 1500 EPNs with an average output of 20 Gb/s each. In overall, the O2 processing system will operate at terabits per second of throughput while handling millions of concurrent connections. To meet these requirements, the software and hardware layers of the new system need to be fully evaluated.

In order to achieve a high performance to cost ratio three networking technologies (Ethernet, InfiniBand and Omni-Path) were benchmarked on Intel and IBM platforms.

The core of the new transport layer will be based on a message queue library that supports push-pull and request-reply communication patterns and multipart messages. ZeroMQ and nanomsg are being evaluated as candidates and were tested in detail over the selected network technologies.

This paper describes the benchmark programs and setups that were used during the tests, the significance of tuned kernel parameters, the configuration of network driver and the tuning of multi-core, multi-CPU, and NUMA (Non-Uniform Memory Access) architecture. It presents, compares and comments the final results. Eventually, it indicates the most efficient network technology and message queue library pair and provides an evaluation of the needed CPU and memory resources to handle foreseen traffic.

I. Introduction

A. The ALICE Experiment

ALICE (A Large Ion Collider Experiment) [1] is the heavy-ion detector designed to study the physics of strongly interacting matter and the quark-gluon plasma at the CERN Large Hadron Collider (LHC). ALICE consists of a central barrel and a forward muon spectrometer, allowing for a comprehensive study of hadrons, electrons, muons and photons produced in the collisions of heavy ion. The ALICE collaboration also has an ambitious physics program for proton-proton and proton-ion collisions.

After a successful Run 1 ALICE has been taking data of Run 2 since the beginning of 2015. In the end of 2018 the LHC will enter a consolidation phase – Long Shutdown 2. At that time ALICE will start its upgrade to fully exploit the increase in luminosity.

The upgrade foresees a complete replacement of the current computing systems (Data Acquisition, High-Level Trigger [2] and Offline) by a single, common O^2 (Online-Offline) system.

B. The ALICE O^2 system

The ALICE O^2 computing system [3] will allow to record Pb-Pb collision at 50 kHz rate. Some detectors will be read out continuously, without physics triggers. Instead of rejecting events O^2 will compress data by online calibration and partial reconstruction.

The first part of this process will be done in dedicated FPGA cards that are supplied with raw data from detectors. The cards will perform baseline correction, zero suppression, cluster finding and inject the data into FLP's (First Level Processors) memory to create a sub-timeframe. Then, the data will be distributed over EPNs (Event Processing Node) for aggregation and additional compression.

The O^2 facility will consist of 268 FLPs and 1500 EPNs. Each FLPs will be logically connected to each EPN through a high throughput network. The O^2 farm will receive data from detectors at 28.8 Tb/s, which after processing will be reduced to 720 Gb/s.

II. Motivation

Transferring and processing Tb/s of data inside the O^2 system is a challenge for the network and computing resources. Assuming a throughput of 40 Gb/s the distance between Ethernet frames is very small – 300 ns. During that time the Linux kernel has to go through the whole TCP/IP stack and deliver the data to user space which consumes a large amount of computing resources. This work aims at estimating the CPU needs for data transport inside the O^2 system.

III. Performance Tuning

The following improvements were implemented to increase network throughput per CPU core:

V. Chibante Barroso, U. Fuchs and A. Wegrzynek are with CERN – European Organization for Nuclear Research, Genève 23, CH-1211, Switzerland

A. Wegrzynek is with Warsaw University of Technology, Faculty of Physics, Koszykowa 75, 00-662 Warsaw, Poland

978-1-5090-2014-0/16/$31.00 ©2016 IEEE

- Increased MTU (Maximum Transmission Unit) – to decrease distance between Ethernet frames.
- Increased buffers sizes – increased size of TCP and IP buffers to avoid fluctuation and packet losses.
- Enabled TSO (TCP Segmentation Offloading) [4] – offloads CPU from data segmentation; the network card chops the stream of data into number of needed segments, the process is done in the device's hardware and works only on the sender side (no support on the receving side), TSO is widely supported by Linux starting from kernel 2.6.
- NUMA tuning – in multi-CPU and multi-core era its paramount to configure CPU affinity and IRQs of the network card properly; the CPU memory should be used to avoid inter-CPU bus transfers.
- Intel DDIO [5] – allows network adapters to communicate directly with the processor's cache, reducing transfers to the main memory and therefore lowering the latency; cache sizes in modern CPUs are large enough (20MB) and can be shared among other cores. DDIO is supported by Intel Xeon E5 and E7 v2 processor families.

IV. Network Technologies

The average outgoing traffic from a single FLP is estimated to be 20 Gb/s and incoming traffic to a single EPN to be less than 10 Gb/s. Therefore, EPNs can be equipped with standard 10 Gb/s cards, but FLPs need a more effective solution. The list of chosen network candidates is the following:

- 40 Gigabit Ethernet (40 GbE) – widely used, next version of network standard also optimized for shorter distances [6].
- InfniBand (IB) FDR – 56 Gb/s; dedicated for interconnecting computers at high throughput and low latency, especially in HPC (High-performance computing) systems.
- Omni-Path (OPA) [7] – 100 Gb/s; Intel's interconnect compatible with InfiniBand, it outstands with fabric integrated into CPU; end-to-end solution is provided: switches, software stack, fabric interface.

V. Methodology

The measurements, described in this paper were performed by connecting a single FLP and EPN through a switch. This allows to test how hardware and software deal with large traffic, examine stability and performance, characterize data flow and produce input data for simulations. Further tests with more advanced architecture are ongoing – see Future Work section.

A. Test Setups

Four hardware set-ups, based on Intel and IBM platforms, were used – Table I.

Each set-up consists of two identical servers equipped with the same CPU and network card, one acting as FLP and the other as EPN.

TABLE I. TEST SETUPS

Setup name	Network	Network adapter	CPU
Intel/GbE	40 GbE	Chelsio T580	Intel E5-2690
Intel/IB	IB FDR	Mellanox MT27500	Intel E5-2690
Intel/OPA	OPA	-	Intel E5-2680v4
IBM/IB	IB FDR	Mellanox MT4115	IBM P8 2822LC

B. Configuration

By default, kernel available in current versions of Linux distributions is optimized for throughputs lower than 10 Gb/s. Therefore, to cope with the large traffic, several tweaks were deployed:

- irqbalance service, distributing interrupts over multiple CPUs, was turned off as it's more efficient to keep IRQs at the same NUMA node. Therefore, manual configuration of IRQs and CPU affinity was made.
- Network adapter and its interrupts were handled by the same CPU. High number of interrupts may kill the performance of the application running in the user space, therefore they were handled by separated, dedicated core.
- Benchmark application was pinned to chosen core of the same NUMA node. Such configuration minimizes usage of the inter-processors interconnect.

Further tuning concerns network stack buffers and parameters – see Table II.

TABLE II. NETWORK STACK BUFFERS AND PARAMETERS

Parameter name	Value
TCP recv buffer	4096 87380 16777216
TCP send buffer	4096 87380 16777216
Socket backlog	250000
Maximum Transmission Unit	9000
Transmit queue length	50000

C. Benchmarks

The benchmark simulates data transmission between FLP and EPN. On the FLP side it allocates a large fragment of memory and fills it with dummy events of a given size. Then it indefinitely iterates over these events and pushes them to the EPN. The EPN receives, unpacks data and immediately discard it. Four benchmarks were prepared based on the following libraries:

- ZeroMQ – message-based library supporting a large number of socket patters that help to create complex, distributed systems;
- nanomsg – fork of ZeroMQ with ability to plug custom transports, improved threading model and state machine [8];
- asio – asynchronous, low level I/O library;
- FairMQ – high level transport framework with own state machine and ability to work on top of lower level network library such as ZeroMQ and nanomsg;

978-1-5090-2014-0/16/$31.00 ©2016 IEEE

- O2 – development version of the O^2 framework that uses FairMQ.

All mentioned libraries require TCP/IP which natively is not supported neither by IB nor by OPA. There are several ways to provide this functionality:

- SDP (Socket Direct Protocol) – automatically converts TCP so it can run smoothly over IB. It was successfully configured on CentOS 7 and tested with iperf [10] tool. Unfortunately, SDP is not compatible with ZeroMQ.
- IPoIB – implementation of full TCP/IP stack for IB.
- IPoFabric – the same as IPoIB but applies to OPA.

It is worth to mention that there is another solution that supports all mentioned fabric technologies – libfabric framework. Libfabric is outside the scope of this paper but it is foreseen to include it in future tests.

D. Tools

To compare the test setups and network libraries, a selected number of parameters were monitored. Table II lists the tools that were used to acquire them.

TABLE II. TOOLS

Tool name	Usage
nload	Network throughput monitoring
Intel PCM [9]	Memory throughput monitoring
sysstat	CPU usage and IRQs per second
numactrl	Controls NUMA policy
ethtool	Changes network device settings
oprofile	Software profiler

A test utility that launches benchmarks on FLPs and EPNs and collects the output of the monitoring tools was developed, allowing measurements to be done almost automatically.

VI. RESULTS

The results are presented in plots representing: utilization of the CPU core that was handling benchmark application on receiving/transmitting sides and network throughput, both as a function of block size. The range of block sizes was chosen based on information provided by detector teams. The most significant value is 50 MB that corresponds to a detector contributing to 97% of the total traffic.

A. Intel/GbE

Figure 1 shows network throughput as a function of block size for 40 GbE network on Intel platform. Unanticipated behavior of nanomsg can be observed: Its throughput goes down to almost 0 for block sizes larger that 1MB (the exact value is 1048576B). It is caused by internal limitation of the library. The issues was reported to developers and left without response [11].

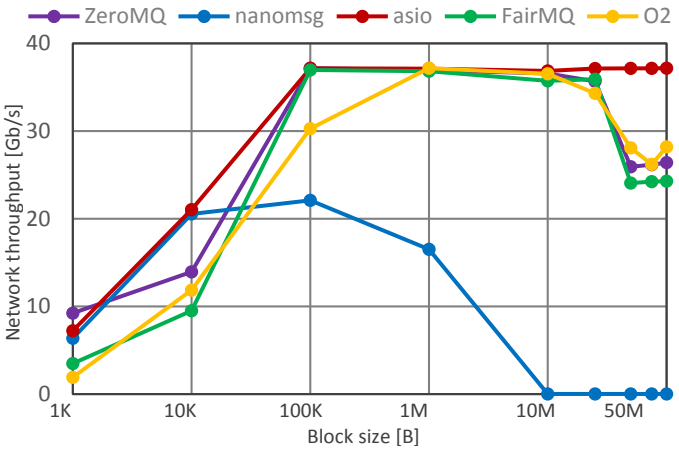

Fig. 1. Network throughput as a function of block size - 40 GbE on Intel platform.

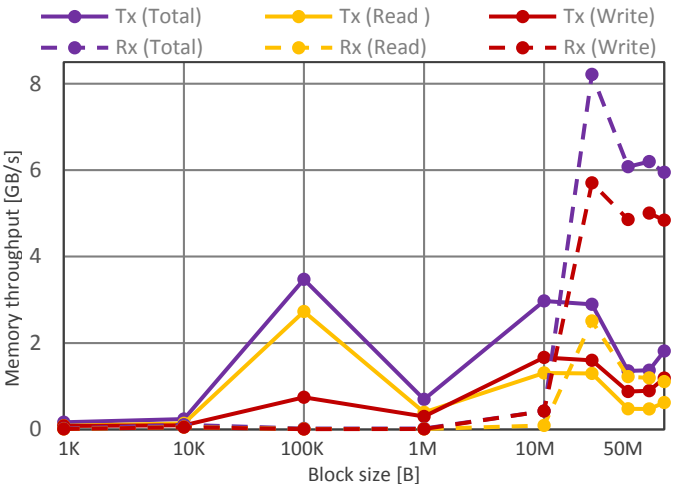

Fig. 2. Memory throughput as a function of block size for ZeroMQ benchmark – 40 GbE on Intel platform

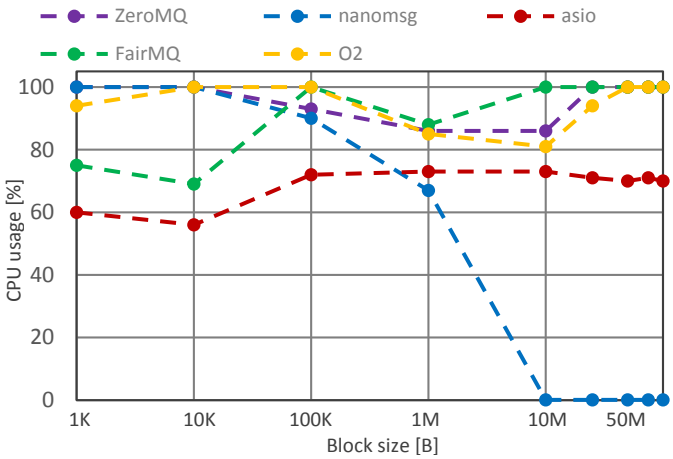

Fig. 3. CPU's core usage as a function of block size on the receiving side (EPN) – 40 GbE on Intel platform

The other unexpected behavior is the throughput decrease for block sizes larger than 25 MB for all benchmarks except asio. It is mainly caused by DDIO which is less efficient for bigger blocks. Increased traffic through the main memory on the EPN side is observed – see Figure 2 that shows memory

throughput as a function of block size for the ZeroMQ benchmark. This also causes higher CPU utilization due to data copy into main memory – see Figure 3. On the FLP side we observed corresponding drop in CPU utilization - see Figure 4.

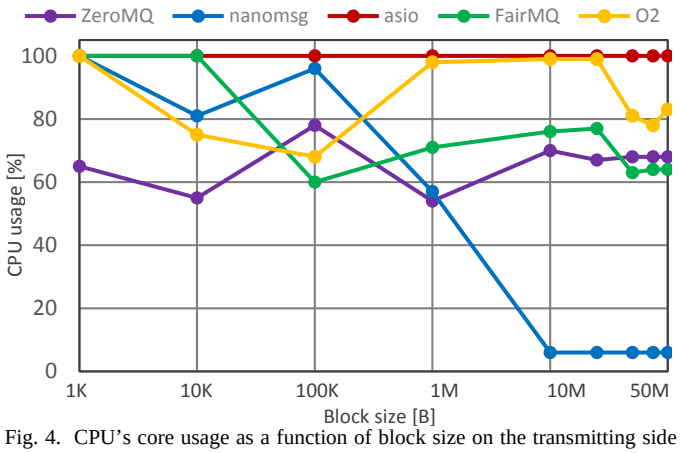

Fig. 4. CPU's core usage as a function of block size on the transmitting side (FLP) – 40 GbE on Intel platform.

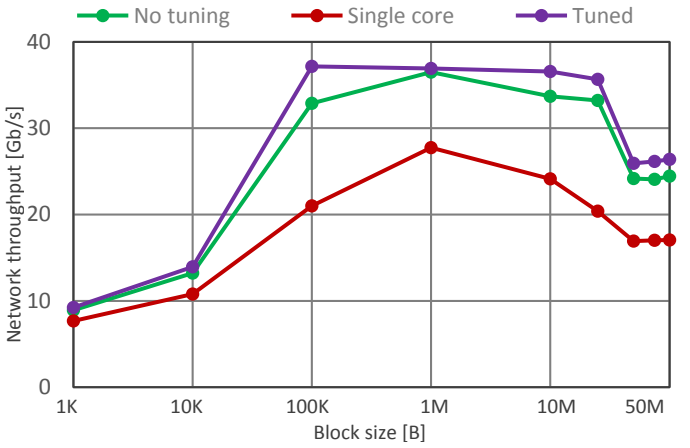

Fig. 5. No tuning and single core cases as a function of block size for ZeroMQ benchamrk – 40 GbE on Intel platform

Figure 5 presents the following additional cases that were measured only for ZeroMQ benchmark:

- No tuning – default sizes of network buffers, but IRQs handled by second core,
- Single core – in addition to "No tuning" case IRQs and the benchmark application are handled by the same CPU core,
- Tuned – original case as on Figure 1.

It can be seen that tuning buffers can slightly increase the performance. For such high transfer rates there are thousands of network's device IRQs per second. For each invoked IRQ, the kernel stops program execution and switches the context to process it, what significantly lowers the throughput.

B. Intel/IB

Figure 6 presents network throughput as a function of block size for IB with IPoIB extension on Intel platform. As for 40 GbE the same unexpected behavior for messages larger than 25 MB occurs. In addition, a large overhead due to IPoIB is observed. The measured throughput is limited to 25 Gb/s out of 56 Gb/s of available bandwidth.

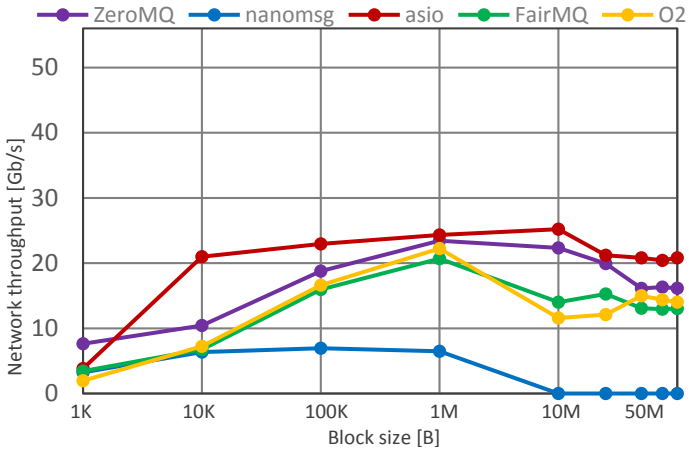

Fig. 6. Network throughput as a function of block size – IPoIB (IB FDR) on Intel platform.

C. Intel/OPA

Figure 7 presents network throughput as a function of block size for OPA fabric with IPoFabric enabled. As for other Intel based setups the issue of decreasing throughput for messages larger then 25MB can be observed. The overhead of IPoFabric is even larger than for IB (only 37.5 Gb/s out of available 100 Gb/s).

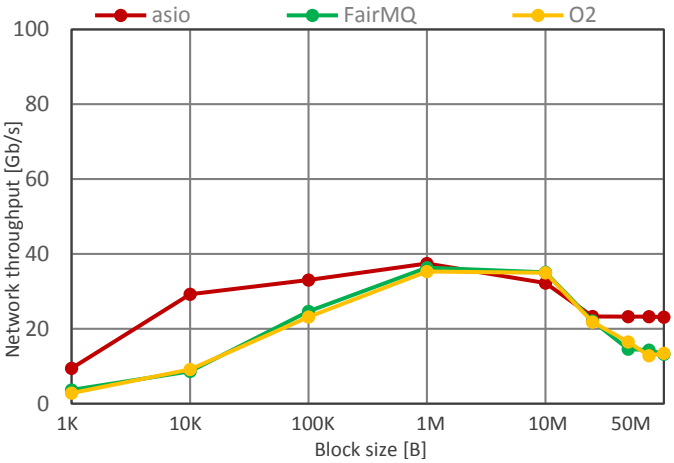

Fig. 7. Network throughput as a function of block size – IPoFabric (OPA) on Intel platform.

D. IBM/IPoIB

Figure 8 shows network throughput as a function of block size for IB network with IPoIB running on IBM platform. The throughput is constant after reaching the maximum value limited by IPoIB. The results are better comparing to the Intel based solution thanks to the ConnectX-4 network controller of the newest Mellanox card.

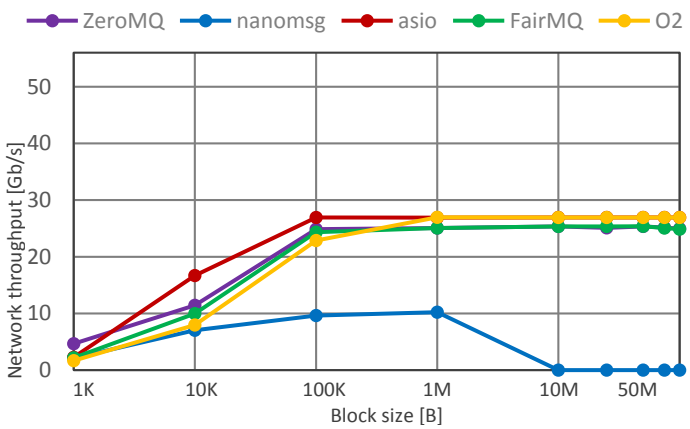

Fig. 8. Network throughput as a function of block size – IPoIB (IB FDR) on IBM platform.

VII. Conclusion

Ethernet with its long-serving TCP/IP stack reached its maximum speed. In this case single core of the modern CPU is enough to transport needed traffic from FLP to EPN.

The solutions not supporting TCP/IP natively such as IB and OPA have the large overhead. IPoIB and IPoFabric use less than half of the available bandwidth. IPoIB and IPoFabric running on Intel platform require two cores of the CPU and IPoIB on IBM platform singe core to transport given traffic.

The accelerating mechanisms such as TSO and DDIO improves network throughput while running at high CPU load.

Tuning network buffers and parameters, setting up IRQ and NUMA correctly allows to utilize the potential performance from the hardware.

VIII. Future Work

The results presented in this paper refer to single sender and receiver architecture. The final setup is more complex, therefore further test will be performed starting from a single FLP dispatching data to multiple EPNs and finishing with N FLPs to M EPNs. This will allow to evaluate how the system behaves in terms of load balancing and scalability when handling thousands of concurrent connections.

Another point of improvement is the utilization of the bandwidth of IB and OPA. Using alternative solution to IPoIB and IPoFabric may increase the throughput. One example that will be tested is libfabric that can run over IB, OPA and Ethernet.

References

[1] ALICE Collaboration, “The ALICE experiment at the CERN LHC”, 2008 JINST 3 S08002, 2008.
[2] ALICE Collaboration, “ALICE technical design report of the trigger, data acquisition, high level trigger, and control system”, CERNLHCC-2003-062, 2004.
[3] ALICE Collaboration, “Technical Design Report for the Upgrade of the Online–Offline Computing System”, CERN-LHCC-2015-006, 2015.
[4] Chelsio Communications, “Chelsio T5/T4 Unified Wire for Linux”, 2015.
[5] Intel Corporation, “Intel Data Direct I/O Technology Overview”, 2012, unpublished.
[6] John D’Ambrosia, David Law, Mark Nowell, “40 Gigabit Ethernet and 100 Gigabit Ethernet Technologu Overview”, Ethernet Alliance, 2010.
[7] Mark S. Birrittella et al, “Intel Omni-Path Architecture”, IEEE 2015.
[8] Haikel Guemar, “Nanomsg: ZeroMQ done right”, 2006.
[9] Intel Performance Counter Monitor, https://software.intel.com/en-us/articles/intel-performance-counter-monitor, accessed on May 2014.
[10] iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool, https://github.com/esnet/iperf, accessed on May 2016.
[11] Nanomsg GitHub issues, https://github.com/nanomsg/nanomsg/issues, accessed on May 2016.