Report Optical Interconnect Energy Efficiency – Intermediate release

Co-funded by the Horizon 2020
Framework Programme of the European Union

DELIVERABLE NUMBER	D4.7
DELIVERABLE TITLE	Report Optical Interconnect Energy Efficiency – Intermediate release
RESPONSIBLE AUTHOR	J.Nider (IBM)

GRANT AGREEMENT N.	688386
PROJECT REF. NO	H2020- 688386
PROJECT ACRONYM	OPERA
PROJECT FULL NAME	LOw Power Heterogeneous Architecture for Next Generation of SmaRt Infrastructure and Platform in Industrial and Societal Applications
STARTING DATE (DUR.)	01/12/2015
ENDING DATE	30/11/2018
PROJECT WEBSITE	www.operaproject.eu

WORKPACKAGE N. | TITLE	WP4 | Energy efficiency on Heterogeneous Architecture
WORKPACKAGE LEADER	Certios
DELIVERABLE N. | TITLE	D4.7 | Optical Interconnect Energy Efficiency - Intermediate
RESPONSIBLE AUTHOR	J.Nider (IBM)
DATE OF DELIVERY (CONTRACTUAL)	30/11/2017 (M24)
DATE OF DELIVERY (SUBMITTED)	30/11/2017 (M24)
VERSION | STATUS	V1.0
NATURE	R(Report)
DISSEMINATION LEVEL	PU(Public)
AUTHORS (PARTNER)	J.Nider (IBM), M.Rapoport (IBM)

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	first draft	09/11/2017	J.Nider (IBM)
0.2	Integrated comments from other partners	19/11/2017	J.Nider (IBM)
0.9	Ready for submission	21/11/2017	J.Nider (IBM)
1.0	Final review	30/11/2017	Giulio Urlini (STM)

PARTICIPANTS		CONTACT
STMICROELECTRONICS SRL		Giulio Urlini Email: Giulio.urlini@st.com
IBM ISRAEL SCIENCE AND TECHNOLOGY LTD		Joel Nider Email: joeln@il.ibm.com
HEWLETT PACKARD CENTRE DE COMPETENCES (FRANCE)		Cristian Gruia Email: cristian.gruia@hpe.com
NALLATECH LTD		Craig Petrie Email: c.petrie@molex.com
ISTITUTO SUPERIORE MARIO BOELLA		Olivier Terzo Email: terzo@ismb.it
TECHNION ISRAEL INSTITUTE OF TECHNOLOGY		Dan Tsafrir Email: dan@cs.technion.ac.il
CSI PIEMONTE		Vittorio Vallero Email: vittorio.vallero@csi.it
NEAVIA TECHNOLOGIES		Stéphane Gervais Email: s.gervais@lacroix.fr
CERIOS GREEN BV		Frank Verhagen Email: frank.verhagen@certios.nl
TESEO SPA		Stefano Serra Email: stefano.serra@eiffage.com
DEPARTEMENT DE L'ISERE		Olivier Latouille Email: olivier.latouille@isere.fr

ACRONYMS LIST

Acronym	Description
CAPI	Coherent Attached Peripheral Interconnect
DMA	Direct Memory Access
FPGA	Field Programmable Gate Array
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
NIC	Network Interface Card
OS	Operating System
PCIe	Peripheral Component Interconnect Express
QSFP	Quad Small Form-factor Pluggable
RDMA	Remote Direct Memory Access
SFP+	Small Form-factor Pluggable – enhanced
TCP	Transmission Control Protocol
VM	Virtual Machine

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Position of the deliverable in the whole project context

This deliverable (D4.7) reports on findings that are based on the design specified in D6.5 of the low-power, high-speed, low-latency interconnect used for container migration. The results from these measurements will be used to further influence the design in WP6, and will be reflected in D6.9 which is the final version of D6.5 (Integration of FPGA and Low Power Server in a Small Form Factor Data Center).

The measurements in this deliverable will be updated and expanded in the follow-up deliverable D4.4, which is the final version of D4.7 (this deliverable). D4.4 will also compare measurements of our solution as compared to the baseline measurements of current solutions as described in D4.7.

Description of the deliverable

This deliverable reports on the energy efficiency of the optical interconnect used to transfer memory pages within the heterogeneous cluster. The goal is to measure the energy consumption of remote page faults using current technology, and use that knowledge to influence the design of the protocol and NIC that we are developing beyond the state of the art. In the next version of the deliverable, we will measure the energy efficiency of our design & implementation, and compare it to the current state of the art.

TABLE OF CONTENTS

1 INTRODUCTION

One focus of OPERA is on heterogeneous data centers, which most readers automatically associate with the CPU. However, another important component of the heterogeneous data center is the network which connects the various machines together. This deliverable reports on our progress and plans for measuring what we can do with the network to help save power consumption. Our approach is to use the interconnect to enable execution context migration, that is the migration of containers, processes or VMs between various compute nodes in the data center. Context migration has been shown to be an effective method of rebalancing workload in a data center, but to be effective it is important to know the cost of the migration to be able to make an educated decision if and when to perform a migration.

In D6.5 we outline our new low-latency protocol that can enable remote page faults between machines that goes beyond the state of the art. To truly build efficient data centers that go beyond what is currently possible in terms of physical space, computational capabilities and power consumption, we must develop our equipment so that all selected components are interoperable and there is harmony between them. As of today, competition between multiple vendors at many levels is making interoperability a huge challenge. A very important part of our work in OPERA is to help overcome these challenges and show how equipment from multiple vendors can, and should, work together. In this deliverable, we show the measurements and plans for future measurements that support the claim, and how to measure the level of success.

As explained in D6.5, our original plan was to have Nallatech work on the FPGA implementation of our page fault protocol, which relies on the CAPI protocol on the host side. Due to unforeseeable circumstances, we were not able to execute that plan. Instead, we requested the help of an IBM product team who was responsible for the implementation of the CAPI protocol in firmware. Again our plans have been forced to change since that team has been reassigned, and is no longer able to help us with the prototype implementation. While we continue to look for additional help to implement the FPGA version of our protocol, we also outline our backup plan which does not suffer from external dependencies and can be used to give useful results, although not fully compliant with our original goals.

Despite our setbacks, we show good progress towards understanding the capabilities and shortcomings of today's interconnects in the data center, and are positioned to be able to make recommendations as to how these interconnects should operate in the future. These recommendations come with real measurements on real workloads as taken from the OPERA use cases, with experiments and results that can be reproduced so that others may build on this knowledge in the future.

2 POST-COPY MIGRATION

As discussed in the deliverables in WP5, context migration is an important tool when balancing workloads in a data center. Based on previous work, we know that post-copy migration is a powerful technique that can reduce downtime and overall network traffic, thus allowing us to rebalance often to get maximum efficiency in the face of a changing workload.

The main drawback of post-copy migration is the time it takes to copy a memory page on-demand from the remote server. This is of paramount importance, since the executing thread is blocked until the memory page can be retrieved from the remote server. Our goal is to serve the page fault as quickly as possible, so the execution can continue unhindered. We are developing a remote page fault protocol that is detailed in D6.5 to minimize the latency taken during a remote page fault. Since there is a chance that our prototype implementation will have further delays, we have instead identified other methods of implementing a similar protocol based on existing technology.

We identified the RoCE (RDMA over Converged Ethernet) protocol as having basic message types that are very similar to our protocol. By using the RoCE protocol in conjunction with the CAPI protocol, we can approximate our original design with off-the-shelf hardware. This is our plan in the case that the FPGA implementation of our protocol will not be completed on time.

3 SAVING POWER

3.1 CONNECTION TO HOST

The purpose of a NIC (network interface card) is to package information generated by the host computer and send it over a long-distance link to another host which can then decode and read the information. The NIC is connected to the host over a local bus which must be capable of reading and writing at high speeds. Generally today NICs are connected over PCIe (PCI express [1]), which is true for most server platforms, and even some mainframes. In some examples from embedded computing, hosts may have a network core that is connected over another kind of local bus (such as AXI) which does not have the same capabilities as PCIe, but may provide other benefits such as a simpler (yet less flexible) design. We were able to find only one example of a server-class machine - the Oracle Sonoma [2] - that has an Infiniband controller tightly integrated with the CPU local bus, and therefore does not use PCIe for the interconnect. While we were not able to measure its performance, we believe this kind of design would give the best power consumption and lowest latency for an interconnect, at the cost of not being able to upgrade the NIC. Since there are fewer components on the datapath, there is a more direct connection between CPU and the external connector, which means the data does not have to be converted as many times, saving the hardware.

In the server market, the focus of these interconnects is generally on performance, and not on power savings. In the datasheets we analyzed, there is a clear trend towards higher power consumption with later versions of the PCIe protocol. This is because performance (i.e. throughput) is the highest design priority for this kind of product, and power consumption is secondary. If hardware manufacturers were to put power consumption as a design priority, it is possible they would have some options available to reduce power. Technology from ARM servers for example, was first introduced on embedded ARM processors which ran on batteries where power consumption was very important. They use techniques such as clock gating to save power by turning off clock signals when the hardware block is not in use. These features became standard on embedded ARM platforms, and are being carried over into server technology.

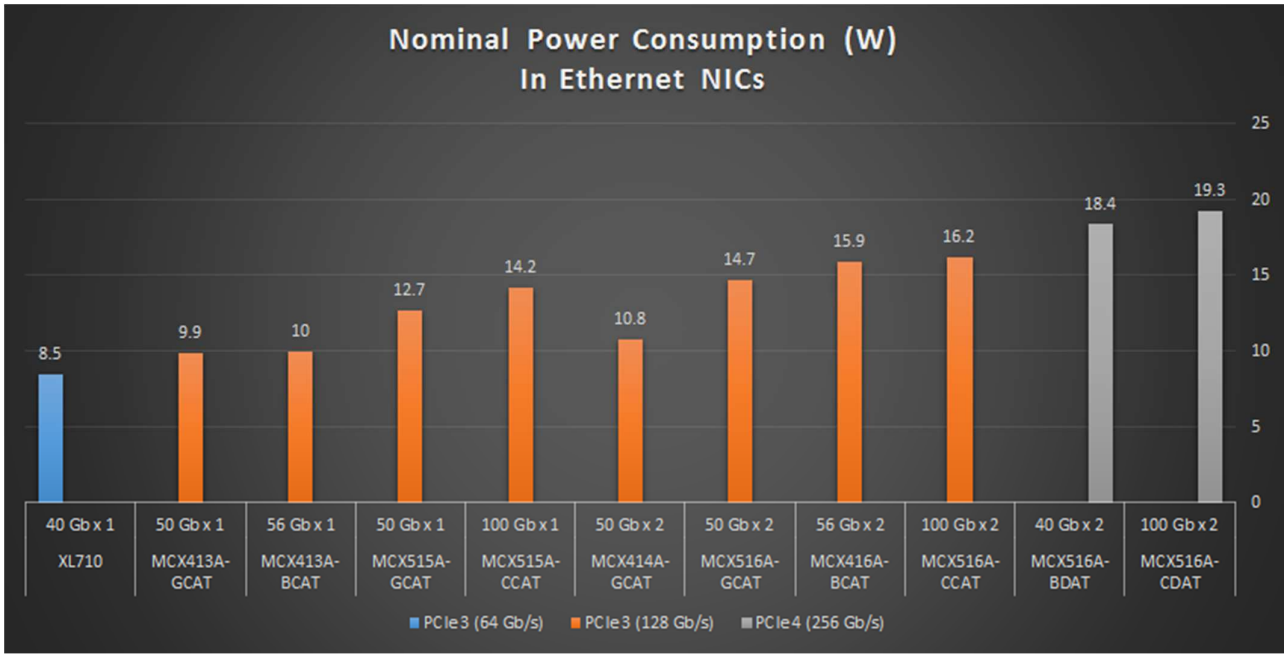

Table 1: Published power ratings of modern NICs from various vendors

When using off-the-shelf hardware, we are limited by the features in that hardware. Our main strategy to save power on the host bus is by carefully reviewing the transfer algorithms to remove any unnecessary data transfer, and reducing the amount of work done per byte transferred for the remaining data.

In Table 1, we can notice the trend of increasing power consumption in NICs. These data were taken from the datasheets of the respective cards, as published by the manufacturers [3] [4] [5]. Some manufacturers (such as Emulex and Broadcom) have chosen to not publish their power measurements in the datasheet. Therefore, we include only Intel and Mellanox cards in the chart. Intuitively, there are two reasons for the increase in power consumption; increase in bandwidth at the host bus (i.e. PCIe) and increase in bandwidth at the interconnect (i.e. Ethernet). By comparing similar cards from the same manufacturer that differ only in PCIe bandwidth (such as Mellanox MCX516A-BDAT and Mellanox MCX516A-CDAT), we can see the effects are significant (increase from 16.2W to 19.3W).

Interestingly, (and perhaps counter-intuitively) if we compare cards that differ only in the number of ports (MCX515A-GCAT and MCX516A-GCAT), the increase is much less than would be expected (12.7W vs 14.7W). In fact, even cards that have the same PCIe host bus bandwidth and same external bandwidth (MCX414A-GCAT and MCX516A-GCAT), and differ only in the generation (ConnectX-4 vs ConnectX-5) seem to have a large jump in power consumption (10.8W vs 14.7W). This increase is likely due to more advanced features in the card such as offloading, filtering and preprocessing engines, which would show a comparable decrease in the CPU consumption of the host which no longer has to perform these features in software. For this reason, the apparent increase in power consumption when moving to a later card generation is somewhat misleading, as the power consumed by the other components in the system is likely reduced.

3.2 OPTICAL INTERCONNECTS

On the surface, it may seem obvious that optical interconnects save power because light transmissions using a laser are more efficient than sending electrically encoded bits over a wire, however this is only partially true. The NIC logic (and in fact, the rest of the computer) works electrically, so the electrical signals still need to be converted into light, and back again on the other end of the link. This is performed through a component called a transceiver which uses a standard interface such as a SFP+ or QSFP28, depending on the application. This conversion still takes power, so in a data center with short links (i.e. inside the rack) there is not much savings over a pure electrical link since the vast majority of the power is used during the computation and transmission logic. The real gains come from the speed of transmission, and distance at which transmissions can occur without drastically increasing power consumption (QSFP28 can generate signals for communication over 10 Km). Since this is already well-known in the industry, practically all high speed data center networks today work over optical links (Ethernet, Infiniband, Fibre Channel).

4 MEASUREMENTS

When a container (or VM) is migrated, only the bare minimum of the context is copied before the process is resumed. This is according to the post-copy method that has been explained previously. After the process is resumed on the target machine, it is missing most of its memory, which needs to be fetched from the source machine before it can be used. The memory is fetched on-demand, which means the latency is of utmost importance, since the application cannot continue until the page is available in local memory. Therefore our goal is to use networking equipment that can minimize both latency and power consumption, at the cost of throughput. Since we have a fixed size transmission (one memory page is generally 4KB), throughput is not as important as latency.

To design our network equipment, we must have a realistic target for migrating the memory page. To help set that target, we measure several aspects to get an overall picture:

1. Latency of the operating system to map an empty page on a local machine
2. Latency of various competing technologies used for remote page faults
3. Statistics of VM migration using QEMU (KVM hypervisor)
4. Measurements of remote page fault time using CRIU

4.1 MEMORY TRANSFER LATENCY

In order for us to determine the lower bound for the transfer operation to complete, we want to measure the time Linux takes to handle a non-swapped, not-present page fault. That means that the operating system recognizes that the accessed address is valid, but that there is currently no physical memory page mapped at that virtual address. When an application attempts to access such memory, the correct behaviour for the operating system is to allocate a new physical memory page, map it to the correct virtual address in the page tables, and resume the process to retry the memory access.

4.1.1 Methodology

We wrote a program to cause a page fault by accessing (writing) a value to memory at an address that has been mapped using mmap() but does not have a physical backing page. When the page is written to, it causes a not-present fault in the processor which notifies Linux. The page fault handler sees that this is a valid virtual memory address for the faulting process, and allocates a new physical memory page for that address. It then maps the physical page into the page table, and allows the process to retry the access. This procedure is timed by using the CPU's internal cycle counter by reading the counter before and after the access. The following is a high-level description of the algorithm:

1. Use mmap() to allocate a range of virtual memory for a process
2. Read the cycle counter
3. Write to a memory address in the first page
4. Read the cycle counter - calculate the difference
5. Increment address to next memory page, and repeat

4.1.2 Preliminary Results

This measurement was performed on three machines - an Intel(R) Xeon(R) CPU X5570 @ 2.93GHz (released 2009), POWER8 8247-22L @ 2.06GHz (released 2014) and APM XGENE X-C1 @ 1.6 GHz (released 2014). With the POWER and APM machines being 5 years newer, we expect the Intel to give the higher number (more latency) providing an upper-bound, and the POWER to give the lower latency (lower-bound). All of the measurements were taken using Ubuntu 16.04 with Linux kernel version 4.12. The measurements were taken 5000 times on each system (5000 page faults), and statistics were

collected in CDF (continuous distribution frequency) graphs to show the distribution of the response times.

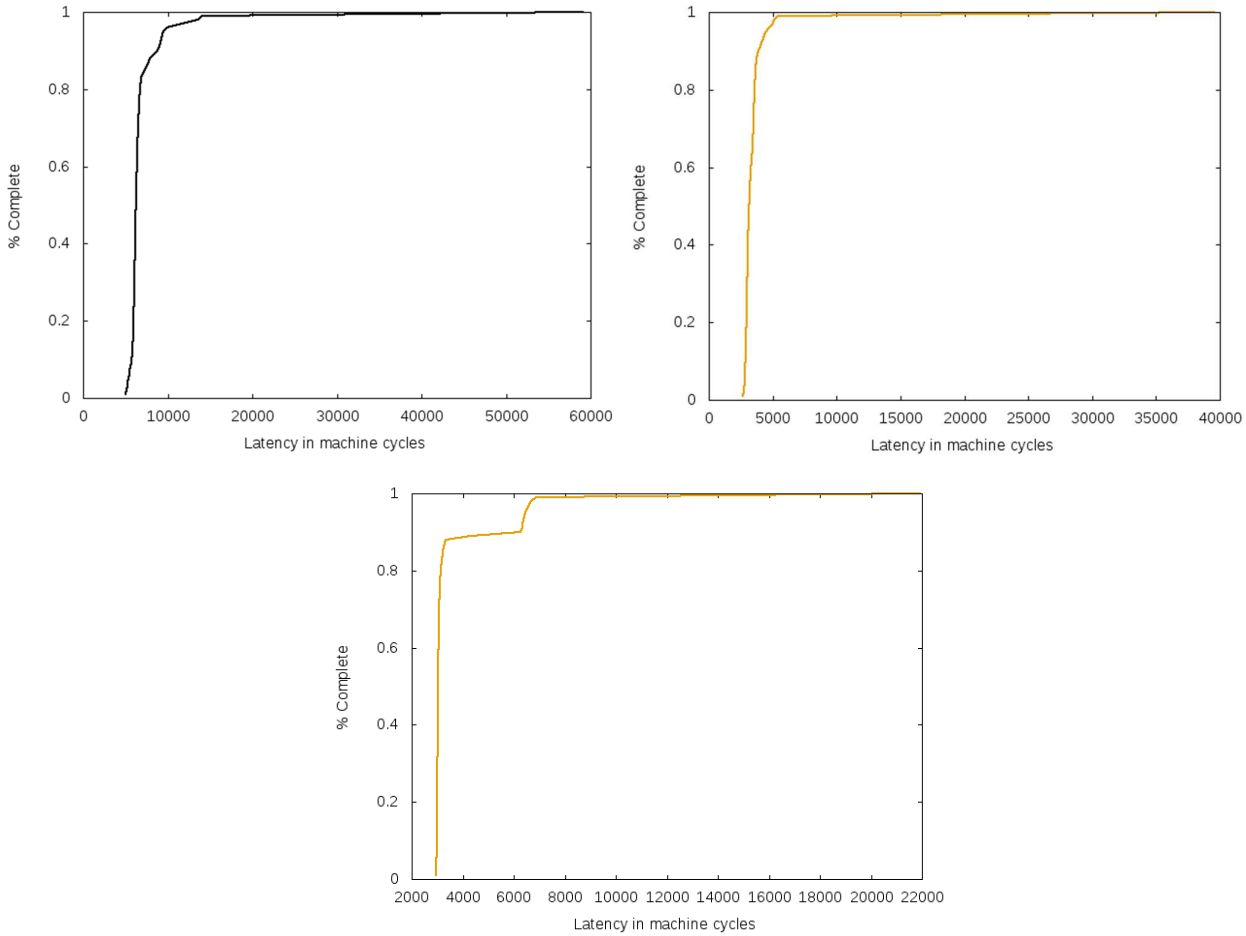

Figure 1: Page fault latencies on Intel Xeon (top left), POWER8 (top right) and ARM64 (bottom)

4.1.3 Analysis

The three graphs show the same general behaviour. This is expected, since the implementation of the page handling code in the Linux kernel is largely generic, despite belonging to different architecture families. Even the parts that are architecture specific have similarities. We can see that over 90% of page faults complete within 10000 cycles (approximately 3.75µs) on the Intel Xeon, and on the POWER8, over 90% of the page faults complete within 5000 cycles (approximately 2.5µs). This gives us the target of completing the network transfer within 2.5µs.

A quick look at the graph reveals something very interesting - there is a very long tail. That means that occasionally, some page faults take an extremely long time to be handled (up to 20x longer than the average time). This may be due to longer code paths being taken, such as in the case that multiple levels of page tables don't exist and need to be allocated in order to complete the original page fault. Since we are trying to optimize for the common case, we don't expect this tail to be critical for our use case, but in any case, it is something interesting to note and should be investigated further.

4.2 REMOTE PAGE FAULT LATENCY

The purpose of this experiment is to know how long a round trip RDMA request takes, as compared to TCP/IP.

4.2.1 Methodology

This test was carried out using Mellanox Connect-X5 cards in Ethernet mode (40GbE) between the POWER8 and OpenPOWER machines. Each test was executed 5 times, and the average was calculated from all of the runs.

TCP/IP

Open a normal socket, send a single byte from the client, and when it is received by the server, have the server send a single byte back. This simulates the case of sending a page fault request, and having the server return a page in response. A more accurate test is to send an entire page, but we first want to determine the lower bound of just the connection overhead. We expect the latency to grow linearly with the number of bytes as the payload increases.

RoCE

Open a connection over a normal TCP/IP socket to exchange RoCE QP (queue pair) information, then open the QP.

Use the QP on the client side to write, then read, a single byte on the server using Infiniband verbs IBV_WR_RDMA_WRITE and IBV_WR_RDMA_READ. This is to approximate the above situation that was performed with TCP/IP. For the purposes of implementing the remote page fault code, only the IBV_WR_RDMA_READ will be required, since the read message will contain the faulting address, and the response will be the data.

4.2.2 Preliminary Results

Protocol	Average Page Fault Time
TCP/IP	58.4268 µs (29985 cycles)
RDMA write + RDMA read	5.1493 µs (2641 cycles)
Only RDMA read	3.377 µs **

** as measured in protocol analyser (Wireshark), not at the client application

4.2.3 Analysis

RDMA is more than 10x faster than TCP/IP when moving a single byte (5.1493 µs vs. 58.4268 µs).

We have some breathing room for OS latency using this method. This also does not include latency for memory mapping, and is for 1 byte (not 4K page).

We can probably do better by using our own protocol. RDMA still has inherent setup costs (such as registering a memory window) which incur high overhead and can likely be avoided in our use case. Our goal in this project is to balance the time taken by the OS to map a page with the time taken by the interconnect to retrieve the page.

4.3 VM MIGRATION

QEMU uses the userfaultfd mechanism to capture accesses to memory pages that have not yet been copied. QEMU calls the userfaultfd() system call that allows registering memory ranges with userfautlfd. The areas of the guest memory to be copied from the source node during migration are registered with userfaultfd. Now when a VM causes a page fault by accessing a memory address that hasn't yet been mapped, the resulting page fault handler will call into userfault to notify the user, and give userspace a

chance to handle the fault. In our case however, we want userfaultfd to call another kernel module, which will send an RDMA message to get the page instead.

4.3.1 Methodology

Visit the QEMU technology page [6] for setup details. This technology was created as part of the ORBIT FP7 project. The migration was performed with upstream QEMU (version 2.9). A virtual machine with a 4GB memory footprint was migrated from one machine to another, using the various methods. QEMU code already contains the capabilities for measuring various statistics such as total time, downtime.

4.3.2 Preliminary Results

Pre-copy over TCP/IP

total time: 692 milliseconds

downtime: 51 milliseconds

setup: 9 milliseconds

transferred ram: 15447 kbytes

throughput: 233.21 mbps

4.3.3 Analysis

We don't have much to analyse in this experiment. When tested with post-copy (results not shown), the downtime often increased, which points to an incomplete (or inefficient) implementation. The RDMA tests were inconclusive – we were able to perform the migration, but didn't see the improvements expected in the results. These tests will be re-run before the final version of the deliverable.

4.4 CONTAINER MIGRATION

The goal is to measure remote page fault latency during a container migration, using current technology. The current technology consists of CRIU accessing pages remotely over a TCP/IP socket. The socket is tested on a 10GbE link on x86, and a 40GbE link on POWER8.

4.4.1 Methodology

This consists of migrating a test application process using CRIU using the post-copy method. Before migrating, the application maps a region of memory using mmap(), and then fills this region with random data to ensure the memory pages are used, and not mapped to the zero page. After resuming the application on the target, it generates 5000 page faults which must be resolved by copying a page from the remote server and mapping it into local memory. Since the live migration mechanism is not yet working satisfactorily, the application is paused and dumped to a file and later restored from the saved memory image. To approximate the situation of a live migration as closely as possible, the memory image is saved to a ramdisk, which avoids any latency that would be incurred by reading pages from a real disk.

In the final version of the deliverable, this test will also be performed with the containerized applications from the VDI use case to show the impact of the technology on a "real world" application.

4.4.2 Preliminary Results

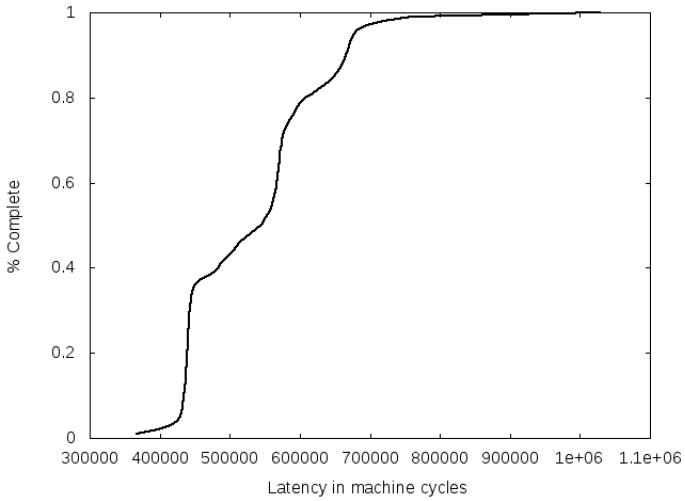

Figure 2: Remote page fault from ramdisk on x86

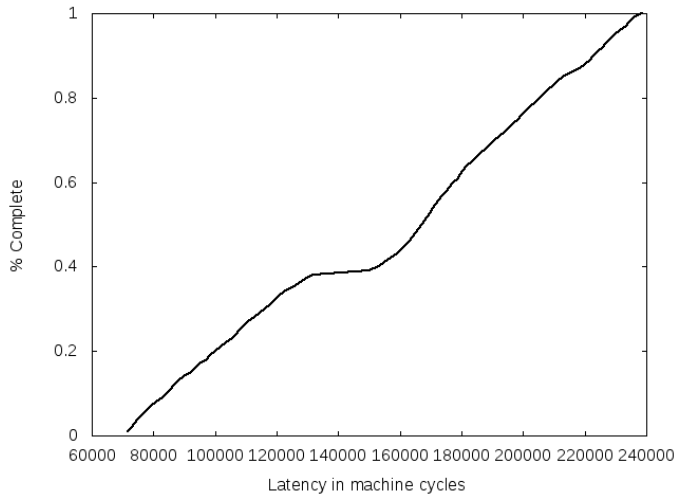

Figure 3: Remote page fault from randisk on POWER8

4.4.3 Analysis

These are the preliminary measurements of container migration using CRIU. CRIU still relies on the checkpoint/restore mechanism to perform migrations. Since the migration is not yet live, we cannot compare the overall migration times or downtime of the application as we can with VM migration (QEMU).

There are many differences between the systems (including interconnect latency, CPU cycle speed, internal memory bus bandwidth, etc.), so it is not correct to compare them to each other. We can use these data to get a feeling for the overall behaviour, and set some expectations for the future. Once the final system is working over RoCE, we will measure these baseline numbers again to gain confidence in the implementation.

5 STATE OF THE ART HARDWARE

The state-of-the-art in interconnect technology is constantly changing, and has changed even since the OPERA project has started. In the OPERA project, we are interested in low latency interconnects in order to support the use case of container migration (microservices) in order to re-balance the load in the data center. Containers have only recently started to find a use in data centers, and migration of containers is not yet in focus as a necessary technology for cloud data centers. Therefore, existing techniques for container migration are quite primitive, and rely on commodity networking equipment (10Gb Ethernet) and standard networking stacks (TCP/IP). It does not take a lot of experience or research to realize that for minimizing remote page fault latency, there are several drawbacks to the technology currently in use, and already today there are existing viable hardware solutions that would provide a marked improvement. In this section we briefly enumerate and describe these existing technologies, so it will become clear how we intend to move past these technologies as part of the OPERA project. All of the existing technologies already use optical links (Infiniband and Ethernet), which has already been recognized in the industry as more cost effective, lower latency, and lower power consumption.

5.1 TCP/IP OVER ETHERNET

This is the de facto standard in place today for data center communication, including migration of VMs and containers, where applicable. Generally data centers today use 10Gb Ethernet, and we use this model as our baseline measurement.

40Gb Ethernet is now available, as well as 100Gb Ethernet. 200 Gb has been announced, and is on the way. These technologies increase bandwidth (i.e. throughput) without regard for latency or power consumption. We do not see these faster versions of Ethernet as a good solution to our particular problem. In many cases, the latency is reduced as a natural artifact of having a faster link, but this comes at the expense of power consumption.

5.2 INFINIBAND

Infiniband is a networking protocol that was designed specifically for data centers. It is superior to Ethernet in several ways, including higher bandwidth, lower latency, and lower overhead on the application and processor. In recent years however, Infiniband has fallen out of favour with data center designers due to the equipment (NICs, switches and cables) being more expensive, harder to configure, and the networking APIs being less well understood, and therefore harder to program. This is unfortunate, since there are nice features in Infiniband such as RDMA which are designed for low latency memory transfers.

5.3 RDMA

RDMA (remote direct memory access) is built on the idea that memory buffers can be set up in advance in order to save time and CPU processing at the time the contents are being transferred. This affects not only the interconnect latency, but also frees up the CPU, since the DMA is offloaded to the card. RDMA began as a feature of the Infiniband protocol suite, but can now be found encapsulated in Ethernet, which exists as RoCE (RDMA over Converged Ethernet). RoCE is the protocol that we feel is the most fitting for our experiments, and the most likely to become popular in the data center. It is because RoCE combines the best of the low latency protocols with inexpensive and well-understood Ethernet equipment that has increased its popularity.

6 CONCLUSIONS

Interconnects play an important role when performing remote page faults. Many factors contribute to the performance, including bandwidth of the host bus, bandwidth of the external bus, and the protocol in use to transfer the data. By being aware of the impact of each of these factors, we will be able to correctly specify and design a NIC that will be used for remote page faults in a heterogeneous data center. We will be able to know what to expect in terms of power consumption of the NIC for a given level of performance, which is measured as page fault latency.

Based on our latest measurements published in this deliverable, we expect to be able to improve performance of remote page fault by approximately 10x by selecting a more appropriate protocol. In addition, this will reduce power consumption of the transfers by reducing or eliminating work done by the CPU to perform unnecessary TCP/IP communication. In addition, (and probably most importantly) this will enable context migration to occur at the scale of the data center, which will allow for power savings at a much greater scale by enabling power-aware scheduling of cloud applications.

7 REFERENCES

[1] PCI SIG, "PCI SIG Specifications," November 2017. [Online]. Available: https://pcisig.com/specifications/pciexpress/. [Accessed November 2017].

[2] B. Vinaik and R. Puri, "Oracle's Sonoma Processor: Advanced Low-cost SPARC Processor for Enterprise Workloads," August 2015. [Online]. Available: https://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.24-Monday-Epub/HC27.24.30-HP-Cloud-Comm-Epub/HC27.24.330-sonoma.Vinalk-oracle-v3.pdf.

[3] Intel Networking Division (ND), "Intel® 82599 10 GbE Controller Datasheet," March 2016. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/82599-10-gbe-controller-datasheet.pdf.

[4] Mellanox Technologies, "ConnectX® Ethernet Single and Dual QSFP28 Port Adapter Card User Manual," October 2017. [Online]. Available: http://www.mellanox.com/related-docs/user_manuals/ConnectX-5_Ethernet_Single_and_Dual_QSFP28_Port_Adapter_Card_User_Manual.pdf.

[5] Mellanox Technologies, "ConnectX®-4 Ethernet Single and Dual QSFP28 Port Adapter Card User Manual," October 2017. [Online]. Available: http://www.mellanox.com/related-docs/user_manuals/ConnectX-4_Ethernet_Single_and_Dual_QSFP28_Port_Adapter_Card_User_Manual.pdf.

[6] D. Gilbert, "QEMU Features: PostCopy Live Migration," Red Hat, 24 November 2015. [Online]. Available: https://wiki.qemu.org/Features/PostCopyLiveMigration.