LOw Power Heterogeneous Architecture
for NExt Generation of SmaRt Infrastructure and Platforms
in Industrial and Societal Applications

Optical Interconnect Energy Efficiency

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

DELIVERABLE NUMBER	D4.4
DELIVERABLE TITLE	Optical Interconnect Energy Efficiency
RESPONSIBLE AUTHOR	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 Metrics
WORKPACKAGE LEADER	Certios B.V.
DELIVERABLE N. | TITLE	D4.4 | Optical Interconnect Energy Efficiency
RESPONSIBLE AUTHOR	J.Nider
DATE OF DELIVERY (CONTRACTUAL)	30/09/2018
DATE OF DELIVERY (SUBMITTED)	30/09/2018
VERSION | STATUS	v1.0 | Final
NATURE	R(Report)
DISSEMINATION LEVEL	PU(Public)
AUTHORS (PARTNER)	IBM, Nallatech, Certios

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Table of contents	01/08/2018	J.Nider
0.2	draft	12/09/2018	J.Nider
0.3	Applied all comments; added conclusion, executive summary	23/09/2018	J.Nider
0.4	Merged final comments	26/09/2018	J.Nider
1.0	Final review	30/09/2018	Giulio URLINI (ST)

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)		Gallig Renaud Email: gallig.renaud@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

API	Application Programmer Interface
FPGA	Field Programmable Gate Array
Gbps	Gigabits per second (unit of measurement of network throughput)
IaaS	Infrastructure as a Service
IP	Internet Protocol
MMU	Memory Management Unit
MTU	Maximum Transmission Unit
NIC	Network Interface Card
PaaS	Platform as a Service
RDMA	Remote Direct Memory Access
RoCE	RDMA over Converged Ethernet
SaaS	Software as a Service
SLA	Service Level Agreement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VHDL	Hardware Description Language
LAN	Local Area Network

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This is the final version of the report on the optical interconnect (NIC) and related page fault protocol. Optimization of the remote page fault mechanism is an important component to enabling container migration as a tool for power management in a compute cluster. By encapsulating microservices and other application components inside containers, power-aware cluster management software can implement various power-saving policies by balancing the workload at runtime among any available node in the cluster, without loss of service.

This report contrasts the original plan to use the FPGA as a prototyping tool for the ultimate remote page fault protocol, with what was actually implemented. The use of off-the-shelf hardware supporting the RoCE protocol limited our ability to experiment and innovate, while at the same time provided a more solid platform for testing, and a stronger roadmap for commercial acceptance. We report on the feasibility of using RoCE (RDMA) as a remote page fault protocol and show that it is not far from the optimum, and that we believe the tradeoffs are worth taking.

It is important to understand the impact on the ability to implement power management policies that are caused by the design of the low-level infrastructure. The infrastructure design has an impact on the rest of the system, up to the application level, which in the end determines how efficiently we can manage the cluster. We analyse 3 layers of the network (physical, network & protocol) by comparing our design of the low latency page fault protocol to the version that was finally implemented using RoCE. The analysis shows that RoCE is a good fit to the requirements set out in D6.5.

Position of the deliverable in the whole project context

The deliverable reports on the hardware used for accelerating remote page fault handling. This is one portion of the overall analysis of the infrastructure for container migration in a heterogeneous data center. The implementation of the software for remote page fault handling as well as much of the results of the experimentation is documented in D6.9.

This report is also related to D5.8 and D5.9 which report on the reasoning behind remote page faults as part of the post-copy container migration strategy for load balancing in the heterogeneous cluster.

TABLE OF CONTENTS

1 INTRODUCTION

Deliverable “D4.7 | Optical Interconnect Energy Efficiency - Intermediate” outlined the plan to take measurements of our experimental NIC based on the Nallatech FPGA card that was designed for extremely low latency transfers of memory pages. Due to unforeseen circumstances early in the project, we were not able to take advantage of the FPGA board as planned. Instead we selected an off-the-shelf NIC (Mellanox ConnectX-5, 1 port, RDMA enabled Ethernet card (MCX515A-CCAT [1]) as a replacement.

Nevertheless, we still report on the design of the page fault protocol, and its initial implementation in VHDL. We compare the design to that of RDMA and discuss the relative performance gap. We show that RDMA Verbs can be used to implement remote page fault handling in an efficient manner.

The use of an off-the-shelf component is often a good choice, because the part has likely undergone more intensive debugging, and therefore fewer issues are expected to arise due to the hardware or software drivers. On the other hand, no modifications can be made to such a card, which makes experimentation more limited. In our particular case, we were not able to implement the protocol that we designed during the first part of the project, which would have lead to the most optimal latencies for remote page faults. In addition, the card we selected did not have the power measurement capabilities that are available on the OPERA FPGA card. However, the results obtained with the RDMA protocol provided by the card have not been disappointing. In fact, by selecting a NIC that has support on several platforms with the backing of a large company behind it has the added benefit of being more likely to be used in a real setting, and thus our research becomes more relevant to a larger audience. We believe this gain offsets the small performance loss taken by not using a customized card.

The details on the replacement of the card, its characteristics, and our software design to use it will be laid out in D6.9 (to be released in M35). This deliverable D4.7 focuses on the impact that the card can have on power savings in the data center. We explore 2 main avenues: the direct energy savings the card itself can provide (which is insignificant) and the overall effect on the data center by enabling workload migration (large effect). Most of these effects as well as the experimental results have been described already in other deliverables.

2 POWER-SAVING STRATEGIES

Our theory for the low-power server portion of the OPERA project revolves around the idea of treating the data center as a single entity. This is an idea that was proposed and implemented first by Google as Warehouse-Scale Computing [2]. Whereas Google's implementation revolves around the correct decomposition and scheduling of 1000's of tasks in a parallel job, our VDI use case is different because it focuses on a small set of long running tasks that have a changing load over time. Despite the differences in workload, the concept of treating the data center as a single entity is the key to efficiency. The different kind of workload means that we need a different approach to managing the workload. Rather than depending only on the initial placement of tasks on different machines, we rely on migration of containerized components between different machines in the data center as they change over time. This is explained in more detail in D5.6 "Workload Characterization - Final Version".

The ability to migrate these long-running application components requires a mechanism to be in place that can perform the migration. We have selected CRIU to perform the migrations, and this is described in detail in WP5 deliverables (D5.3, D5.9). CRIU performs the migrations by opening a standard TCP/IP socket for transferring the application between machines. This is the minimum required functionality, but we proposed to do better. Using such a socket is relatively high latency, which can incur application downtime during migrations, and higher CPU load to process the data. Section 2.1 describes the post-copy migration strategy, and how we can take advantage of it to reduce downtime, but at the cost of increasing latency sensitivity.

Even though the interconnect enables the data center to reduce energy consumption, we do not rely on the NIC itself to provide any power savings. Instead, we rely on the NIC to enable low latency communication between the various servers in the cluster. Only then we can enable cluster management software to be power-aware and to manage the workload according to a power-aware policy. In the following sections we describe the mechanism by which the workloads can be managed, which is a necessity to reduce power consumption in a heterogeneous data center.

By a simple thought experiment, we can show that reducing the power consumption of the NIC would yield very small gains. Let us look at the power consumption of an average cloud server under load. We have measured very small ARM-based servers which draw approximately 50W, and quite large POWER8 servers which draw over 700W at load. Of course, these have very different configurations (physical RAM, number of CPU cores, top clock speed, instructions per cycle, available disks & peripherals, etc.) but let us take the average of 375W as a rough estimate of a cloud server. Referring to Table 1: "Published power ratings of modern NICs from various vendors" in D4.7 "Optical Interconnect Energy Efficiency - Intermediate", we can see the average, modern (≥40Gbps) Ethernet NIC takes about 13.5W at peak. That means the NIC only consumes approximately 3.5% of the total power of the server under load. So even if we were to reduce the power consumption of the NIC to 0W, this would hardly be a significant saving.

2.1 POST-COPY MIGRATION

There are 3 general strategies for container migration: naive, pre-copy and post-copy. These are explained in more detail in D5.3 "Cloud Software Interface – Intermediate Version" Section 6.1 "Comparing Post-copy and Pre-copy Migration Approaches". In reality, a combination of these strategies may be used to perform a migration which minimizes the drawbacks of any one particular strategy. In the VDI use case, the focus is on availability of the service. Especially in the case where services are "scaling up" and need to be migrated to a bigger machine, it is important to be able to continue to serve the clients that are currently using the service and have minimal disruptions during the scale-up. Also in the case of "scaling down", we want to ensure minimal disruptions to the service, at the cost of having slightly longer migration times. To support that behaviour, we focus on the post-copy migration strategy.

As mentioned in D5.3, the biggest drawback to the post-copy approach is the sensitivity to the latency of the page transfers. Since the migration is happening in real-time, the application is restored to a running state before all data has arrived, which can cause hiccups if the memory is not copied quickly enough. This is the reason that we look for the lowest latency approach to serve the page faults from the remote machine.

3 DESIGN CRITERIA

In order to design the best page fault protocol, we must know the requirements that will affect the performance. We analyzed the environment in which container migration is used most, and the effects on the migration performance. Below is a recap of the requirements we set out in the initial design, and an analysis in section 4 compares these to what is being used with RDMA.

3.1 RACK SCALE

We expect the majority of container migrations to take place at the rack-scale. Racks in a data center are often viewed as a logical unit, share a top-of-rack switch for network communication, and it has been shown that most traffic inside a cloud data center is local to the rack [3]. That means we must have an interconnect that can operate well with a node count of between 10 to 40. There are at least two viable options as shown in the SCI (Scalable Coherent Interface) spec [4], and any option that provides low latency connectivity may be used. Figure 1 shows the network architecture of a connected rack.

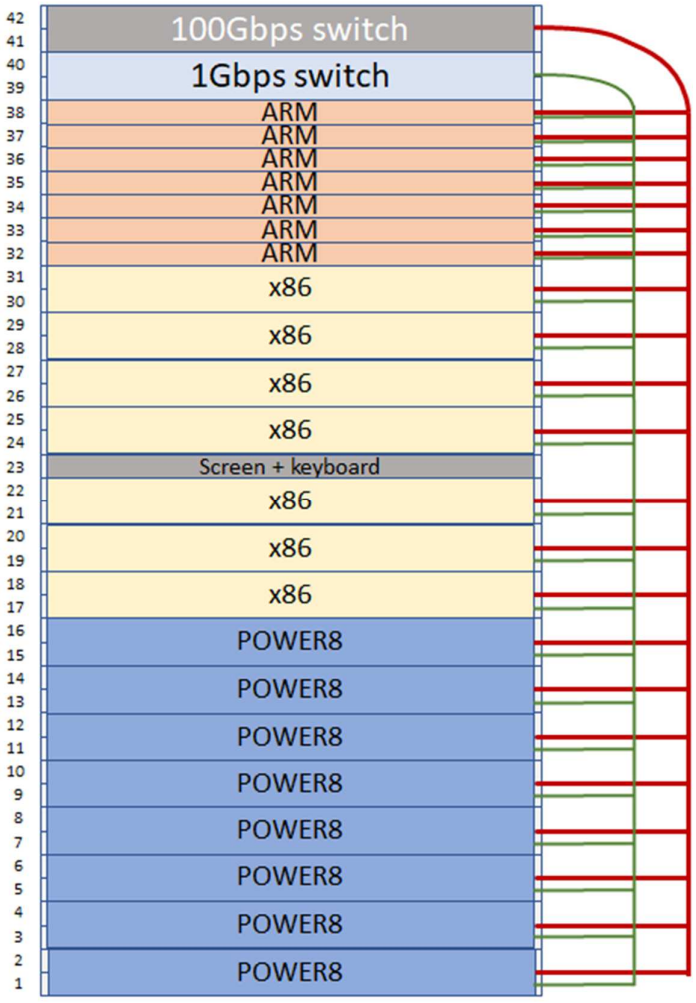

Figure 1 - Standard 19" rack showing a possible configuration of heterogeneous server architectures. 22 servers are connected with a 1Gbps network for management, as well as a 100Gbps network for data and container migrations

3.2 COMMUNICATION BETWEEN NODES

When the host copies a page of memory, we want to be sure that it arrives intact. Therefore, we must assume that the communication protocol employs error detection (such as CRC) and guaranteed delivery in the link layer. Since the communication is local, the protocol should optimize for the best case (assumes delivery) and any recovery (i.e. retransmits) can take a slower path.

3.3 PACKET FORMAT

Since there is no routing involved, the packet format can be very simple, saving the overhead of encoding and decoding many fields. For each process, We are establishing a globally distributed virtual address space, which means the page contents may exist on any machine in the cluster, and the virtual address of the page can be used to locate the data through a broadcast mechanism.

As is shown in the analysis in section 4, the packet format does not have a large impact on the latency and is a relatively small detail of the protocol.

3.4 VIRTUAL ADDRESSING

Before each access by the interface card to memory, the virtual address is translated by an embedded MMU that enables the card to use an address space of a particular process. Thus, the card effectively works in the address space of the target application, and benefits from the existing page tables that are maintained by the operating system. This 'built-in MMU' makes setting up a shared region trivial, as the entire virtual address space of the application is automatically shared, once a process registers for migration. Since the ASID (address space identifier) is added to each request, the card can service page requests from multiple address spaces simultaneously.

Luckily for us, RoCE is designed around the virtual address since it is intended for use directly by the application. That means the NIC contains hardware specifically intended for translating virtual to physical addresses, which is exactly what is needed for resolving remote page faults.

4 PAGE FAULT PROTOCOL

The page fault protocol was designed during the first part of the project and was reported on in D6.5. We completed the first implementation in VHDL but were unable to proceed to the testing and deployment phases after first losing support from Altera, who were acquired by Intel, and later the IBM team that was helping us on a volunteer basis. More details will be provided in D6.9.

After some research, we came across the idea to use RDMA for page faults as a replacement for the FPGA. In this section, we show the analysis of how close the RDMA protocol fits the requirements we set out in the previous deliverables, and how it compares to our prototype implementation.

4.1 PHYSICAL LAYER

At the physical layer, we find that the Mellanox card is an excellent fit to our requirements. At approximately 14.5W, the NIC is at the higher end for power consumption, but still contributes only a small percentage of the overall draw of the server (~3.5% as calculated in section 2). There are other cards available on the market which consume less power and still support the RDMA protocol, however this one has the lowest available latency and we believe the lower latency is more important the saving a few Watts by choosing a lower power NIC. It is possible that a lower rate of power consumption could be attained on this particular card as well by disabling hardware blocks, but we didn't explore this option in depth.

On the positive side, the card provides 100Gbps bandwidth per port (we selected the single port version). The bandwidth is not the most important feature though, but it comes hand-in-hand with low latency. We measured 3.3μs round trip latency, which is quite low. Many other 10Gbps cards are in the 10-15μs range (round trip). According to the datasheet [1], this card has less than 600ns latency (one-ended). As a comparison, the raw latency of the Arria10 FPGA (used by the Nallatech 385A card) is approximately 390ns in a loopback scenario. The FPGA is a 40Gbps link (10Gbps x 4 lanes) which shows that there is no direct correlation between throughput and latency. This also shows that despite being implemented in different material technologies, FPGAs can still be competitive against state-of-the-art ASIC designs.

4.2 NETWORK LAYER

This card is an Ethernet card. Ethernet is a double-edged sword for the OPERA project, giving both positive and negative effects. The positive side of Ethernet is that it is standard. That means we can easily obtain an Ethernet switch, which means building a rack of 20-40 interconnected machines is possible (although 100Gbps switches are a bit expensive). A 4-port switch is about $10,000 [5], and a 16-port switch is about $18,000 [6]. However, it is likely that this kind of expensive equipment is already an existing part of the data center. That means we may reuse such equipment, and the data center operator can get low latency container migration over RDMA for no additional hardware cost. This kind of plan is aligned with the common trend towards converged networks, in which storage and communication now share the same network (specialized Infiniband and FibreChannel networks are being used less and less in new installations). One other issue of selecting Ethernet is the latency of the switch. Infiniband switches are generally lower latency than Ethernet (and also more expensive). A 36-port Infiniband switch shows approximately 90ns port-to-port switching time [7], while the equivalent Ethernet switch is approximately 300ns [8]. However, the difference in software latency is still an order of magnitude larger than the switching latency, which makes it hard to justify the most esoteric and expensive Infiniband network.

The price that we pay for having this backwards compatibility with Ethernet is being stuck with the Ethernet frame layout. Table 1 shows the standard Ethernet frame layout with a variable payload to account for MTU (maximum transmission unit). In fact, the layout itself is not bad, and does not

contribute a lot of overhead. We can calculate the overhead for a small MTU (1500 bytes) and large MTU (4200 bytes) to see the impact. The overhead of the header for a 1500-byte payload is:

$$42/1542 = 2.7\%$$

And for a 4200-byte payload is:

$$42/4242 = 0.9\%$$

Table 1 - Ethernet frame layout

Preamble	Start of frame	MAC dest	MAC source	802.1Q tag (optional)	Length	Payload	CRC32	Interpacket gap
7	1	6	6	4	2	1500 or 4200	4	12

The main source of overhead is tunneling the RDMA request inside the Ethernet frame. Tunneling is a method by which a payload packet is encapsulated by another packet which is used for transmission. Encapsulating an Infiniband RDMA request inside an Ethernet frame is called RoCE v1 (RDMA over Converged Ethernet). Since there is no network layer (L3), such requests can only be sent to machines that are on the same Ethernet segment (i.e. local LAN). In order to get routed packets, RoCE v2 was released. This has even higher overhead, both in frame layout as well as processing at the endpoint because it is tunneled over UDP/IP which adds more encapsulation overhead which consequently requires more processing to extract the RDMA request. For our purposes, RoCE v1 works quite well, because we do not intend to support migrations outside of the rack, and therefore don't require a routable protocol. We can therefore benefit from the lower overhead of the simpler protocol.

4.3 PROTOCOL LAYER

We briefly touch on the implementation of the remote page fault resolution here, but more details are to be found in D6.5 and D6.9. Figure 2 shows the page fault sequence as we designed to be used on the FPGA. We were able to implement this same sequence using RDMA messages rather than our own protocol. There is some additional overhead due to frame layout and MTU, but overall the implementation using RDMA follows the plan quite closely.

At step 3, we send the "remote page request" by using the RDMA READ verb. At step 9, we poll for a completion event on the RDMA completion queue. The rest of the implementation is either specific to the operating system, or to the RDMA hardware.

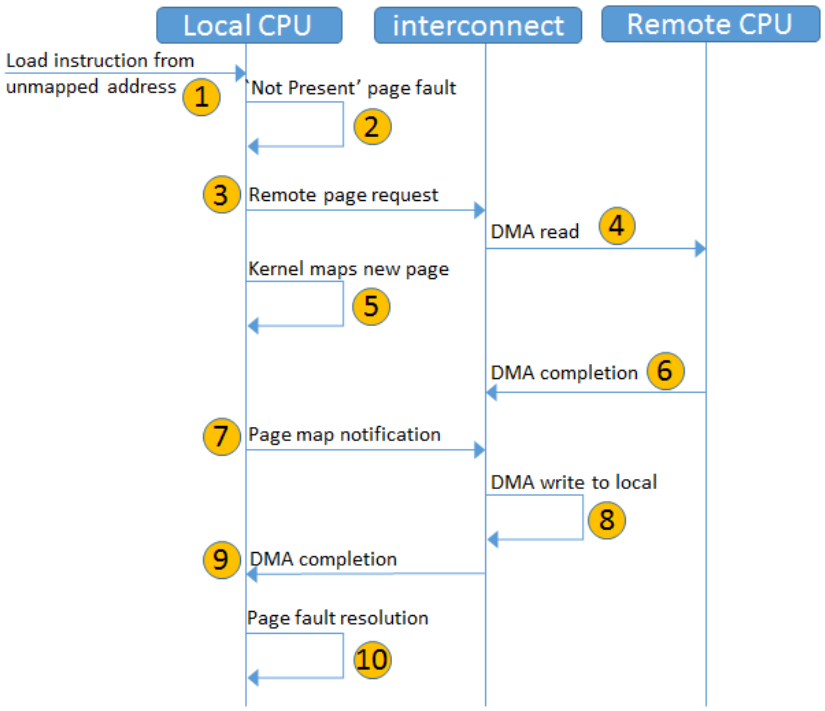

Figure 2 - Remote page fault request sequence

5 REMOTE PAGE FAULT LATENCY

The purpose of the interconnect is to transfer one page of memory from a remote machine to the local machine as quickly as possible, in response to a page fault. Note that during the time the system is resolving the page fault, the process is not doing any work, which causes lags in processing or external user requests to the application. The goal is to resolve the page fault as quickly as possible so as to minimize any impact it may have on application performance. In order to do so, our proposed interconnect would ideally connect to the platform's system bus directly, very much the same as the Remote Memory Controller described in soNUMA [9]. We expect that if this concept is proven useful, the controller would be integrated directly into the chip in the same way DDR or PCIe controllers are integrated today, precluding the need for an external NIC.

So far, we have seen that industry leaders have moved slowly towards integrating memory-attached controllers. This slowness is to be expected since silicon is expensive to produce and debug, and the chip manufacturers want to maximize the effectiveness of their designs and be sure about the usefulness of new features. Therefore, the solution is far more likely to gain acceptance by first using existing equipment (such as PCIe attached controllers), even if it is not ideal. In this light, the move to RDMA over Ethernet makes a lot of sense, even if it was the result of circumstance that drove us to make the decision.

If the technology proves itself in the future and remote page faults become a more important feature, we could see the move to a memory attached controller as being beneficial, since it would further reduce latency.

5.1 METHODOLOGY

There are several factors that affect page fault latency (and therefore impact post-copy container migration). These factors can be divided into those that affect the network, and those that affect the packet processing. Among the factors that affect the network: speed of the physical interface, RoCE version, the MTU size, transfer (page) size, and network congestion. The factors that affect the packet processing are described in D6.5. We take these factors into account when performing the page fault measurements, and report on the importance and impact of each of the major factors on page fault latency.

5.1.1 RoCE Version

The ConnectX-5 card supports two different versions of RoCE - v1 and v2. RoCE v1 is a simpler protocol in which the RDMA requests are tunneled directly inside Ethernet. This has the advantage of being easier to decode as well as slightly less overhead on the wire. RoCE v2 on the other hand, is encapsulated inside UDP and IP. This has the advantage of being routable, which means RDMA requests may be sent through a router between networks, even over the public Internet. Since our use case is inside the data center where there is generally only one network (at least per tenant), we don't have a need for RoCE v2. Even in the case of multiple networks for a particular tenant, the container migrations will likely occur inside a single rack, which means the network will terminate at the top-of-rack switch, belaying the need for a routable protocol. This is important, because the simpler encapsulation method of RoCE v1 has a notable (positive) impact on performance.

5.1.2 MTU

MTU (maximum transmission unit) is the maximum number of bytes that a network interface can send at one time on the media. This was originally intended to limit the amount of time each interface would have to wait in order to transmit on shared media. On today's networks with point-to-point links, is used to cap the amount of traffic that switches and routers need to buffer before forwarding the packet.

This can have an effect on latency since the data may need to be broken up into several smaller Ethernet frames rather than being sent as one large one.

On Linux, the MTU can be set with:

```
sudo ifconfig enp1s0 mtu 1500
```

The Mellanox ConnectX-5 card supports MTU sizes ranging from 256 to 4096 in powers of 2. These are the commonly supported MTU sizes for Infiniband. Since the NIC actually supports Ethernet at the link layer, the MTU must be mapped to the nearest supported size for Ethernet. Therefore, setting the MTU size must be done by specifying a larger size that is supported by Ethernet. For example, setting a 1500 byte Ethernet MTU will set the card to use a 1024 byte RDMA MTU.

We tested with the most common Ethernet MTU sizes supported by the card (1500, 4200) to measure the impact. We have found that the MTU has a small impact on performance, but causes strange behaviour in that sometimes the latency increases (in the case of TCP) and sometimes it decreases (in the case of RoCE).

5.1.3 Page Size

Different architectures have support for different memory page sizes. Each architecture also has its "native" page size, which is the one generally used by Linux for that architecture. That means the amount of data that must be transferred when a page fault occurs is dependent on the architecture. However, the page size is also an important factor when looking at cross-ISA migration, since we want to be able to move individual pages according to the source machine's architecture. That means all machines involved in the transfer should support the same page size, which means we must plan for the lowest common denominator. When looking specifically at x86, POWER and ARM architectures, the lowest common denominator means a page size of 4KB. This is described in more detail in D5.9. The page size affects the time taken to transfer the page (page transfer latency). This also has some effect on the network congestion (see the following paragraph). Table 2 shows the page sizes supported by the 3 server architectures in use in the OPERA project.

	X86_64	POWER	ARMv8
4KB	X	X	X
16KB			X
64KB		X	X
1MB			
2MB	X		
16MB		X	
1GB	X		
16GB		X	

Table 2 - Supported page sizes for various architectures

5.1.4 Network Congestion

Network congestion can be caused by several reasons. The root cause of them all is simply that we are trying to move more data than the available bandwidth will allow. Generally we want to take measurements showing the best case performance, with the understanding that the performance can

degrade down from there. The more resources are available in a network, the less likely we will see the effects of network congestion. Still it is important for network architects to be aware of the expected traffic on each link, and to plan accordingly. For the case of container migration, several factors work in our favour. First, migrations generally do not require a lot of bandwidth. Especially post-copy migrations use little bandwidth because the migration is spread out over a longer period of time. That means we are less sensitive to network congestion, than say a bulk data transfer (such as moving a large file from one storage server to another). The most important factor in container migrations is the latency. Therefore, migration traffic should get a high priority on the network (when quality of service is used in the network) on the understanding that post-copy requires very short bursts of relatively small amounts of data which is extremely latency-sensitive.

5.2 RESULTS ON POWER8

The native page size is 64K, but other page sizes are supported by hardware (as shown in Table 2). We performed microbenchmarks of two page sizes to show the difference. The best results are a 4x improvement using RoCEv1 with MTU=4200 and page size of 4KB. In the worst case, we saw a 37% increase in latency (from 69μs to 95.5μs) using 1500 MTU with a page size of 64KB. In all cases, the jitter was reduced to between 1-2.5μs by using RoCE.

	1500 MTU	4200 MTU
RDMA 64KB	95.5±1.5μs	84±1μs
TCP 64KB	69±7μs	90±16.5μs
RDMA 4KB	16.5±2μs	16.5±2.5μs
TCP 4KB	54 ±22.5μs	64±7μs

Table 3 - Page fault latencies on POWER8

5.3 RESULTS ON OTHER ARCHITECTURES

Results for x86 will be reported in D6.9.

We were not able to measure remote page fault latency on ARM, since we did not have an ARM system that could take advantage of the Mellanox ConnectX-5 NIC. Neither the HPE Moonshot equipped with an ARM cartridge, nor the APM board are able to support a PCIE3 x16 connection required by the NIC. This is another artifact of the immaturity of the ARM server market. In fact, only in the past 2 or 3 months have servers such as Cavium ThunderX2 become available. Even X-Gene (the chip used by the Moonshot as well as our evaluation board from APM) was recently sold to another company called Ampere [10] for further development. Ampere claims that they have shipped samples and are planning production for the second half of 2018. It is still too early to see if they will hit their target for production, but even if they do it will be another 6 months before we will see a board and operating system that supports their chip. This, coupled with the fact that both Qualcomm and Broadcom have recently canceled their ARM server chip development programs does not give us a good feeling for ARM servers in the future.

6 CACHE COHERENCY

6.1 CAPI

CAPI is the cache coherency protocol for external peripherals (attached over the PCIe bus) implemented in POWER8 chipsets. It is intended for connecting peripheral accelerators such as GPUs and FPGAs to become more closely connected to the main processor. The goal is to reduce communication between the main processor and the accelerator, especially to ferry data back and forth. The general idea behind CAPI is to pass cache coherency messages over the PCIe bus to a special hardware component on the card which can process these messages, and update a local cache inside the card. Thus, the card will have an updated picture of what is happening in the main memory, in relation to the memory addresses in use by the card. In this way, the card can share data directly with the main processor (shared memory).

Our original goal was to use the FPGA card provided by Nallatech to create a cross-machine cache coherency domain in order to perform page faults needed for post-copy migration (as explained in Section 2.1). In other words, as the FPGA on machine #1 would receive cache coherency messages from the processor about memory items for the migrating process, it would pass these messages to the second card which would automatically update the main memory on machine #2. While we still believe this design has merit, we were not able to execute our plan to build a prototype of the system. After having designed our system and even implemented much of the FPGA code, we were forced to look for an alternative.

The fact that we could not use CAPI also gave us the chance to rethink the decision from a more practical point of view. We knew from the start that CAPI was only supported on POWER8, and were relying on the capabilities of the FPGA to provide similar functionality on other platforms. This decision was probably too ambitious for the scope of the project and would have led to problems in developing and supporting the cache coherency messaging needed for a full solution. This is the first indication that using off-the-shelf hardware was likely the better decision.

There are several other protocols being introduced that we mentioned in previous deliverables (GenZ, CCIX, OpenCAPI) that provide the same or similar functionality as CAPI. These were “hot” items that generated a lot of hype when they were first introduced, but quickly faded into the background. It is possible that they simply have not yet reached the level of maturity required for uptake by the industry, or that the industry does not yet have a clear need for such protocols. So while theoretically any of these could take the place of CAPI, and we could design a card to take advantage of them, it would be a risky move at this point since it is not clear which hardware/architectures will support them going forward. For this reason as well, it is a good choice to pick an existing standard despite the additional problems encountered during the development.

6.2 ALTERNATIVES

We found that an off-the-shelf hardware design is considerably less expensive than using a custom design. While this conclusion likely does not a come as a surprise, it did make us rethink the cost/benefit of a custom solution. In addition to the cost, using a device that is already accepted in the market is much easier to convince people to buy and try. This reduces the boundaries to adopting the solution, which was not something that we had originally considered when conceiving the optimal solution from a purely technical standpoint. From this perspective, we really have the chance to benefit from the work done by Mellanox and other companies that developed the RoCE (RDMA over Converged Ethernet) standard. While we find that using RoCE has its own challenges, the likelihood of someone using our work after the lifetime of the OPERA project has increased considerably, since both the hardware and software are readily available.

We encountered several problems with the Mellanox NIC (in both firmware and drivers) that we worked with Mellanox to get solved. We have developed software to work with the NIC in both user and kernel mode in Linux, which led to a patent being filed as well as ongoing collaboration with Mellanox in this area. These items are described in more detail in D6.9.

7 CONCLUSIONS

In a long-term project such as OPERA, there can be many unforeseen circumstances that can derail plans. In this particular case, we were not able to take advantage of the FPGA board as planned. Instead we selected an off-the-shelf NIC (Mellanox ConnectX-5) as a replacement.

This change in strategy had an impact on the direction we took in the research and caused us to rethink some of the assumptions we had taken in the early days of the project. In the end, we had positive results and were able to successfully implement a remote page fault protocol over hardware that had not been designed for that purpose. By careful analysis of the requirements that had been set out at the beginning, we were able to find a viable substitute that fulfilled the requirements to lead to positive results.

Despite our initial success, there remains some work left to do to turn our proof-of-concept into a commercially viable solution. However, we believe that we have paved the road to do so and are working with our partners (inside and outside the project) to take advantage of what we have learned.

8 REFERENCES

[1] Mellanox, "ConnectX-5 EN Adapter Card Product Brief," 2018. [Online]. Available: http://www.mellanox.com/related-docs/prod_adapter_cards/PB_ConnectX-5_EN_Card.pdf.

[2] L. A. Barroso, "Warehouse-Scale Computing: The Machinery That Runs the Cloud," 2011. [Online]. Available: https://www.nap.edu/read/13043/chapter/5.

[3] T. Benson, A. Akella and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*, New York, NY, USA, 2010.

[4] IEEE, "1596-1992 - IEEE Standard for Scalable Coherent Interface (SCI)," 1992. [Online]. Available: http://ieeexplore.ieee.org/document/ 347683/references.

[5] Mellanox, "Mellanox SN2010 Ethernet Switch for Hyperconverged Infrastructures," 2018. [Online]. Available: https://www.mellanox.com/related-docs/prod_eth_switches/PB_SN2010.pdf.

[6] Mellanox, "Mellanox SN2100 Open Ethernet Switch," 2018. [Online]. Available: http://www.mellanox.com/related-docs/prod_eth_switches/PB_SN2100.pdf.

[7] Mellanox, "Mellanox SB7800 InfiniBand EDR 100Gb/s Switch System," 2018. [Online]. Available: http://www.mellanox.com/related-docs/prod_ib_switch_systems/pb_sb7800.pdf.

[8] Mellanox, "Mellanox SN2700 Open Ethernet Switch," 2018. [Online]. Available: http://www.mellanox.com/related-docs/prod_eth_switches/PB_SN2700.pdf.

[9] S. Novakovic, A. Daglis, E. Bugnion, B. Falsafi and B. Grot, "Scale-out numa," in *SIGARCH Comput. Archit. News*, 2014.

[10] Ampere Inc., "Ampere Arm Processors," Ampere Inc., February 2018. [Online]. Available: https://amperecomputing.com/product/.