Using ALFA for high throughput, distributed data transmission in the ALICE O2 system

To cite this article: A Wegrynek and ALICE Collaboration 2017 J. Phys.: Conf. Ser. 898 032026

View the article online for updates and enhancements.

Related content
- Evaluating the transport layer of the ALFA framework for the Intel® Xeon Phi™ Coprocessor
  Aram Santogidis, Andreas Hirstius and Spyros Lalis
- Decoding the quark-gluon plasma with jet quenching observables
  Konrad Tywoniuk
- Analysis of the Quark-Gluon Plasma by Heavy Quarks
  J Aichelin, V Ozvenchuck, T Gousset et al.
Using ALFA for high throughput, distributed data transmission in the ALICE O² system

A Wegrzynek¹ for the ALICE Collaboration
¹CERN, Geneva, Switzerland
E-mail: Adam.Wegrzynek@cern.ch

Abstract. ALICE (A Large Ion Collider Experiment) is a heavy-ion detector designed to study the physics of strongly interacting matter (the Quark–Gluon Plasma) at the CERN LHC (Large Hadron Collider). ALICE has been successfully collecting physics data in Run 2 since spring 2015. In parallel, preparations for a major upgrade of the computing system, called O² (Online-Offline), scheduled for the Long Shutdown 2 in 2019-2020, are being made. One of the major requirements of the system 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. The ALFA framework will standardize and handle software related tasks such as readout, data transport, frame building, calibration, online reconstruction and more in the upgraded computing system. ALFA supports two data transport libraries: ZeroMQ and nanomsg. This paper discusses the efficiency of ALFA in terms of high throughput data transport. The tests were performed with multiple FLPs pushing data to multiple EPNs. The transfer was done using push-pull communication patterns and two socket configurations: bind, connect. The set of benchmarks was prepared to get the most performant results on each hardware setup. The paper presents the measurement process and final results – data throughput combined with computing resources usage as a function of block size. The high number of nodes and connections in the final set up may cause race conditions that can lead to uneven load balancing and poor scalability. The performed tests allow us to validate whether the traffic is distributed evenly over all receivers. It also measures the behaviour of the network in saturation and evaluates scalability from a 1-to-1 to a N-to-M solution.

1. Introduction

1.1. The ALICE Experiment
ALICE (A Large Ion Collider Experiment) [1] is a heavy-ion detector designed to study the physics of strongly interacting matter (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 in Run 2 since the beginning of 2015. In the end of 2018 the LHC will enter into 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 and Offline) by a single, common O² (Online-Offline) system.
1.2. The ALICE O² System

The ALICE O² computing system [2] will allow recording of Pb-Pb collisions at 50 kHz interaction rate. Some detectors will be read out continuously, without physics triggers. Instead of rejecting events O² will compress the data by online calibration and partial reconstruction.

The first part of this process will be done in dedicated FPGA cards that receive the raw data from the detectors. The cards will perform baseline correction, zero suppression, cluster finding and inject the data into FLPs (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 O² facility will consist of 268 FLPs and 1500 EPNs. Each FLP will be logically connected to each EPN through a high throughput links. The O² farm will receive data from detectors at 27.2 Tb/s, which after processing will be reduced to 720 Gb/s.

2. Motivation

Transferring and processing Tb/s of data inside the O² system is a challenge for the network and computing resources. Assuming a throughput of 40 Gb/s, the interval between Ethernet frames is just about 300 ns. During that time the Linux kernel has to go through the whole TCP/IP stack and deliver the data to user space that consumes a large amount of computing resources. In addition the high number of TCP streams can have negative influence on the scalability of the system. This work aims at benchmarking different network libraries and technologies to quantify how many computing resources should be reserved for data transport.

3. Performance tuning

The following improvements were implemented to increase the network throughput:

- Increased MTU (Maximum Transmission Unit).
- Increased buffers sizes – increased size of TCP and IP buffers to avoid fluctuation and packet losses.
- Enabled TSO (TCP Segmentation Offloading) [3] – offloads the CPU from data segmentation. The network card chops the stream of data into a number of needed segments. The process is done in the device's hardware and works only on the sender side (no support on the receiving side). The TSO is widely supported by Linux starting from kernel 2.6.
- NUMA (Non-uniform memory access) tuning – in a multi-CPU and multi-core era it’s paramount to configure the CPU affinity and IRQs (Interrupt Requests) of the network card properly. The local memory should be used to avoid inter-CPU bus transfers.
- Intel DDIO (Data Direct I/O) [4] – 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 several cores. DDIO is supported by Intel Xeon E5 and E7 v2 processor families.

4. Network technologies

The average outgoing traffic from a single FLP is estimated to be 20 Gb/s and the 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 performant solution. The list of possible network candidates is the following:

- 40 Gigabit Ethernet (40 GbE) – widely used, next version of network standard also optimized for shorter distances [5].
- InfiniBand (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) [6] – 100 Gb/s interconnect compatible with InfiniBand API. OPA features fabric integrated into CPU and full support of RDMA (Remote Direct Memory Access).
5. Methodology

The measurements, described in this paper, were performed first by connecting a single FLP and EPN through a switch. This allowed us 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 were done using multiple physical FLP and/or EPN machines. In addition, at each node multiple FLP/EPN devices (processes) were started to simulate a higher number of simultaneous data streams.

5.1. Test setups

Three hardware set-ups, based on Intel platform, were used, see Table 1. Each set-up consists of two identical nodes equipped with the same CPU and network card, one acting as FLP and the other one as EPN.

Table 1. Test setups.

| Network            | TCP support | Network adapter | CPU     |
|--------------------|-------------|-----------------|---------|
| 40 GbE             | (native)    | Chelsio T580    | Intel E5-2690 |
| IB FDR (56 Gb/s)   | IPoIB       | Mellanox MT27500| Intel E5-2690 |
| OPA (100 Gb/s)     | IPoFabric   | -               | Intel E5-2680v4 |

5.2. Configuration

By default, the 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, which distributes interrupts over multiple CPUs, was turned off as it’s more efficient to keep IRQs at the same NUMA node. Therefore, manual configuration of the IRQs and CPU affinity was made.
- Benchmark application, the network adapter and its interrupts were handled by the same NUMA node. Such a configuration minimizes the usage of the inter-processors interconnect.
- A high number of interrupts may kill the performance of the application running in the user space. Therefore the benchmark was pinned to a separate, dedicated core.

Further tuning of the network stack buffers and parameters was performed, see Table 2.

Table 2. Network buffers and parameter settings

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

5.3. Benchmark

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

- ZeroMQ 4.0.6 – message-based library supporting a large number of socket patterns that help to create complex, distributed systems;
• nanomsg 1.0 – rewrite of ZeroMQ with the ability to plug custom transports with an improved threading model and state machine;
• asio 1.61 – asynchronous, low level I/O library;
• FairMQ [7] – high level transport framework with an internal state machine and ability to work on top of a lower level network library such as ZeroMQ and nanomsg;
• O2 – software package based on the development version of the O² framework that uses FairMQ;
• FDT (Fast Data Transfer) – measures the bandwidth of the network, uses all available CPU cores and multiple TCP streams. It’s treated as reference benchmark.

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 the iperf [8] tool. Unfortunately, SDP turned out to be incompatible with ZeroMQ.
• IPoIB – implementation of the full TCP/IP stack for IB.
• IPoFabric – the same as IPoIB but applies to OPA.

5.4. Tools
To compare the test setups and network libraries, a selected number of parameters were monitored. Table 3 lists the tools that were used to acquire these. 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.

Table 3. Monitoring tools.

| Tool name | Usage                      |
|-----------|----------------------------|
| nload     | Network throughput monitoring |
| Intel PCM | Memory throughput monitoring |
| systat    | CPU usage and IRQs per second |
| numactrl  | NUMA policy                |
| ps        | Memory usage               |

6. Results
The results are gathered into plots representing: CPU, memory usage and network throughput as a function of block size. The range of block sizes was chosen based on information provided by the detector experts. The most significant value is 50 MB that corresponds to a detector contributing to 97% of the total ALICE traffic.

6.1. Single receiver, single sender
All listed benchmarks where tested on all platforms using a single FLP, single EPN architecture. The measurements allowed us to evaluate the performance of each pair of network library and technology, and chose the most efficient combination for further tests. The FDT benchmark indicates maximum throughput that can be reached via a given network.

6.1.1. Ethernet
Figure 1 shows the network throughput as a function of the block size for the 40 GbE network. Version 1.0 of nanomsg provides low but stable results for all block sizes (which was not the case in the version 0.8 [9]). The unexpected behaviour of the benchmarks except asio can be observed – the throughput
decrease for block sizes larger than 25 MB. This is partly caused by the DDIO which is less efficient for bigger blocks. Increased traffic through the main memory on the EPN side is observed. This also caused higher CPU utilization due to the copying the data to the memory which translates to lowered throughput.

6.1.2. IPoIB

Figure 2 presents the network throughput as a function of the block size for IPoIB. As for 40 GbE, the nanomsg turned out to be the slowest. 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.

6.1.3. IPoFabric

Figure 3 presents the network throughput as a function of the block size for the IPoFabric. As for other Intel based setups, a decreased 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).
6.2. Multiple receivers or multiple senders

The measurements performed with single sender and single receiver architecture allowed us to select the best network technology and library. Since the benchmarks require TCP/IP, Ethernet, which natively encapsulates this protocol stack, gave the best results. Among the MQ libraries ZeroMQ turned out to be more performant reaching significantly higher throughput than nanomsg. Therefore, Ethernet and ZeroMQ (with O^2 framework on the top of it) were tested in two following, more complex architectures (see Figures 4 and 5).

**Figure 4.** Saturating receiver. **Figure 5.** Saturating sender.

**Figure 6.** Network throughput, CPU and memory usage as a function of block size.
6.2.1. Saturating receiver
Figure 6 shows the network throughput as a function of block size between 3x80 FLP devices (240 FLP devices distributed equally among 3 servers) and a single EPN device (see architecture on Figure 4) measured on the EPN side for two socket configurations: bind (B), connect (C). The tests were performed using the O² benchmark (with ZeroMQ) and 10/40GbE. The CPU and network throughput results are quite similar for both socket configurations. On the other side ZeroMQ utilized buffers on the EPN side in ‘connect’ socket configuration only, therefore the difference in memory usage was observed.

Moreover, Figure 7 shows measurements for two different buffer sizes: (1) one data block and (5) up to 5 data blocks can be buffered. The presented plot applies to bind socket configuration only. As expected the latter buffer configuration uses more memory but provides slightly better results.

![Figure 7. Network throughput, CPU and memory usage as a function of block size.](image)

6.2.2. Saturating sender
Figure 8 presents the network throughput as a function of block size between a single FLP device and 4x60 EPN devices (240 EPN devices distributed equally among 4 servers; see architecture in Figure 5) measured on the FLP side for two socket configurations: bind (B), connect (C). Tests were performed using O² benchmark (with ZeroMQ) and 40GbE.

![Figure 8. Network throughput and CPU usage as a function of block size.](image)
7. Conclusions
The key result of the performed tests indicate the ALFA framework serves its purpose and can be successfully adopted for the O² needs.

Ethernet, which with its long-serving TCP/IP stack reached maximum speed is a good candidate for the O². The other networking solutions need to be utilized in a more efficient way (eg. using libfabric [10]) as both IPoIB and IPoFabric barely used half of the available bandwidth.

The CPU computing power is sufficient for data transfer – two cores (one physical) are enough to receive or transmit data blocks of 10KB-100MB from a single FLP to multiple EPNs or from multiple FLPs to a single EPN.

The following observations were drawn regarding the ZeroMQ and nanomsg–based benchmarks:

- Software buffering in ZeroMQ happened on the connect side (in general ZeroMQ allows the data to be buffered on both sides).
- The ZeroMQ sockets in “connect” configuration block unless asynchronous version is called explicitly.
- The nanomsg performance is not as good as the other libraries but its modular architecture allows the usage of the custom transports.

8. Future work
It is foreseen to repeat these tests when the almost final set-up will be available in the O² Development Lab to verify whether the system scales properly and to keep track of the O² software framework overhead.

References
[1] The ALICE Collaboration 2008 The ALICE experiment at the CERN LHC”, JINST 3 S08002
[2] The ALICE Collaboration 2015 Technical Design Report for the Upgrade of the Online–Offline Computing System CERN-LHCC-2015-006
[3] Chelsio Communications 2015 Chelsio T5/T4 Unified Wire for Linux
[4] Intel Corporation 2012 Intel Data Direct I/O Technology Overview, http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/direct-i-o-technology-overview-paper.pdf
[5] J. D’Ambrosia et al. 2010 40 Gigabit Ethernet and 100 Gigabit Ethernet Technology Overview, http://www.ethernetalliance.org/wp-content/uploads/2011/10/document_files_40G_100G_Tech_overview.pdf
[6] M. Birrittella et al. 2015 Intel Omni-Path Architecture Technology Overview, http://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-architecture-fabric-overview.html
[7] FairMQ, https://github.com/FairRootGroup/FairRoot/tree/master/fairmq
[8] iperf, https://iperf.fr/
[9] A. Wegrzynek et al. 2016 Benchmarking message queue libraries and network technologies to transport large data volume in the ALICE O² system IEEE-NPSS Real Time Conference
[10] Libfabric Programmer’s Manual, https://ofiwg.github.io/libfabric/, Accessed on 11/3/2016