Low latency and high throughput dynamic network infrastructures for high performance datacentre interconnects

Small or medium-scale focused research project (STREP)
Co-funded by the European Commission within the Seventh Framework Programme
Project no. 318606
Strategic objective: Future Networks (ICT-2011.1.1)

Start date of project: November 1st, 2012 (36 months duration)

Deliverable D2.4

Analysis of the application infrastructure management using the proposed DCN network architecture

Due date:	7/09/15
Submission date:	31/08/15
Deliverable leader:	BSC
Author list:	Milica Mrdakovic (BSC), Jose Carlos Sancho (BSC), Hugo Meyer (BSC), David Carrera (UPC), Alessandro Predieri (IRT), and Matteo Biancani (IRT)

Abstract

This deliverable presents the LIGHTNESS technique to properly manage the traffic generated by data center applications and accommodated by the LIGHTNESS network infrastructure. This technique is based on the understanding of the pros and cons of the different optical technologies (OCS and OPS) that LIGTHNESS provides and their performance impact on applications. The performance characterization shows that some applications may show communication patterns that are not well suited to either OPS or OCS. This is due to different factors. OPS is able to switch packets at nanoseconds speed leveraging statistical multiplexing. However, for a certain traffic pattern, packet collisions occur when multiple packets are forwarded to the same output port. Only one packet can be transmitted and the other packets are dropped and retransmitted by the originating server. Thus these packets are suffering communication latency of several tens of nanoseconds. On the other side, we are showing that OCS impacts performance of applications because the milliseconds set up time of the switch. This is because applications change the destination of packets and they need to make new path connections at the OCS. However, once the OCS is set, no additional latency is added. It is shown that both process mapping and application characterization methodology could significantly take advantage of the better performance that OCS technologies are providing in order to efficiently execute HPC workloads. Based on these insights we have proposed an efficient traffic management technique that is based on an efficient mapping which dynamically allocates application traffic to either OCS or OPS depending on the traffic characteristics. In addition, this deliverable shows the strategy for optimizing the energy efficiency in data centers. This is based on the key LIGHTNESS solution that deploys optical network technologies for intra-data centre networks. Also, this document describes the tracing tool needed to obtain traces from real executions on BigData applications needed for the application characterization methodology.

Dissemination Level

☒	**PU:**	Public
☐	**PP:**	Restricted to other programme participants (including the Commission Services)
☐	**RE:**	Restricted to a group specified by the consortium (including the Commission Services)
☐	**CO:**	Confidential, only for members of the consortium (including the Commission Services)

Table of Contents

Figure Summary

Table Summary

0. Executive Summary

This deliverable shows different studies that we have been carried out concerning application management for the Lightness network. These studies can be briefly summarized as follows:

- **Characterization of the impact of optical devices on applications.**

  An understanding of the impact on performance of the different optical devices such as Optical packet switching (OPS) and Optical circuit switching (OCS) to applications have been conducted. The study shows how different characteristic in the communication pattern of applications could perform differently in OPS and OCS. For example, applications that often communicate to multiple destinations could not be appropriate to OCS because the milliseconds set up time of the switch to make new path connection. On the other hand, some application's communication pattern could be prone to produce packet collisions at the OPS. OPS collisions are happening when multiple packets are forwarded to the same output port. Therefore, only one packet can be transmitted at the same time and the other packets are dropped. A methodology has been designed to check for these intrinsic features in the communication pattern of the applications in order to predict the suitability to OCS and OPS.

- **Efficient management traffic technique.**

  We have analysed the impact of process mapping to servers in the system and we have found that mapping plays an important role in order to minimize the negative effects of the optical devices in the performance of applications. This is due to the fact that we can allocate multiple application processes in the same rack in order to reduce the number of either OPS collisions or the number of needed OCS configurations. Based on this insight we have proposed an efficient application management technique that dynamically allocates traffic to either OCS or OPS in an efficient manner to balance the traffic across these different optical technologies. In particular, it tries to allocate traffic first to OCS switches if it's possible to find an efficient mapping and if that is not possible then an OPS will be used instead. Simulations results show that this technique is the one that provides the best performance.

- **Study on energy consumption.**

  The last decade has seen a continuous growth of e-commerce, big data applications and Internet traffic in general, which is turning data centres into the top consumers of energy and electricity. Thus, optimization and efficiency of power consumption is a key business objective for any data centre provider, including Interoute who has a strong interest in operating its pan-European infrastructure in an efficient and cost effective way. Interoute strategy for energy efficiency relies on a set of management procedures based on four key sustainability actions: measure energy consumption, optimize resource utilization, automate data centre operation, monitor and improve the infrastructure. The key enabler is the Interoute proprietary energy consumption model, used to measure and monitor power consumed and energy costs against KPIs defined for each Interoute data centre. On top of these energy efficiency strategy and procedures, Interoute believes that two key LIGHTNESS solutions are crucial technology enablers for power consumption optimization: i) data centre virtualization and multi-tenancy (already implemented by Interoute with its VDC product), ii) optical network technologies for intra-data centre networks (planned for adoption as part of the Interoute infrastructure evolution roadmap).

- **Tracing BigData applications**

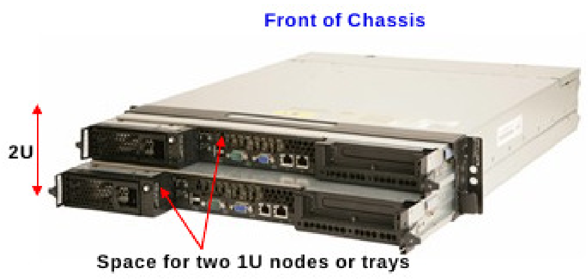
  Big Data frameworks, such as Apache Hadoop, Apache Spark and Apache Storm, are widely adopted across many industries such as health care, manufacturing, public sector, and agriculture in order to deal with the ever growing amounts of information that companies have to deal with. But there is a general lack of understanding of how these frameworks behave when it comes to evaluate their performance and tune their large set of configuration parameters. BigData applications are very network intensive as they usually rely on distributed file systems and distributed processing models to hide the complexity of cluster computing over vast amounts of data. To better understand the use of networks that BigData frameworks perform, in this document we introduce a complete instrumentation environment for BigData frameworks, which is built on top of previous HPC performance tools such as Paraver and Extrae. The tracing and instrumentation environment attaches to the BigData frameworks (initially Hadoop) and collects all kind of performance data. More interestingly for the purposes of LIGHTNESS, it also collects packet-level information about the communication of all the daemons and tasks involved in the execution of Hadoop jobs. The whole instrumentation environment is freely available[1] as an open source project on GitHub.

[1] https://github.com/Aloja/hadoop-instrumentation

1. Introduction

The role of the application management is to efficiently map the different applications' traffic flows to the different optical devices. For this purpose, this document addresses the challenge of understanding the characteristic of the applications that are the key to their performance and how these characteristics are influenced by the usage of optical technologies.

This analysis has been carried out using traces collected from runs of a diverse set of real applications in the Marenostrum supercomputer [1] located in BSC. This supercomputer is the largest supercomputer in Spain with a peak performance of 1.1 Petaflops. MareNostrum's nodes consist of two processors Intel SandyBridge-EP E5-2670/1600 20M 8-core at 2.6 GH with 32GB DDR3-1600 memory modules. The interconnection network is based on InfiniBand FDR10 and Gigabit Ethernet and operating system Linux – Suse distribution. Traces have been collected using the Extrae tool developed at BSC [2]. Figure below shows the single compute node and rack of 84 IBM dx360 M4 compute nodes. Compute nodes are grouped into a 2U Chassis, having two columns of 42 2U Chassis.



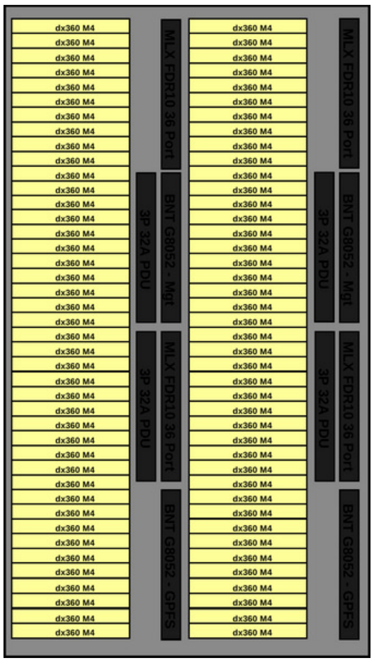


Figure 1: Marenostrum node and compute rack

A diverse set of applications has been analysed consisting of both High Performance Computing (HPC) and Big Data workloads. HPC workloads are selected from diverse set of scientific domains ranging from molecular dynamics to quantum computing. Also, some common Bigdata micro-benchmarks such as Terasort which performs the quicksort function on the MapReduce programming model [3], has been taken into account for this analysis. A more detailed overview of these workload traces is described in the next section.

The methodology followed to understand the impact of the different optical devices consists of developing models that estimate the performance of the workloads on each optical device. These models have been built only looking at the collected workload traces. These traces show the time intervals of applications' communication and computation. Specifically, it could be seen when packets are transmitted in the network and when they arrive at the destination. Note that we are looking solely at the traces without assuming any optical device. Therefore, these traces represent the intrinsic behaviour of the application on a current supercomputer.

For this purpose, we have developed our own tools to look at key parameters that are relevant to predict the performance that this workload could experience when using a certain optical device. For example, for Optical packet switches (OPS) [4] we have looked at the number of concurrent messages in the network. This is because messages could be delayed by the collisions occurring in the OPS when multiple packets are forwarded to the same destination. Similarly, for Optical circuit switches (OCS) [5] we have looked at the number of different destinations of the messages that an application process could transmit because the paths between OCS ports have to be set up prior to transmission. The Paraver tool [6] which is a visualization and analysis tool of the computation and communication events in traces is helping us with this analysis of the applications.

Feature	OPS	OCS
Path setup time	25ns	25ms
Number of ports	16	192
Latency	25ns	0
Packet drop	yes	no

Table 1: Features of the OPS and OCS technologies

Table 1 depicts some of the most important features of the OPS and OCS technologies. These technologies are providing important benefits that no other technology can provide. For example, the path setup time is very short for OPS. On the other case, OCS is very efficient to transmit packets once the path setup is done because there is no latency when transmitting packets through the switch. Another important feature of OCS is that there is no packet drops because there are no optical collisions. And finally, OCS is bigger in size than the OPS technologies. These benefits are going to be explored in more detail in the next sections. In addition, a characterization methodology has been developed to identify the suitability of the workloads to these optical technologies.

Furthermore, based on these insights, it has developed an application management technique to harness the benefits of both OCS and OPS technologies.

The effectiveness of the models has been tested on the LIGHTNESS simulator framework. Figure 2 shows a diagram that illustrates different components that our simulation framework consists of. In particular, applications are run in a supercomputer to obtain traces of their execution and later those traces are fed to our simulation tool. The simulation tool consists of two components: the Dimemas tool [7] that allows us to replay the traces and the LIGHTNESS simulator that models the transmission of packets on the optical network. Different performance metrics are collected during the simulation as well as a new trace that we can visualize in our Paraver tool [6].

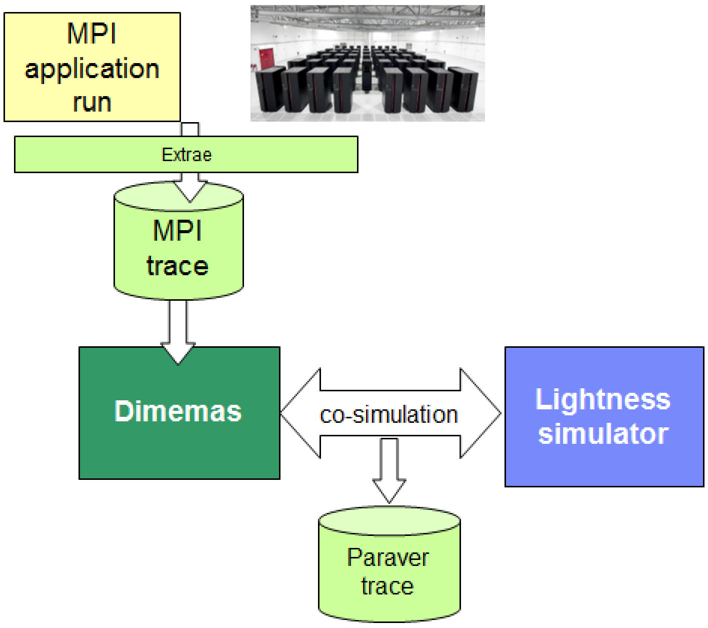


Figure 2: Lightness simulation framework

This study has been extended with the impact of the process mapping on the expected performance experienced by the workload on each optical device. Process mapping is the technique to assign each application process to each server in the system. Results show that this technique is critical in order to minimize the impact of the different optical devices, especially the OCS. The mapping was applied to the HPC and Big Data applications.

The rest of the deliverable is structured as follows. Section 2 describes the workloads analysed. Section 0 presents the different models built to analyse the impact on performance of OPS and OCS optical devices and also the tool to trace Big Data workloads. Section 4 proposes some application management techniques. Section 5 describes the energy consumption in data centres. And finally section 5 contains proposal for improving energy efficiency in data centres.

2. Workloads

This section provides a description on the different workloads analysed in the simulation activities reported in this deliverable. These applications represent a diverse set of applications mostly used in HPC. Both HPC and Big Data workloads have been taken into account. HPC workloads are summarized in the following

Table 2. All these workloads are written using the MPI parallel programming model. In particular, 16 MPI processes are considered per each application.

Name	Description	Global problem size for the default input
HYDRO [8]	Solves Euler equations of hydrodynamics	250x125
MILC [9]	4D-SU3 lattice gauge computations	16x16x16x16
MINI_MD [10]	Molecular dynamics application LAMMPS	32x32x32
SNAP [11]	Particle transport application	4x4x4
CG [12]	Conjugate gradient solver from NAS parallel benchmarks	14000
MG [12]	Mesh-based multigrid solver from NAS parallel benchmark	256x256x256

Table 2: Description of the HPC workloads

Traces for these applications have been collected on the Marenostrum supercomputer. In order to compare the behaviour of these applications we have taken a representative cut of the related traces with same duration in time. The duration was fixed to 4ms. This cut includes one or more iterations of the abovementioned applications. An overview of the different traces for this time interval is shown in Figure 3, and as can be seen, these HPC applications show a repetitive behaviour over time. This is because HPC applications iterate over time mostly because they are using iterative solvers. Usually different MPI processes, which an HPC application is composed of, are communicating mostly at the same time. In addition, although this repetitive behaviour characterizes HPC applications we can see that their communication characteristics are quite different among them. For example, *SNAP*, *MILC*, and *CG* are communicating more frequently than the other ones whereas *MG* only shows one communication phase. In addition, *HYDRO* and *SNAP* are the ones that show the use of some collective communications like *MPI_Allreduce* while the rest only uses point-to-point communication operations such as *MPI_Send/MPI_Receive*.

Regarding the time that these applications spend on doing computation without transferring any message, the behaviours are quite different as well. There are applications that spend most of their time computing, such as *MINI_MD* and *MG* and on the other side *SNAP* spends very little time computing.

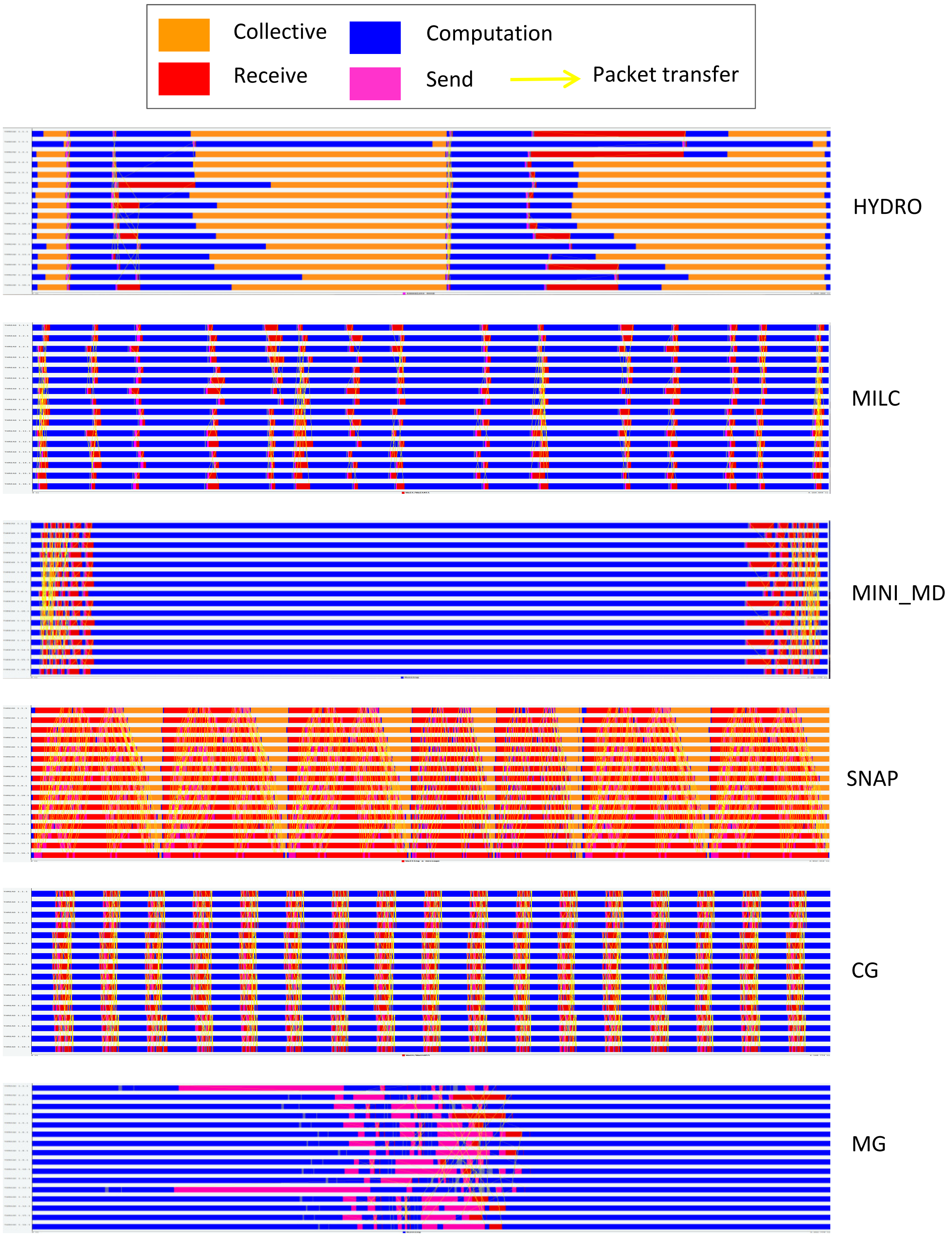


Figure 3: Snapshots of the different traces for HPC workloads

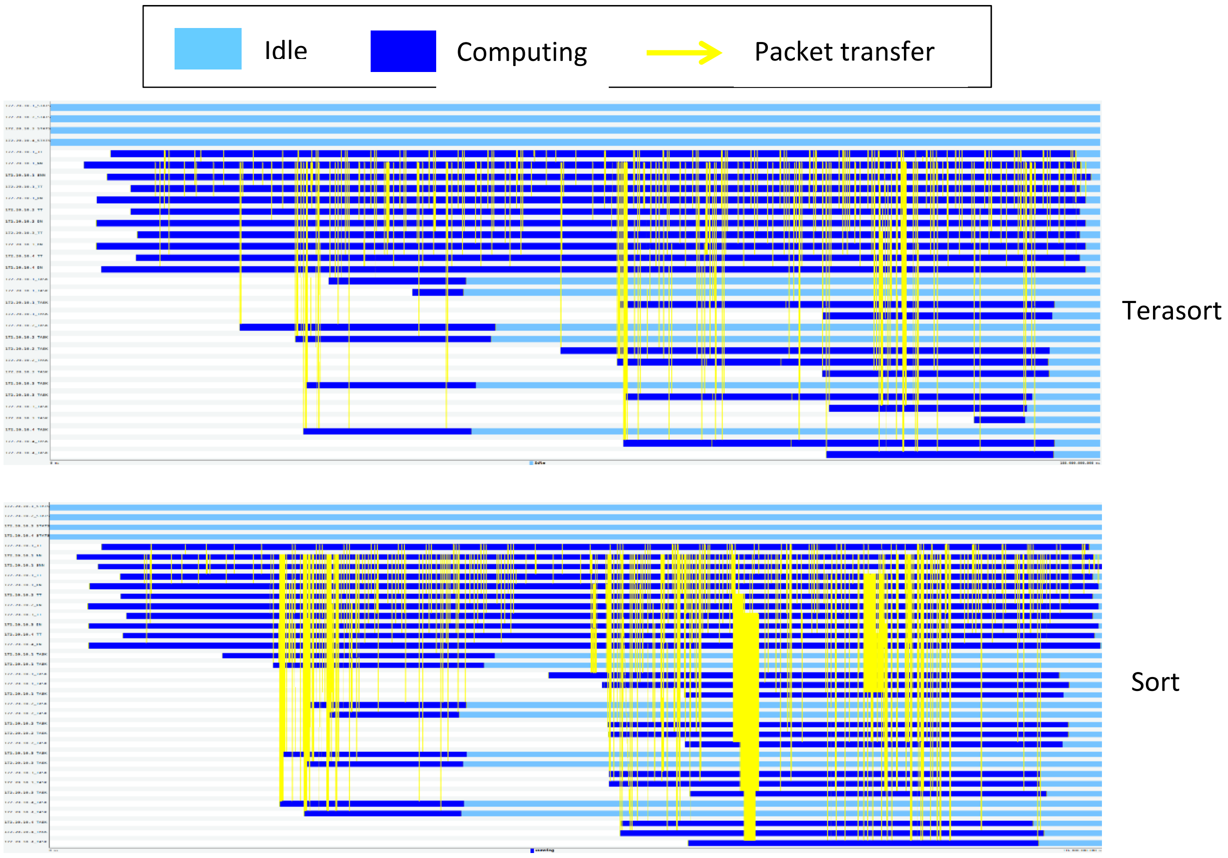


Figure 4: Snapshot of BigData traces.

Figure 4 shows the snapshot of two BigData workloads called Sort and Terasort. These are standard benchmarks of Apache Hadoop programing model of BigData. These snapshots correspond to the whole duration of the benchmark. In total there are 35 processes where 20 of them are workers. Sort is a standard map/reduce sort that sorts the data into a total order. A data set consisting of 100,000 data points was considered. TeraSort is a more sophisticated algorithm with respect to Sort that basically compresses the data prior to communicating through network.

As can be seen, BigData benchmarks behave quite differently from HPC workloads. They are not as regular as HPC workloads. Communications are scattered through the whole execution with no regular pattern apparently. In addition, Figure 5 shows a zoom of a cut of the same duration as in the snapshots for HPC workloads (4ms). As it is shown, not all the processes communicate and the ones that communicate transfer the amount of packets during 4ms much less than the HPC workloads.

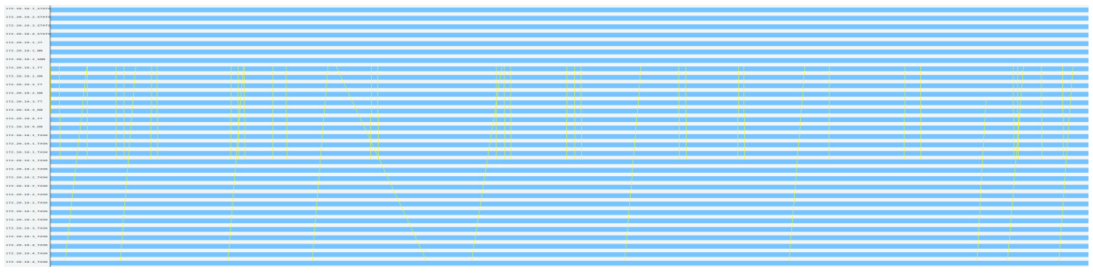


Figure 5: Cut of the Terasort trace

3. OPS and OCS suitability to different workloads

In this section, we introduce the methodology to characterize workloads regarding their suitability to use OPS or OCS switches. The methodology consists of analysing key intrinsic characteristics of the application communication pattern and mapping these characteristic to the potential performance impact that they will suffer from when using a particular optical device.

In order to compare across different workloads, the traces taken for the analysis have the same duration for all the applications. Hence, we can draw conclusions of which workloads are more demanding than others, for example in terms of communication requirements.

Furthermore, this analysis will be independent of the final mapping of application process to optical resources such as wavelengths and fibers and even servers in the data centre. We understand that this mapping will have a significant impact on the performance of the different applications. It could even reduce the performance impact for example in the case of mapping each application server to an independent fiber when using OPS. First, we decided to conduct the analysis from the application point of view without considering any mapping. Therefore, the results presented will be viewed as using a worst mapping. In a second stage, we will show the impact of mapping.

Section 3.1 reports the analysis of the suitability for OPS for both HPC and Big Data workloads. Section 3.2 presents the analysis for the OCS.

3.1. OPS suitability

This section analyses the suitability of the workloads to use OPS. It analyses HPC and Bigdata traces collected from the Marenostrum supercomputer. Key characteristics in the behaviour of their communications that could impact the application performance will be discussed. As we know, OPS could drop packets due to collisions on the output port. A packet drop could introduce a negative impact in the performance of applications because this packet will be retransmitted, delaying the time of receiving it by the destination process. The potential of an application to suffer OPS collisions will be analysed in the methodology that we have developed. Furthermore, this methodology consists of prediction of the performance due to the amount of collisions and the sensitivity of the application to these collisions.

3.1.1. Characterization methodology

In this section we describe the characterization methodology to predict the OPS suitability to the different applications. Figure 6 illustrates this methodology. Basically, it consists of two main building blocks. First, the Concurrency component is looking at the amount of concurrent packets that an application could send over the network in a data centre. The Collisions part will indicate the percentage of messages that will suffer from collisions regarding specific mapping. We also analysed the Sensitivity as the second component. Sensitivity indicates how much the network communication is globally impacting the application performance. There could be the case that the application is mostly computing and hence communications are not significantly impacting application execution. Application metrics such as communication volume or data rate are important to be analysed.

Obviously, a concurrency is needed in order to have collisions at the OPS. Otherwise, when there is no concurrency there will be no negative impact on performance. On the other hand, when the application exhibits some concurrency level, then depending on the mapping, there might be a percentage of collisions. By using an approach without looking at the mapping, we wanted to see if some applications are more likely to suffer from certain level of performance degradation depending how sensitive they are regarding only the intrinsic nature of themselves. Note that the higher is the concurrency level the higher is the number of potential OPS collisions, but we have tried to find mapping technique that could almost fully avoid OPS collisions.

The estimated level of performance degradation is a reference value in order to compare the behaviour across multiple applications. It will be unfeasible to give an exact performance degradation value. This will be accomplished by using our simulation tool. Therefore, without simulation we will only give a coarse grain level of performance degradation. The following sections will describe in detail the algorithms used for analysing the concurrency, collisions and sensitivity.

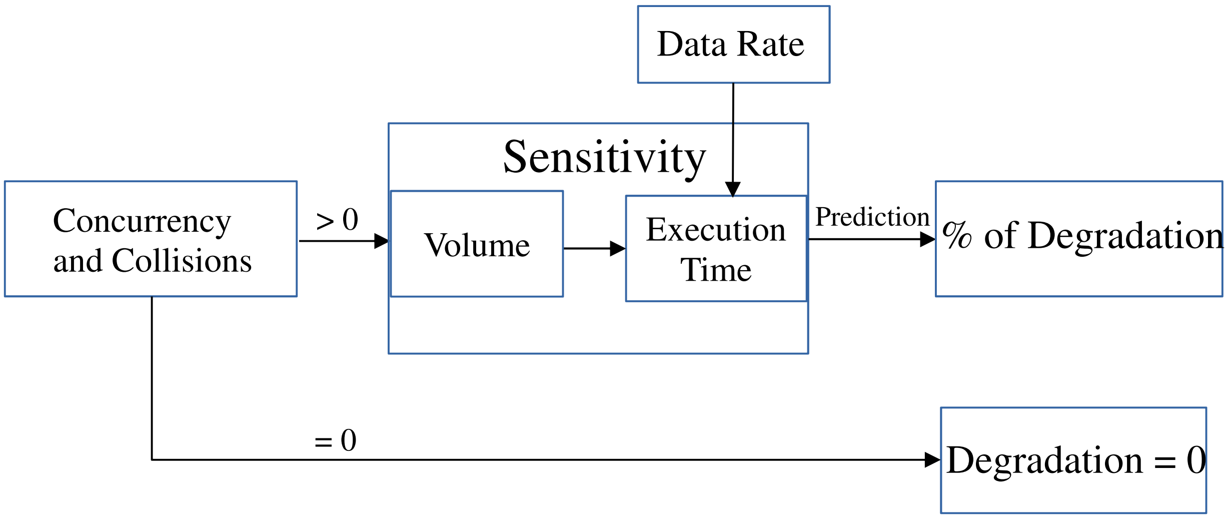


Figure 6: Characterization methodology for OPS

3.1.2. Concurrency and collisions

Applications are characterized by the time/data rate in which processors are sending packets. When more than one packet is transferred through the network at the same time we call that a *concurrency*. Note that this metric is fully dependent on the speed of the network. In this analysis we considered network bandwidth of MareNostrum, 56Gb/s. If the network's bandwidth is lower than the analysed one, we expect that there will be even more concurrency. As known, when there are simultaneous packets in the network there is more possibility for message collisions in the OPS. For example, if there are two packets at the same time in the network then eventually these two packets could produce an OPS collision. Obviously, the OPS collision will only occur if those packets are coming from the same fiber and going to the same OPS output port. As can be seen, the final mapping of the processes to servers and thus optical resources are playing an important role, see Figure 7.

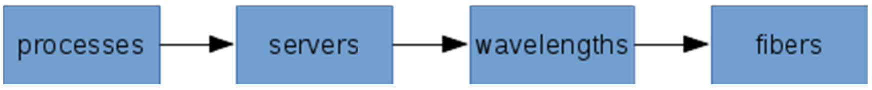


Figure 7: Mapping of process to servers and optical devices

For the sake of illustration, Figure 8 shows an example of packet concurrency on various packets that are sent by four application processes. The blue rectangle represents the transmission time of one message, for example Msg2/2 is in the network with Msg2/4 whereas Msg1/1 is in the network with three other packets, Msg1/2, Msg1/3, and Msg2/4.

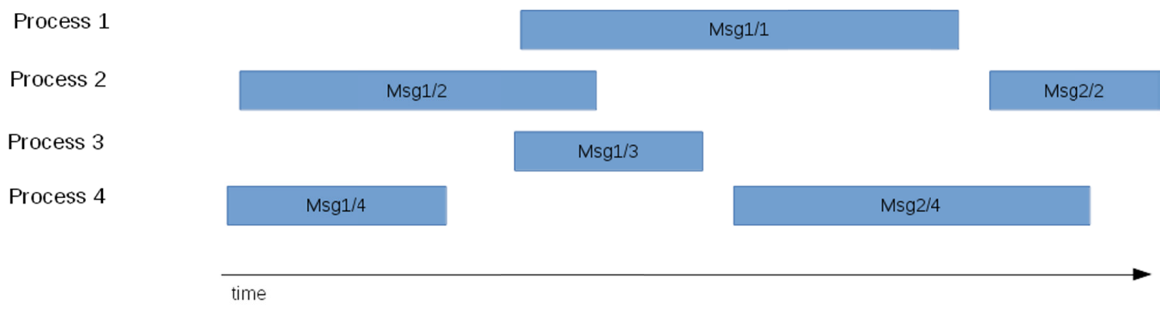


Figure 8: Packet concurrency

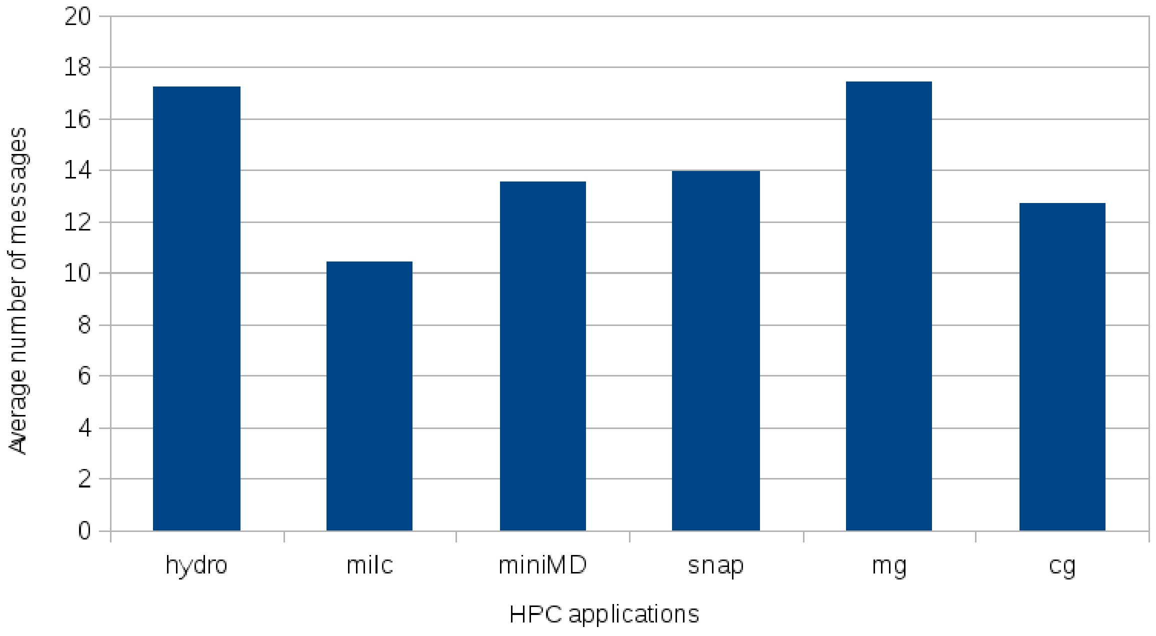
In order to measure the number of simultaneous packets that could be in the network we have developed an algorithm that is presented in Figure 9. The input is obtained from the trace file, taking the necessary information about every packet sent such as the time when the packet was sent by the sender process and the time at which this packet will be received by the destination process. *Trace* represents the collection of all sent messages. For every message we took the *Send* parameter which represents the starting time of message transmission, and *Receive* parameter which represents the receiving moment of the message. This algorithm will show us the average number of messages that are present at the same time in the network per message. *Concurrent* list contains number of messages that flow through the network at the same time for every message in the application. Summarizing all of them (*Summ_of_msgs*) and dividing that with number of messages (*Message_number*), we get the number of simultaneous messages on average per application message (*Average_concurrency*). We also analysed the number of possible collisions (*Average_collisions*) using several different mappings. We counted collisions (*Collisions*) if simultaneous messages are coming from the same rack (*SourceRack*) and are sent to the same destination rack (*DestRack*).

```
Message_number = 0
for msg1 in Trace do
   for msg2 in Trace do
      if(Send(msg2) > (Send(msg1)) and (Send(msg2) < Receive(msg1)) do
            Concurrent[Message_number] ++
            if(SourceRack(msg1) = SourceRack(msg2)) and \
            (DestRack(msg1) = DestRack(msg2)) do
                  Collisions ++
            end
      end
      elif(Send(msg2) < Send(msg1)) and (Receive(msg2) > Send(msg1)) do
            Concurrent[Message_number] ++
            if(SourceRack(msg1) = SourceRack(msg2)) and \
            (DestRack(msg1) = DestRack(msg2)) do
                  Collisions ++
            end
      end
   end
Summ_of_msgs += Concurrent[Message_number]
Message_number ++
end
Average_concurrency = Summ_of_msgs/ Message_number
Average_collisions = Collisions/(Message_number * (Message_number - 1))
```

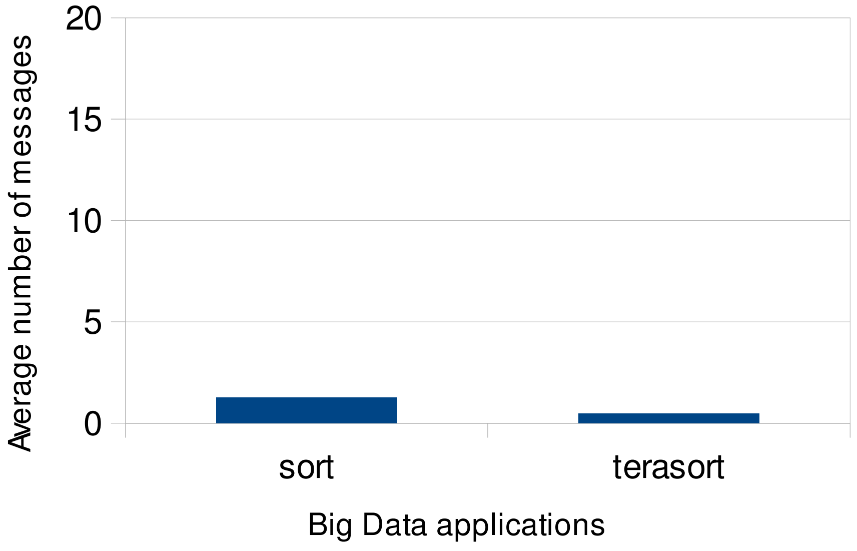
Figure 9: Calculation of concurrency and collisions

Figure 10a shows the concurrency obtained for the different HPC applications presented in section 2. As can be seen, *HYDRO* and *MG* have the highest and approximately the same number of simultaneous messages at the same time. On the other hand, *MILC* has the lowest average number of concurrent messages. In a zoomed section of the *MILC* trace in Figure 11, this application has several groups of parallel communication but in one communication phase sending of messages is not that synchronized comparing to other applications, so the concurrency is lower. On the other hand, for *HYDRO* (Figure 12), sending of all servers (server is represented as *thread*) is quite simultaneous so the concurrency is 100% - the transmission of every message overlaps in time with messages of all other servers. The case of more simultaneous messages than the number of servers is also possible. Servers can send more than one message in short time interval, so the message flows from the same server could overlap and produce an OPS collision. This type of collision was not possible when the applications were run by the simulator. The simulator would never overlap packets of several different messages.

Figure 10b shows the concurrency of Big Data applications. These applications do not transfer a lot of data between the servers so the concurrency is really low.



(a)



(b)

Figure 10: Average number of concurrent messages

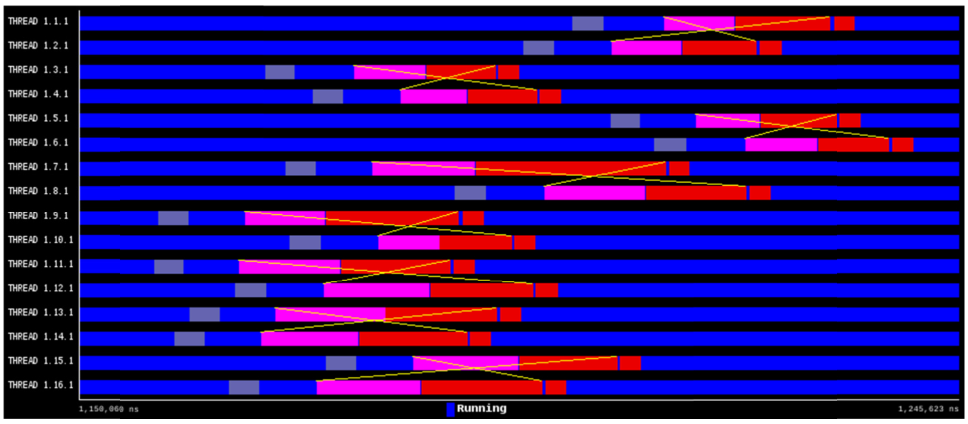


Figure 11: MILC communication phase

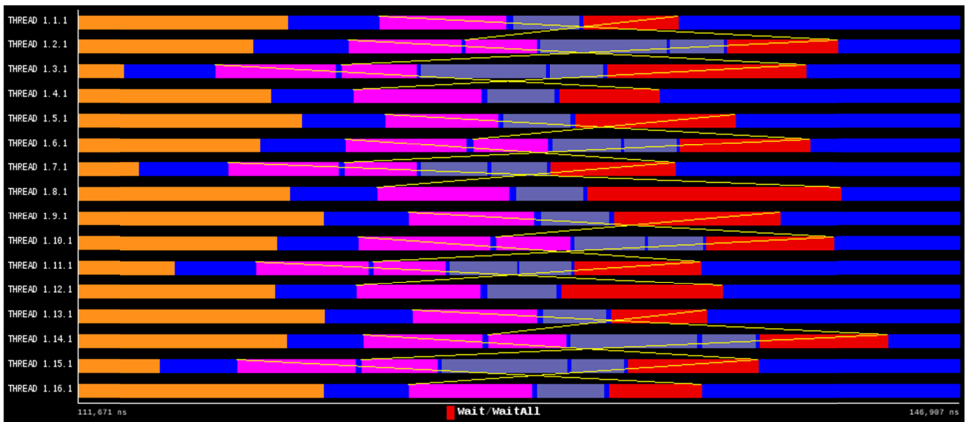


Figure 12: HYDRO - communication phase

Figure 13 shows that number of collisions regarding the number of messages that are sent is really low. The percentage goes under 0.8% for all HPC applications and this percentage is acceptable regarding the impact of one retransmission (60ns for 1m) and the fact that a lot of retransmissions are simultaneous. Using the mapping the impact of high concurrency could be avoided.

Mapping from Figure 13 includes two servers per rack, eight fibers per rack and two wavelengths per fiber so messages for both servers could be sent at the same time. We could conclude that is very rare that two servers from the same rack send their messages simultaneously to two other servers that are packed in the same rack. Also high percentage of messages is exchanged inside the rack. The intra-communication will be even higher if we group more than two servers in a rack. As a result we would not expect that the changing of the mapping can significantly change the percentage of collisions.

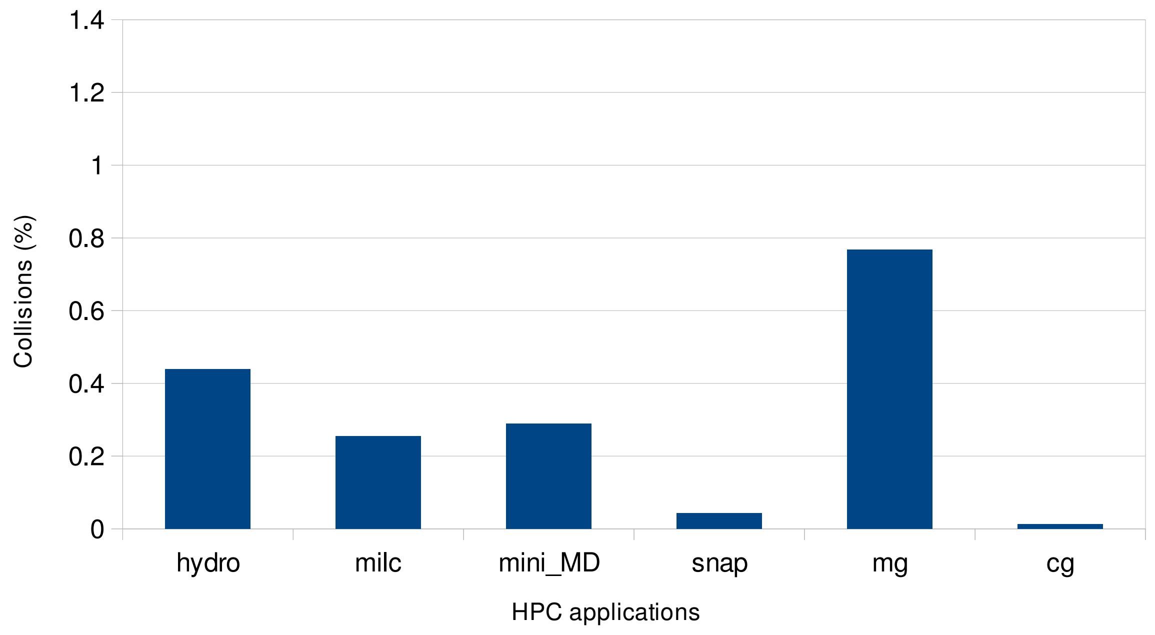


Figure 13: Collisions percentage – two servers per rack

If it is not possible to use exactly the mentioned mapping of processes on servers, fibers and wavelengths, it is important to study the nature of every application. Considering only the characteristic of every application, we analysed the sensitivity of applications to possible collisions. Sensitivity shows how the collisions will impact the performance of an application. As an example Big Data traces have 0% of the collisions.

3.1.3. Sensitivity

All applications are more or less sensitive to packet collisions. The sensitivity tries to capture the impact of the communication delay on the whole application execution. This delay could happen due to existence of OPS collisions.

Figure 14 shows two different scenarios of application execution. In the first scenario, there are two different applications App1 and App2 of the same duration, but App2 is sending more packets than App1. The other scenario assumes that each of the packets got delayed because some OPS collisions. The application that transfers more packets (App2) has a longer execution time. Also note that this application's execution time is higher because it spends more time communicating than App1 during its execution and more collisions occur.

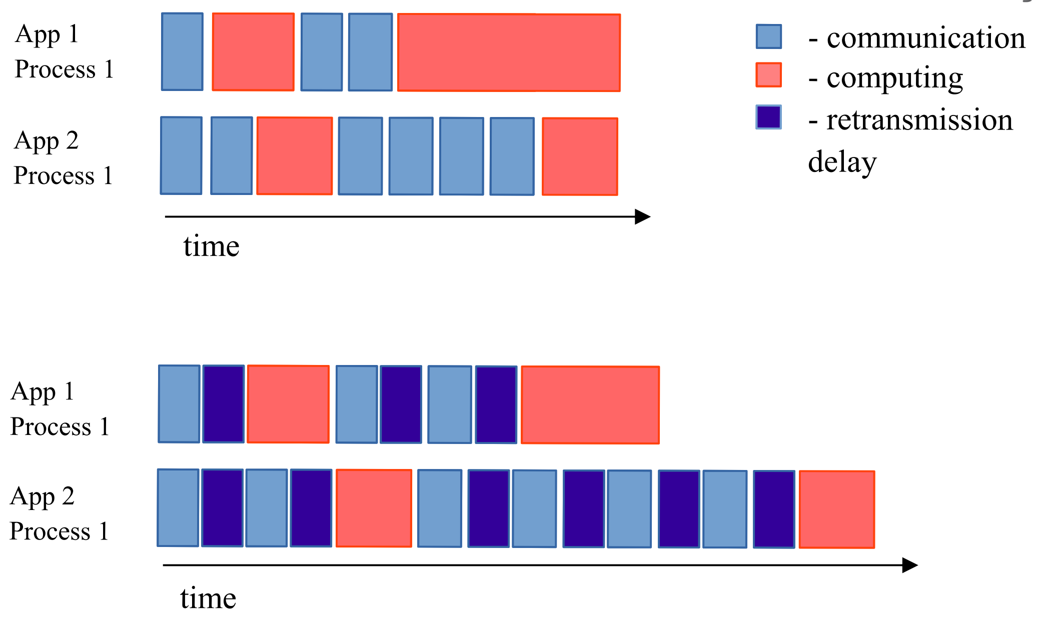


Figure 14: Visualization of the different impact to application execution

The quantity, i.e. total number of bytes that an application transfers, will indicate how specific applications are sensitive are to collisions.

Figure 15 and Figure 16 show the communication volume for various HPC and Big Data applications, respectively. Note that comparison of Big Data and HPC applications is possible because the analysis were done over the same trace duration for each application. We can see that HPC applications could exhibit much more communication volume than Big Data applications. In particular, *CG* shows 22.86MB whereas the most communicating Big Data application is reaching only 0.42MB. In addition, HPC applications show more variety on the communication volume across applications than Big Data applications. For example, the difference between the HPC application with highest volume and the lowest is huge, whereas this difference for Big Data is much smaller. Among the HPC applications the application with highest communication volume is *CG* and the lowest ones are *HYDRO* and *SNAP*.

As expected the applications with a high communication volume are the ones that show the highest data rate because all these traces have the same duration. Figure 17 shows the data rate of the HPC applications.

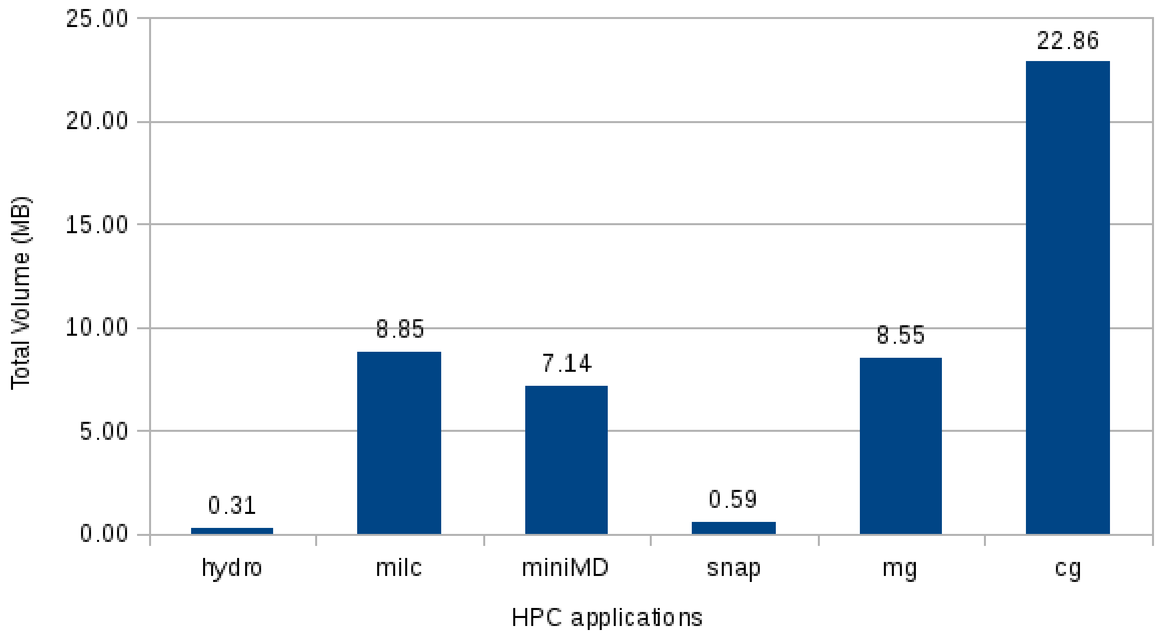


Figure 15: Communication volume for various HPC applications

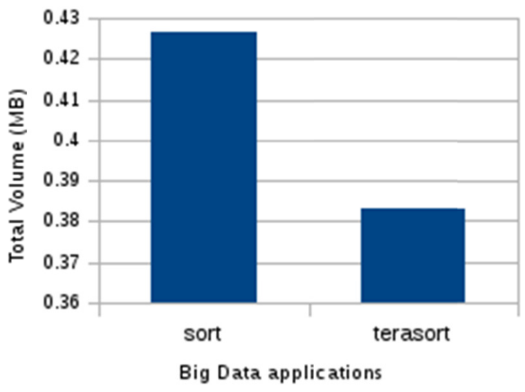


Figure 16: Communication volume for various BigData applications

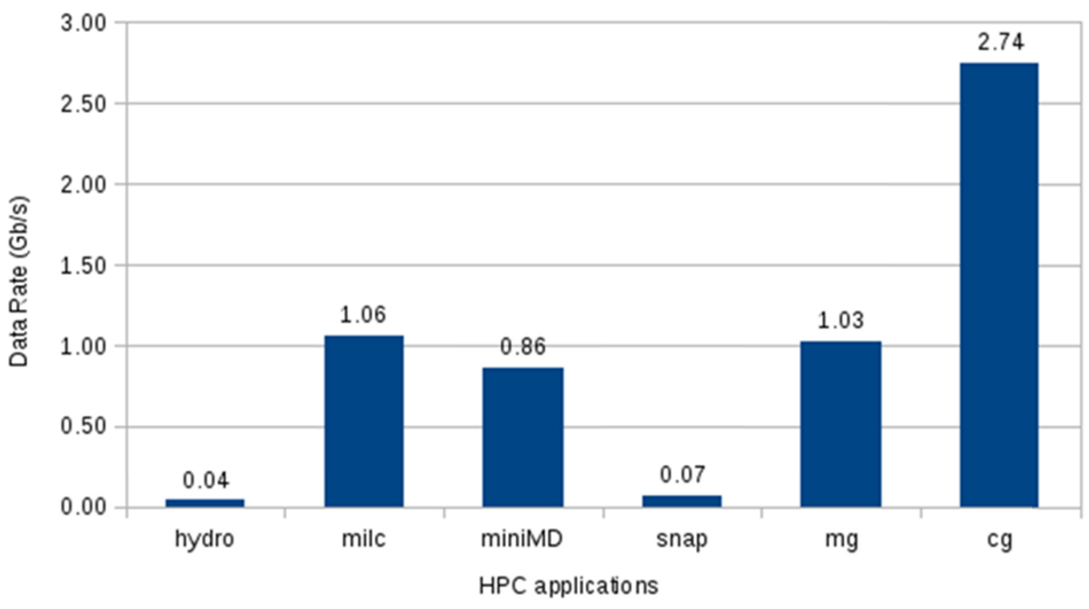


Figure 17: Data rate for HPC applications

If the mapping is not suitable for avoiding the collisions and if we consider that all the applications in reality last much longer than the small parts of traces that we cut for analyses, we can conclude that based on these results it is expected that the impact of OPS collisions is more pronounced to HPC applications such as CG, MILC, MINI_MD, and MG because they show more communication volume. On the other hand, it is expected that OPS collisions have a lower impact on HYDRO and SNAP, and also on Big Data applications because their communication volume is low. This prediction is summarized in the following Table 3.

Application	Potential impact
CG	*High*
MILC	*Medium*
MG	*Medium*
MINIMD	*Medium*
SNAP	*Low*
SORT	*Low*
TERASORT	*Low*
HYDRO	*Low*

Table 3: Potential OPS impact

These predictions are theoretical and they could be significant if we use whole duration of applications. Then the percentage of collided messages could be significant because the number of messages in one application run is much higher than in trace cuts that we analysed.

Regarding selected trace cuts we also got additional results that show us how mapping could make the possibility for collisions really low. This behaviour is validated using our simulation tool that models in detail the behaviour of sending packets through the network (Figure 19). The simulation tool was modelled using OMNEST [13] and the used bandwidth was 10Gb/s. Results are taken combining two servers in one rack, where every rack uses eight fibers and every fiber has two different wavelengths (Figure 18b). Execution time of each application is almost the same or shorter comparing to the default mapping case, so the impact of collisions is insignificant as predicted (Figure 13). Default mapping assumes one server per rack, 16 fibers per rack and one wavelength per fiber (Figure 18a).

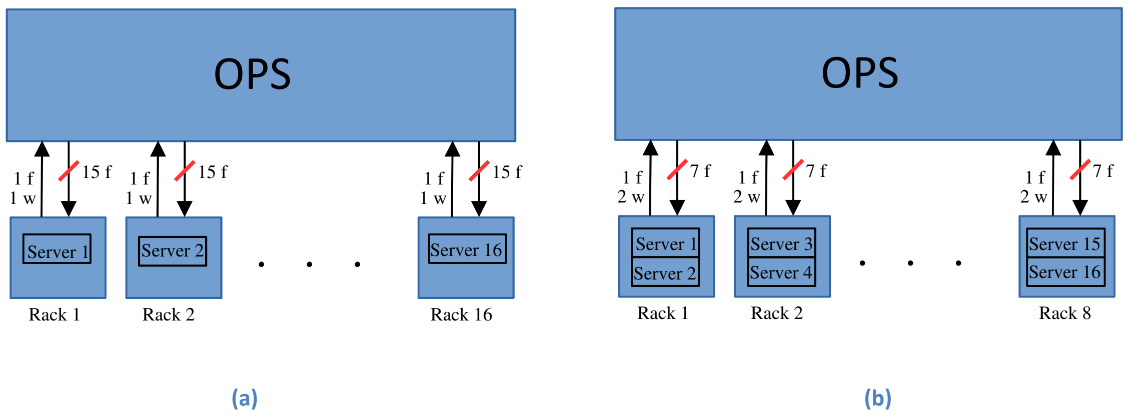


Figure 18: Default mapping and mapping with collisions

One fiber is for outgoing traffic and another 15 fibers are for incoming traffic of each rack. This mapping assures that packets from each server to any other one can pass without collision with other packets. Table 4 shows which values were used to simulate the network.

Parameter	Duration
OPS latency	30ns
OCS latency	0
ACK delay	20ns
cable propagation	5ns
Wavelength bandwidth	10Gb/s
burst mode receiver delay	100ns

Table 4: Simulation parameters

Small differences between predicted and obtained results exist because of different network bandwidths. The predictions used network bandwidth of 56Gb/s and simulations used bandwidth of 10Gb/s. The applications that have shorter execution time than the default case are possible, because the communication between servers in one rack is faster than the communication between the servers through the network, especially when we have lower network bandwidth. This can be observed in the result of *SNAP* application. *CG* application has slightly higher performance degradation, but this is also expected because of its sensitivity to collisions. Although it has quite low percentage of collisions, the amount of traffic is the highest when comparing to other applications which makes the application more sensitive.

We can conclude that mapping suits the applications and makes them more resistant to collisions. At the end there is no significant change in execution time. Only *SNAP* application has noticeably better results when combining servers into racks. In the next sub-section we will see how different mappings affect performance of all applications.

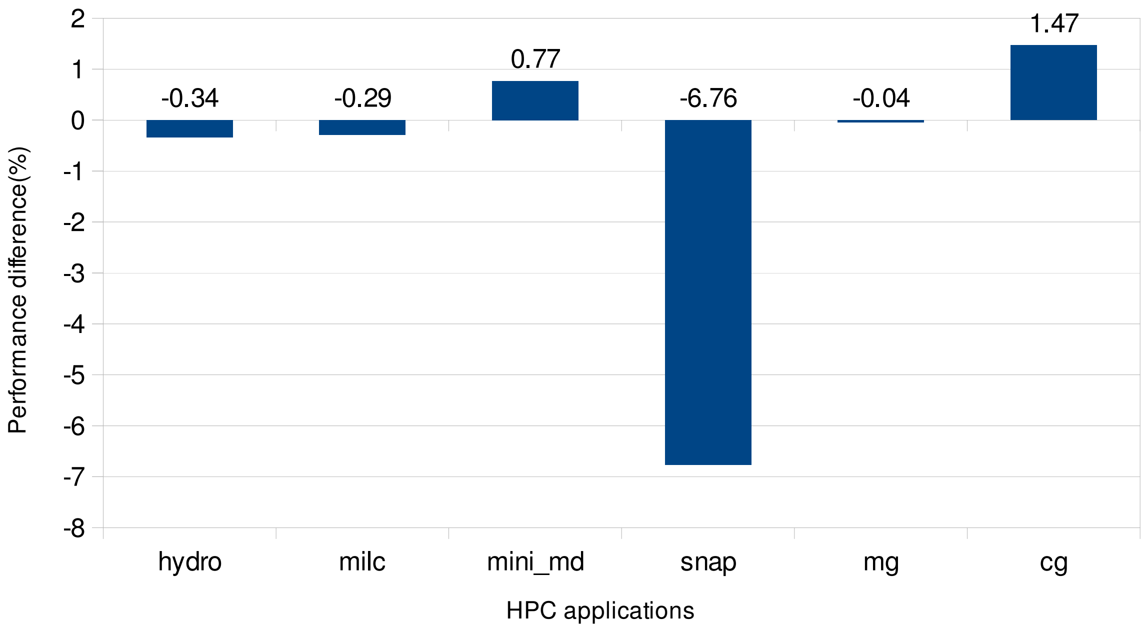
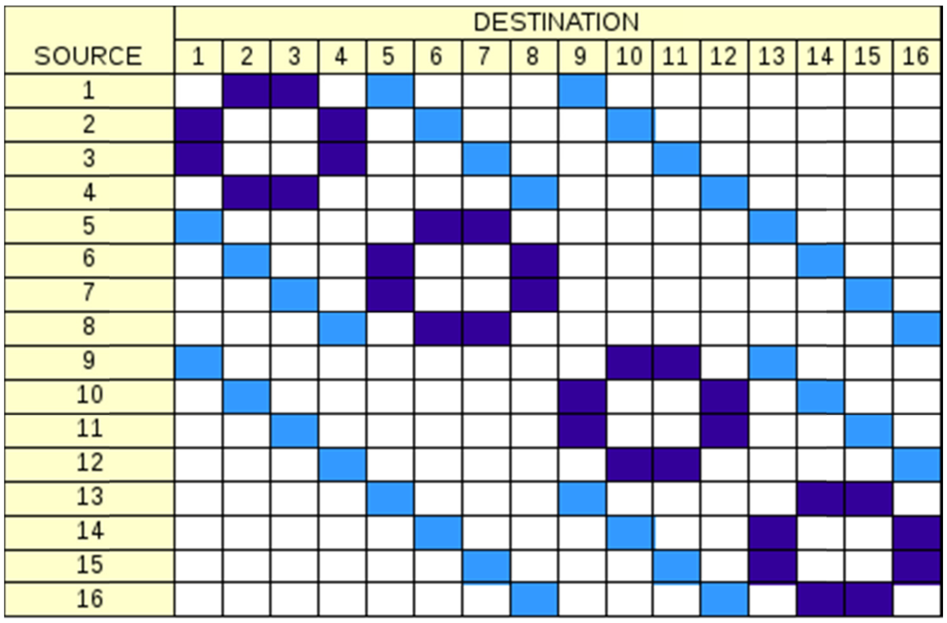


Figure 19: Simulation results – performance differences

3.1.4. Impact of mapping

Grouping of servers could decrease the number of simultaneous messages, since a lot of messages between servers are moved into intra-rack communication and the number of messages in the OPS is expected to decrease. Communication of HPC application (*MILC*) is shown in Figure 20, where dark blue rectangles represent the traffic inside the racks, while the light blue ones represent the traffic that will be a candidate for the concurrency counting.



SOURCE	DESTINATION															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		🟪	🟪		🟦				🟦							
2	🟪			🟪		🟦				🟦						
3	🟪			🟪			🟦				🟦					
4		🟪	🟪					🟦				🟦				
5	🟦					🟪	🟪						🟦			
6		🟦			🟪			🟪						🟦		
7			🟦		🟪			🟪							🟦	
8				🟦		🟪	🟪									🟦
9	🟦									🟪	🟪		🟦			
10		🟦							🟪			🟪		🟦		
11			🟦						🟪			🟪			🟦	
12				🟦						🟪	🟪					🟦
13					🟦				🟦					🟪	🟪	
14						🟦				🟦			🟪			🟪
15							🟦				🟦		🟪			🟪
16								🟦				🟦		🟪	🟪	

Figure 20: MILC – mapping 4 servers per rack

Traffic marked with dark blue colour is more frequent so we had put those servers in the same rack and the concurrency decreased. The communication of other HPC applications is similar, so we used the same grouping for all of them. Figure 21: Average number of concurrent messages - different mapping shows the average number of concurrent messages in regards to the number of servers per rack. No matter how we pack the servers, there will always be certain level of concurrency. Anyway, the number of collisions will always depend on mapping. Figure 22 shows how collisions impact the application performance.

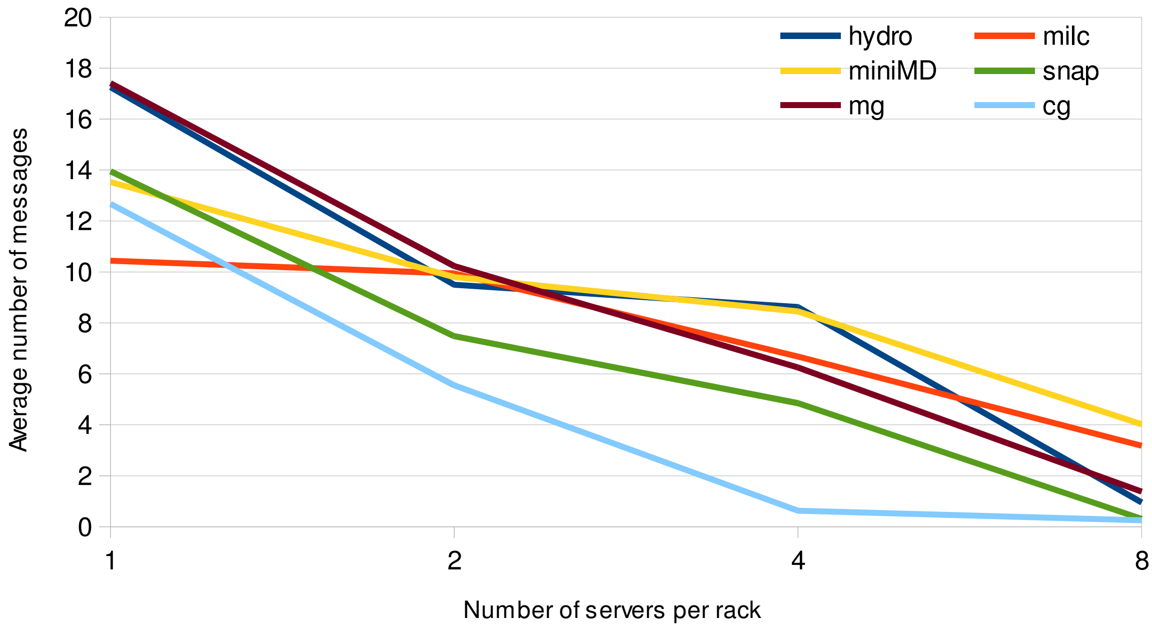


Figure 21: Average number of concurrent messages - different mapping

The performance differences are noticeably low. Mapping has the biggest impact on *SNAP* application where increasing the number of servers per rack gives much better performance. Section 3.2 will show if these results could be even better for *SNAP* and other HPC applications.

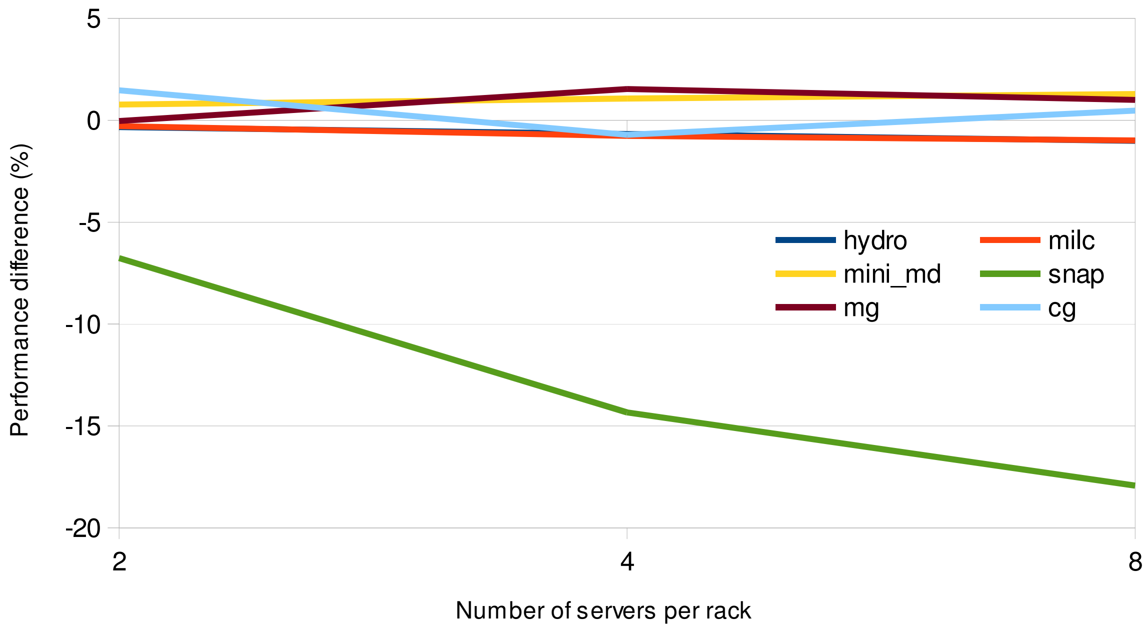


Figure 22: Performance difference – different mapping

3.2. OCS suitability

LIGHTNESS all-optical switching is based on Optical Circuit Switching (OCS) and Optical Packet Switching (OPS). When analysing latency sensitive applications we should count on disadvantage of OCS switches which have a latency measured in milliseconds. This latency comes from configuration time used to set up the internal light paths. In this section we investigate execution of latency sensitive HPC applications and Big Data applications through the OCS.

We analyse the impact of OCS on the performance of different applications. We have developed a characterization methodology to count the reconfigurations that occur during one application run. For the default mapping we used one server per rack and 16 racks connected to OCS switch (

Figure 23). For Big Data applications the number of servers goes to 35. Each rack is connected with two ports to OCS switch, one port is used for outgoing and the other for incoming traffic.

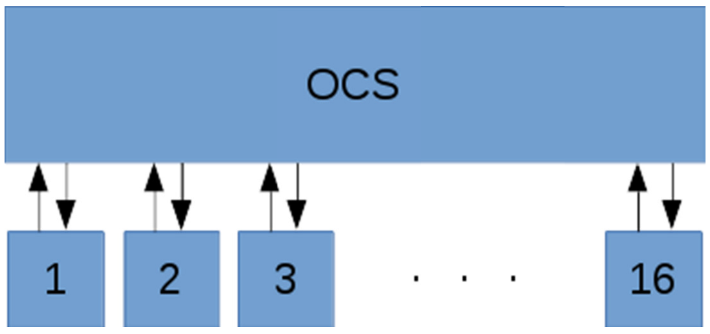


Figure 23: Default mapping of 16 servers and OCS switch

A source server (SRC) can send a message to another server only if the path is established. Once the path is established, the SRC transmits all the packets that form one message without additional latency. When server finishes the transport of all containing packets of one message, this path could be set down in two ways. First way is that SRC wants to send a message to another destination and it has to release[2] his last sending path (black arrow – Figure 24a) and make a new one (red arrow – Figure 24a). Second way is that another server wants to communicate with SRC (Figure 24b).

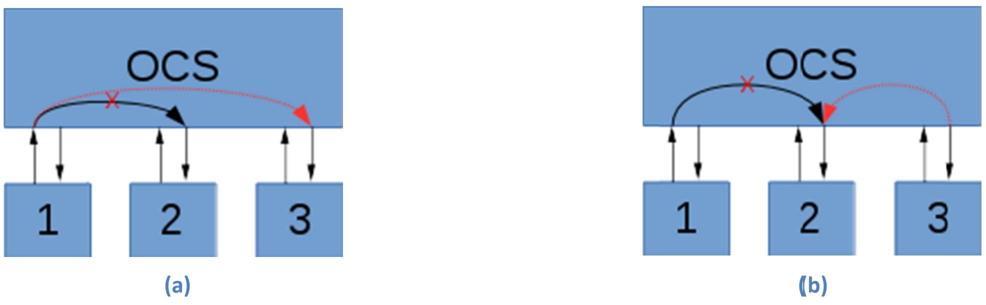


Figure 24: OCS path tear down

3.2.1. Characterization methodology

The characterization methodology shown in Figure 25 counts the number of paths that one SRC should create during one iteration. A higher number of connections imply larger latency and higher performance degradation.

The input (*Trace*) is obtained from Extrae [2] trace file, where every *msg* represents the source and destination of a sending/receiving event. *Trace* represents the collection of all messages. *Source* parameter represents the source rack and *destination* parameter represents the destination rack. *Source_list* contains the last destination of each source. *Dest_list* keeps the last source of each destination. This way we can notice the path changes on both sides, source and destination (Figure 24). Variable *Changes1* represents changes from the Figure 24a and *Changes2* represents changes from the Figure 24b. After checking all the messages, summarized changes are divided with total number of messages.

[2] More accurately: creating and releasing the path is not done by SRC, it just triggers these actions. The SDN control plane (generally the management functions of the DC) is responsible for creating and releasing the paths in network.

```
Changes = 0
message_number = 0
Sum = 0
for msg in Trace do
    if source != destination then
        if Source_list[source] != destination then
Changes1 [source] + +
        elif Dest_list[destination] != source then
Changes2 [source] + +
        end
    end
    Source_list[source] = destination
    Dest_list[destination] = source
    message_number + +
end

for rack in Nracks do
    TotalChanges[rack] = Changes1[rack] + Changes2[rack]
    Sum = Sum+ TotalChanges[rack]
    Average_changes_rack = Sum/Nracks
    Average_changes = (Changes1 + Changes2)/message_number
```

Figure 25: Algorithm to quantify OCS configurations

Applications that have different message sizes, but the same percentage of OCS changes and number of messages, will experience different impact on their performance (Figure 26). The application that has larger messages executes longer comparing to the execution of an application with smaller messages, because the transfer of larger messages in the same network takes more time. Therefore, the applications with larger messages will have lower performance degradation because the latency of creating the OCS path will take the same amount of time for both applications. For example, we counted that the computation lasts the same for both applications. In Figure 26, the ratio of the execution time without OCS changes and the execution time with OCS change of each message is higher for the application that has larger messages. The same latency causes higher performance impact for the applications with smaller messages.

The relation between three values, OCS changes percentage, number of messages and message size will have the impact on overall performance degradation of the whole application.

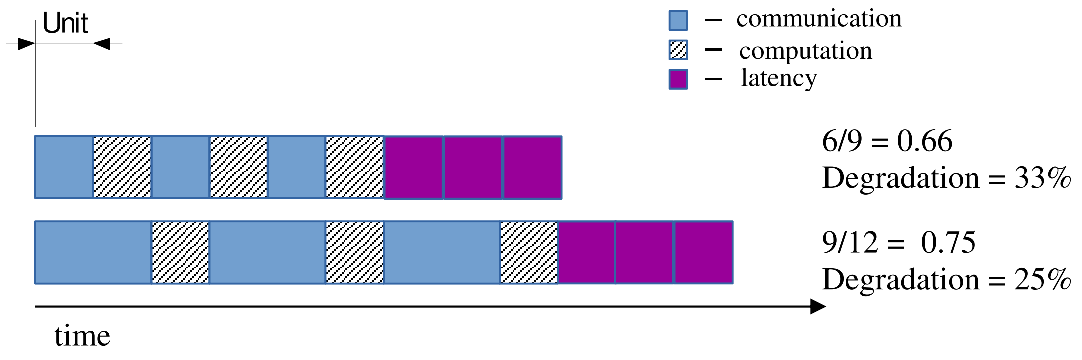


Figure 26: Degradation - different message sizes

Percentage of messages that demands setting of new OCS path is shown in Figure 27. For every HPC application this percentage is quite high (above 60%). Blue bars show the changes that are demanded by the sending server (SRC) because that server wants to change the destination by itself. Red barss represent the changes caused by another server which has broken the previous SRC's path.

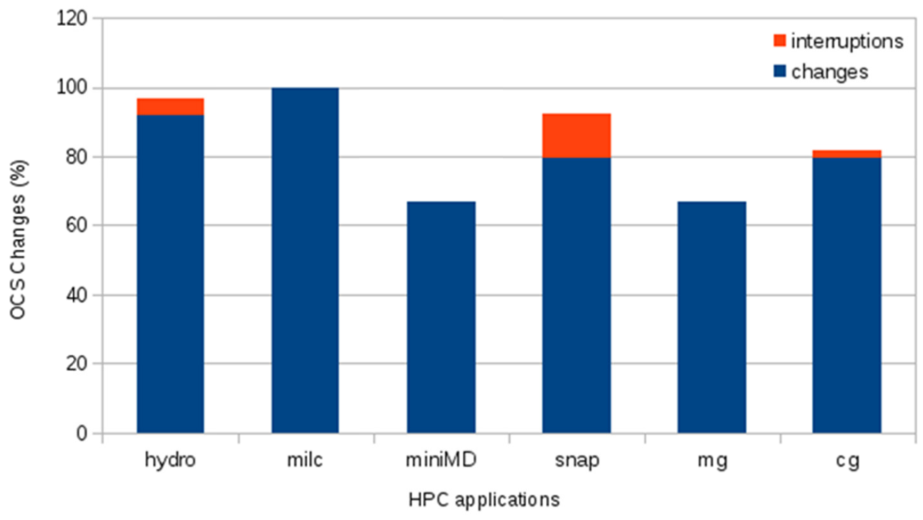


Figure 27: Percentage of transferred messages that demand OCS change

Figure 28 shows the percentage for Big Data traces. Comparing to the HPC applications percentage is more than twofold lower. Nevertheless, here we have much more interruptions that require OCS paths releases. We will explore how to combine the servers so the applications require less OCS path changes.

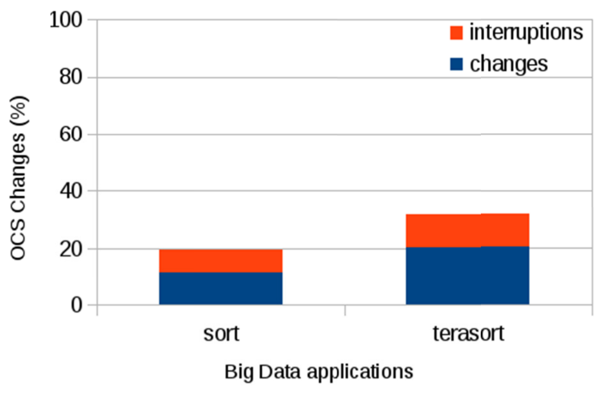


Figure 28: Percentage of transferred messages that demand OCS change

In summary, OCS could significantly impact the execution of HPC, but also the execution of Big Data applications. The potential impact is shown in Table 5: Potential OCS impact.

Application	Potential impact
HYDRO	*High*
MILC	*High*
MINIMD	*High*
SNAP	*High*
MG	*High*
CG	*High*
SORT	*Medium*
TERASORT	*Medium*

Table 5: Potential OCS impact

Understanding the way how one application behaves regarding the timings of sending and receiving, and how often it changes the established path, is very important in order to predict whether an application will be sensitive to OCS latency. Figure 29 shows the results of simulations and we could see that applications are extremely sensitive to OCS latency. For comparison, we used default mapping of OPS which includes one server per rack with no possibility of collisions (Figure 30a). For OCS we used also one server per rack and two ports per one rack, one port for incoming and the other for outgoing traffic (Figure 30b).

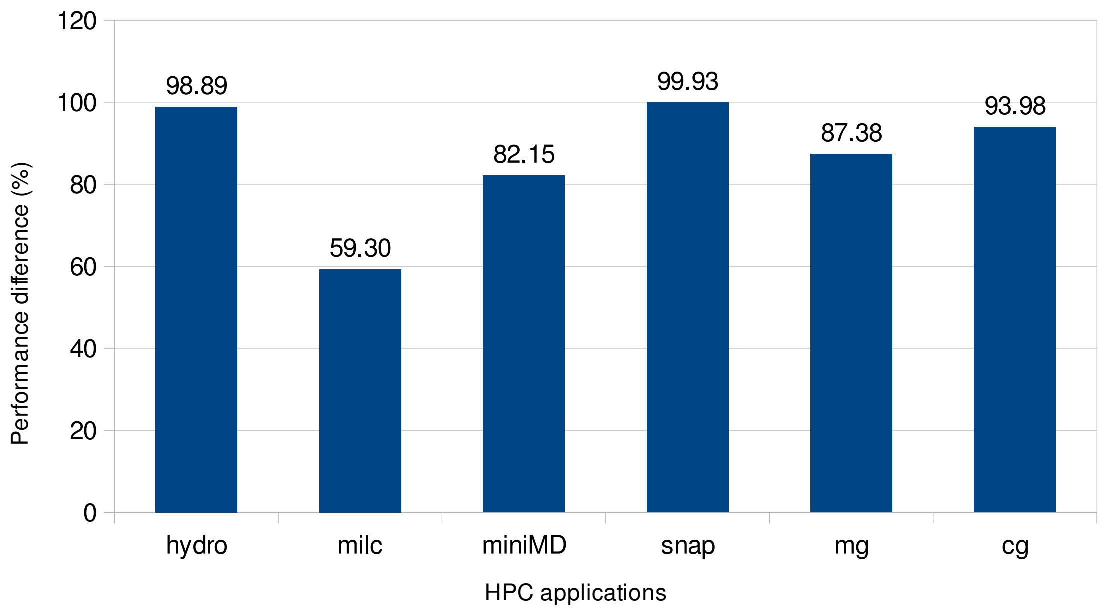
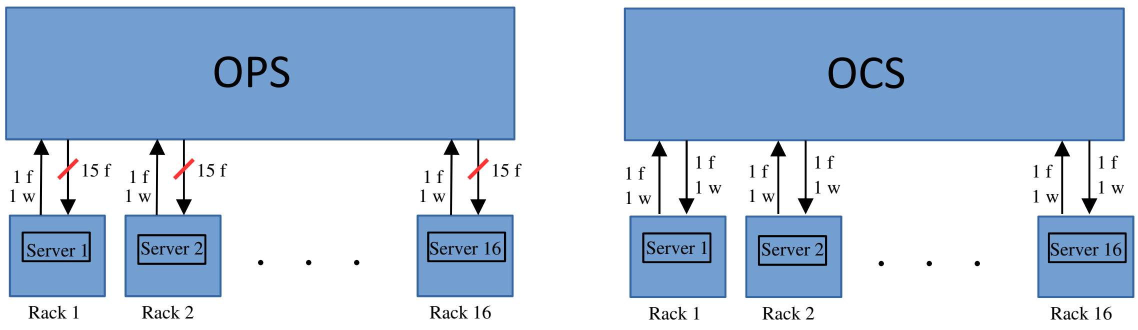


Figure 29: Performance difference – OCS change



(a)

(b)

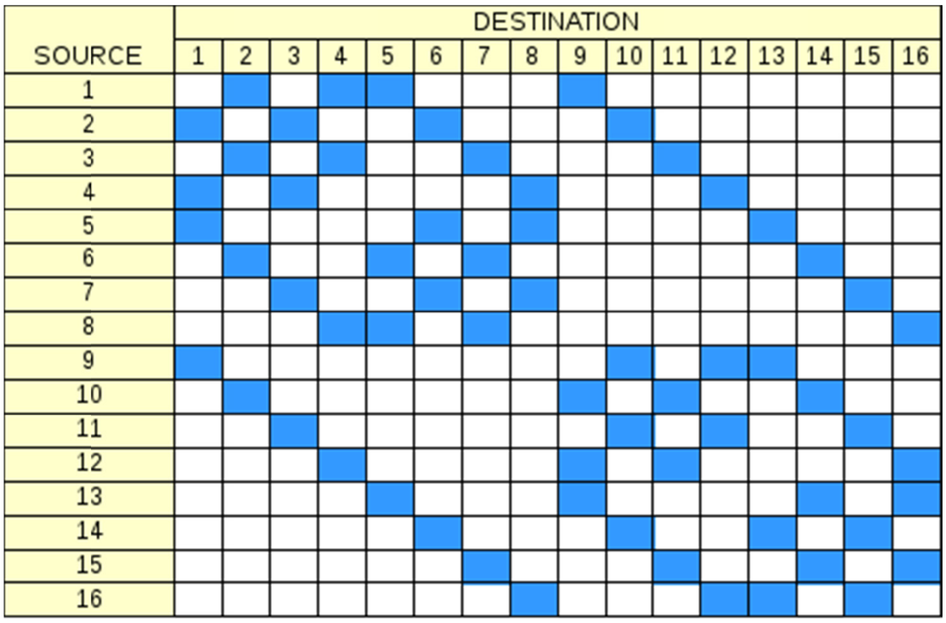
Figure 30: Default OPS mapping and default OCS mapping

In the next section we decreased the number of ports and increased the number of servers per rack, so we could see the impact of different mapping.

3.2.2. Impact of mapping

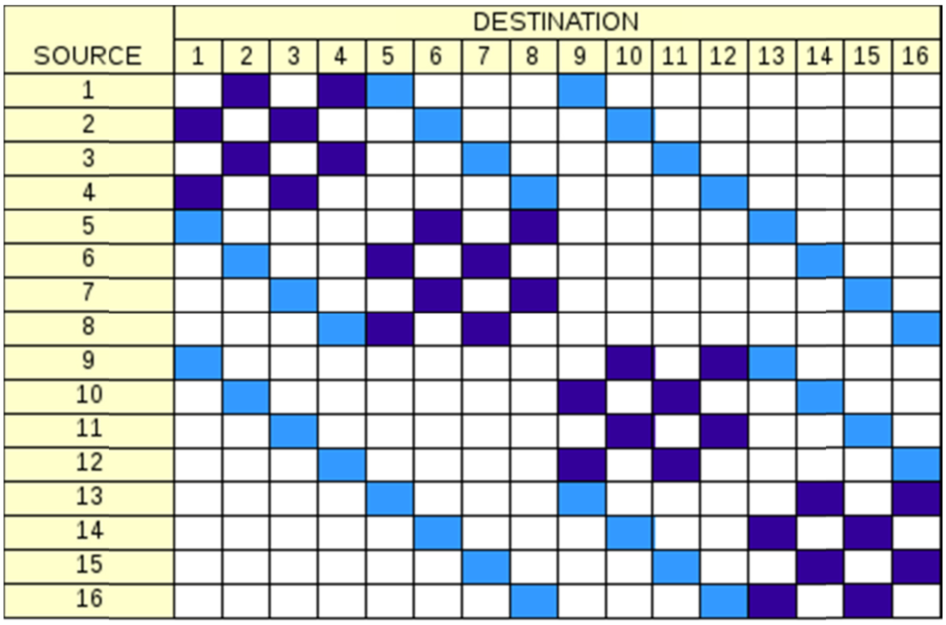
We started from the default OCS mapping of 16 servers in 16 racks (1 server per rack) and 2 ports per rack. While increasing the number of servers per rack we grouped the servers that communicate frequently between each other and this way we minimized the need for creating new OCS paths.

Figure 31 shows the communication of one of the HPC applications, *MINI_MD*. The communication of other HPC applications is similarly grouped. The servers that communicate frequently with each other have been combined in racks. One of these combinations is in Figure 32. Dark blue colour indicates the communication inside the rack.



SOURCE	DESTINATION															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		🟦		🟦	🟦				🟦							
2	🟦		🟦			🟦				🟦						
3		🟦		🟦			🟦				🟦					
4	🟦		🟦					🟦				🟦				
5	🟦					🟦		🟦					🟦			
6		🟦			🟦		🟦							🟦		
7			🟦			🟦		🟦							🟦	
8				🟦	🟦		🟦									🟦
9	🟦									🟦		🟦	🟦			
10		🟦							🟦		🟦			🟦		
11			🟦							🟦		🟦			🟦	
12				🟦					🟦		🟦					🟦
13					🟦				🟦					🟦		🟦
14						🟦				🟦			🟦		🟦	
15							🟦				🟦			🟦		🟦
16								🟦				🟦	🟦		🟦	

Figure 31: MINI_MD communication between servers



SOURCE	DESTINATION 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		🟪		🟪	🟦				🟦							
2	🟪		🟪			🟦				🟦						
3		🟪		🟪			🟦				🟦					
4	🟪		🟪					🟦				🟦				
5	🟦					🟪		🟪					🟦			
6		🟦			🟪		🟪							🟦		
7			🟦			🟪		🟪							🟦	
8				🟦	🟪		🟪									🟦
9	🟦									🟪		🟪	🟦			
10		🟦							🟪		🟪			🟦		
11			🟦							🟪		🟪			🟦	
12				🟦					🟪		🟪					🟦
13					🟦				🟦					🟪		🟪
14						🟦				🟦			🟪		🟪	
15							🟦				🟦			🟪		🟪
16								🟦				🟦	🟪		🟪	

Figure 32: MINI_MD mapping 4 servers per rack

With this server grouping new interruptions could appear, but the great amount of traffic becomes intra-rack. Like this, the interruptions don't have that much of an impact. Still every rack uses two ports, one for incoming and the other for outgoing traffic. In Figure 33 it can be seen that the percentage of needed OCS path settings decreases. For example for SNAP application four servers per rack is mapping that cause almost no need for OCS path setting up.

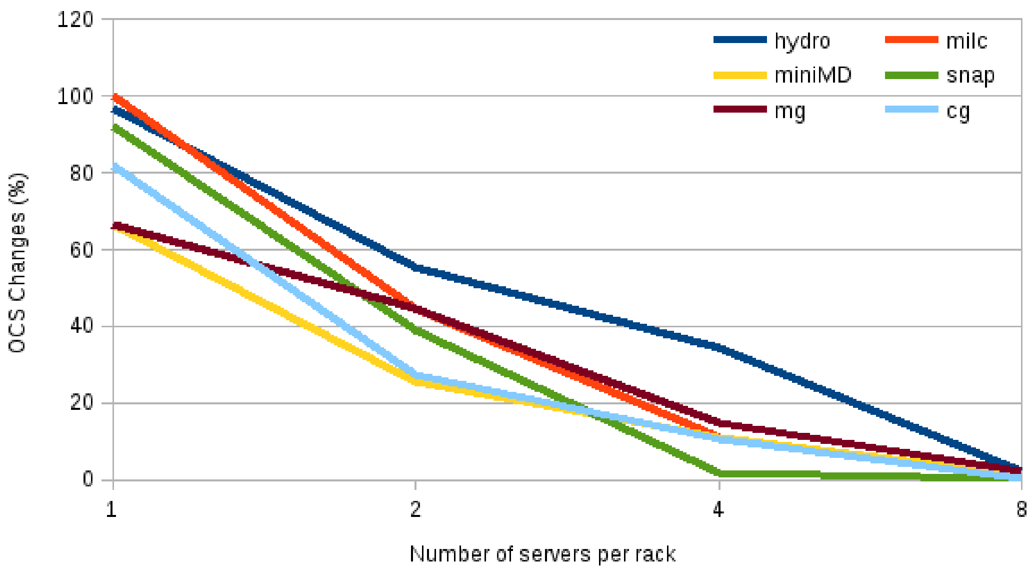
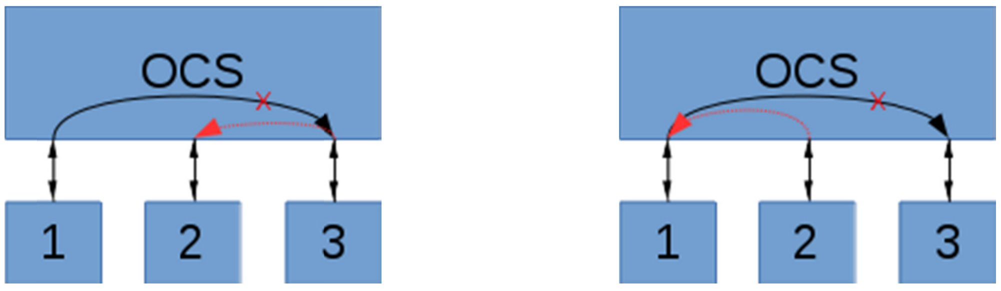


Figure 33: OCS Changes - 2 ports per rack

Thinking of resource cost reduction just one port per rack can be used for incoming and outgoing traffic. Using one port, path between source server (SRC) and destination server will be set in both directions at once. Anyway, this type of connection will have more different possibilities to be released, because one port now represents source and destination at the same time.



(a) (b)

Figure 34: OCS path tear down

Figure 34 shows two different possibilities for releasing the path, beside the two that we've already described (Figure 24). The first way is to tear down the path from the destination server side (Figure 34a). The path between source server (server 1) and destination server (server 3) is formed because server 1 had sent the message to server 3. If server 3 wants to send the message to a server different from server 1, the path 1 → 3 needs to be released, because server 3 has just one port for incoming and outgoing traffic. The other way of tearing down the path is shown in Figure 34b. Again there is a path created between server 1 and server 3. If another server (server 2) wants to send a message to server 1 the path 1 → 3 needs to be torn down.

With this mapping and lowered cost we got results on the Figure 35.

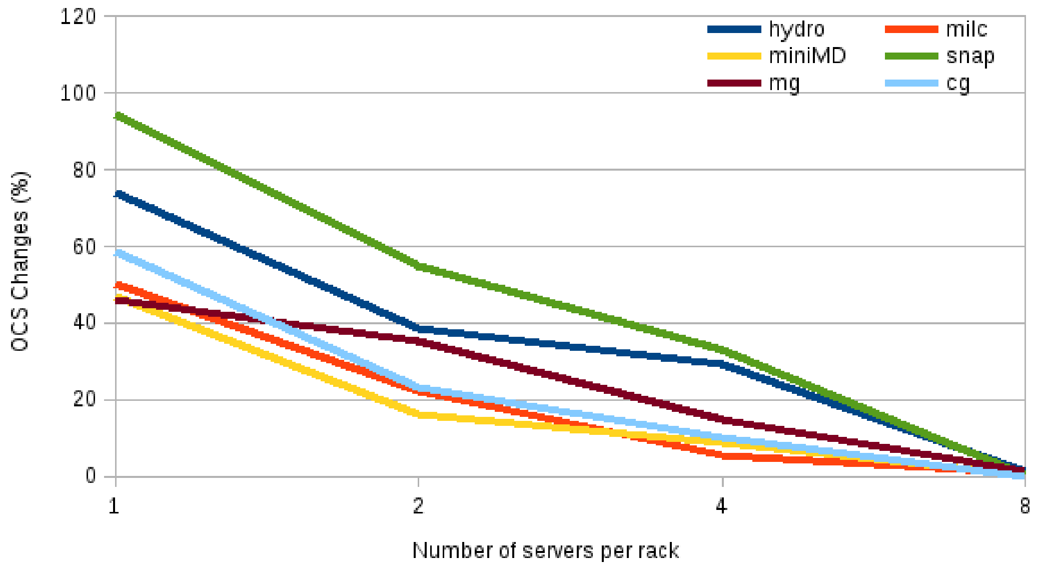


Figure 35: OCS Changes - 1 port per rack

These results show that server grouping reduces the need for OCS path creation, but if we use one port instead of two, reduce for almost all applications is even higher. This means that with creating of one path between sender and receiver we create the path that will be used in both directions, so there is no need to create another path in the opposite direction. In the example of the *MILC* the percentage of creating OCS paths when using two ports per rack (server) compared with one port per rack goes form 100% to 50%. Just for the *SNAP* application this percentage increases when one port per rack is used. Creating two-direction path breaks some paths that need to be created again, so for *SNAP* the optimal approach is to use two ports per rack.

For the Big Data traces we also want to try to group several servers in one rack, but when looking at their communication it seems much harder than it was for the HPC applications. Figure 36 shows the communication of *Minerva SORT* application. Grouping them as HPC applications will not significantly decrease the percentage of OCS changes. We grouped them differently so the grouping has an influence in decrease of OCS changes percentage (Figure 37). In two racks (nine servers per rack) we put the most frequent communication (orange and yellow) and the rest (grey) is the communication that will be exchanged with the servers from two other racks. Like this we got the results in Figure 38. Similar principle is used for five servers per rack. *Minerva TERASORT* application needs 31 server and the racks contain four, eight or 16 servers per rack.

Like with HPC applications, using one port per rack shows even better results regarding OCS and we could also minimize the resource cost of OCS ports and ports per rack (Figure 39).

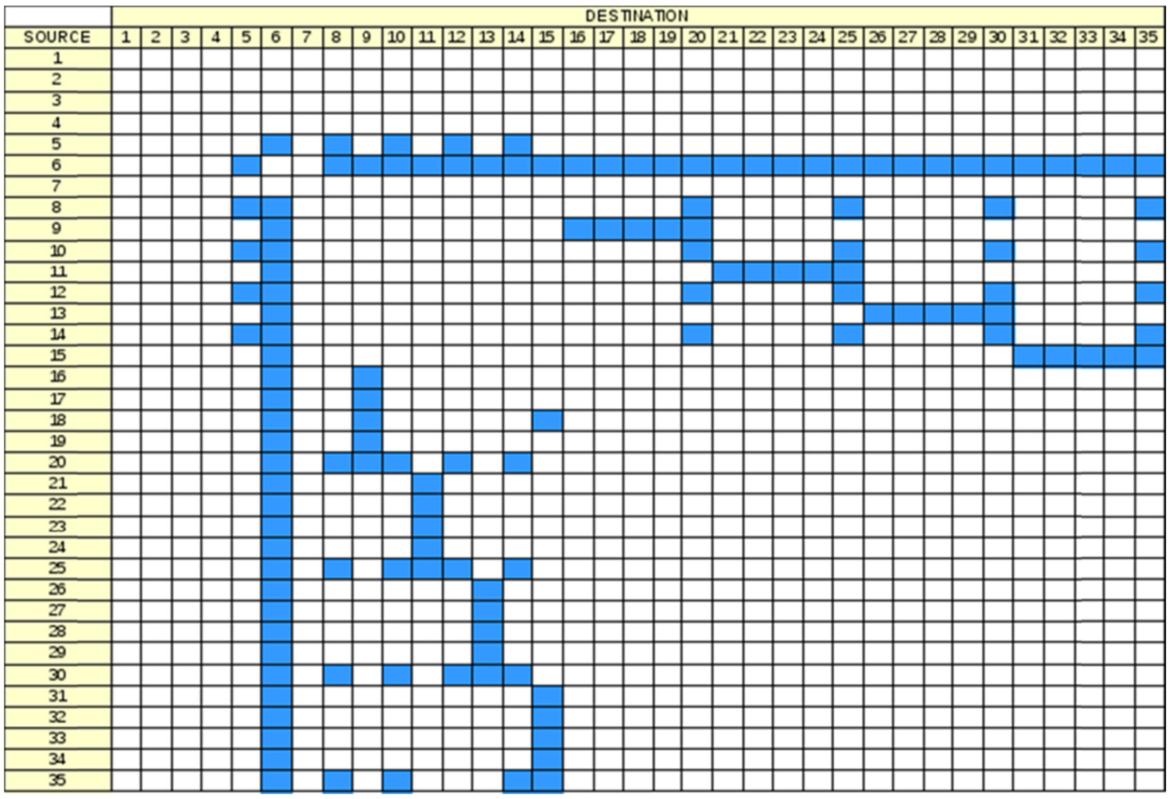


Figure 36: Communication of Minerva SORT application

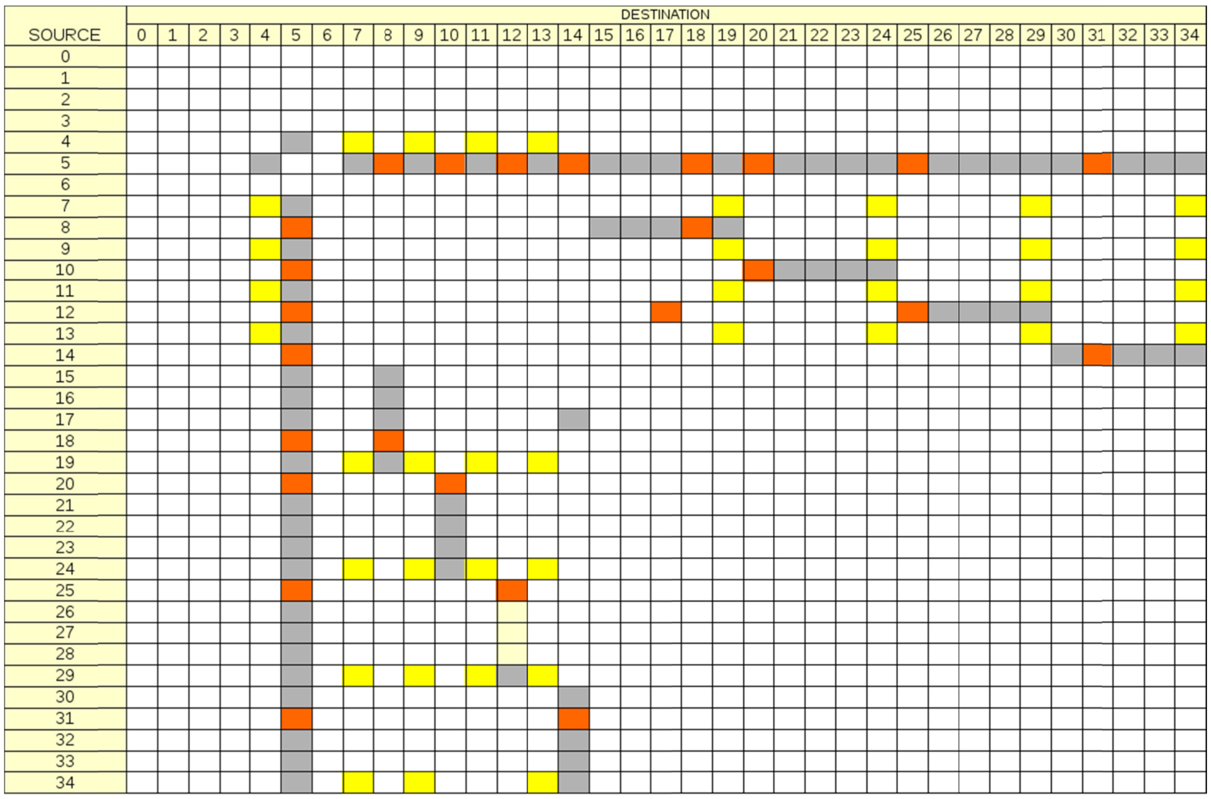


Figure 37: Minerva SORT mapping 9 servers per rack

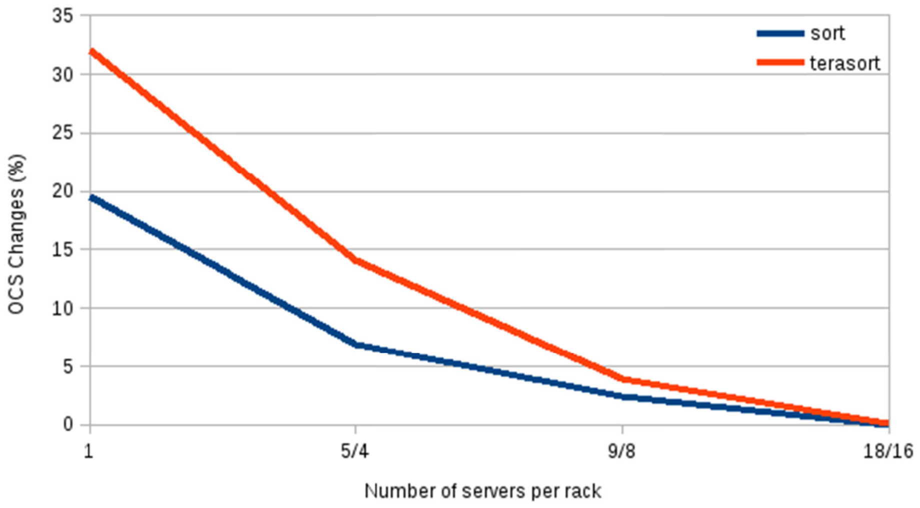


Figure 38: OCS Changes - 2 ports per rack

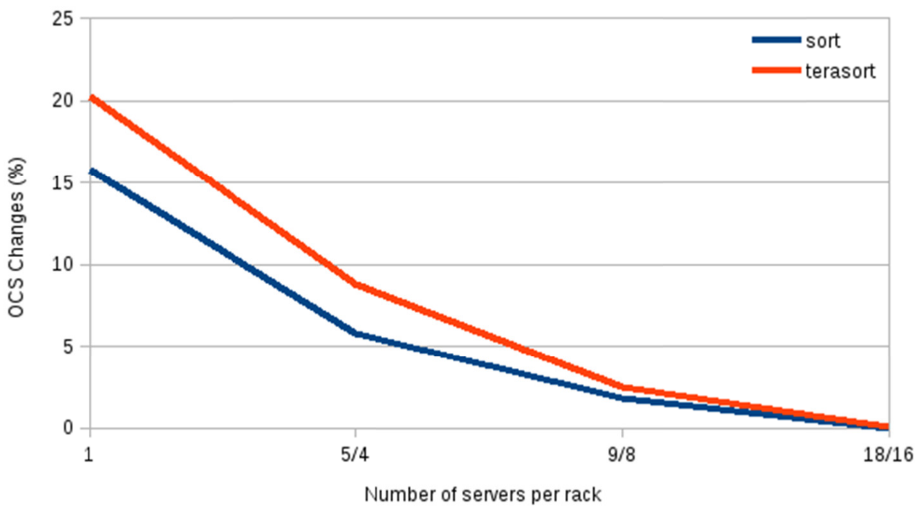


Figure 39: OCS Changes - 1 port per rack

Using this way of combining servers we've got less than 3% of OCS path setting when we placed eight or nine servers in one rack. The cost is also reduced because we used just one port per rack. Anyway, this percentage is still high, because the cost of just one OCS path setting is 25ms. Carefully arranging which servers should be in which rack could lead to really low number of needed OCS path setting.

3.3. Big Data applications sensitivity to network performance

3.3.1. Technical Background: Hadoop

Hadoop is the most popular open source implementation of the MapReduce paradigm. Developed initially by Yahoo in 2005, it was released and migrated under the supervision of the Apache Software Foundation, which currently is the organization responsible for its management and development (with input from many different companies).

Hadoop architecture consists of two main modules: Hadoop MapReduce, which is the framework that implements the MapReduce computing paradigm, and HDFS (Hadoop Distributed File System), which is a distributed file system to store all data to be processed.

The MapReduce paradigm was introduced by Google in 2004 in response to the need to store and process all the data needed to build an Internet scale search index. The main idea behind this paradigm is to use a set of machines that form a distributed system, and divide the work into small tasks that can be performed for each machine separately.

The name comes from the two phases that the paradigm implements: Map and Reduce. In the first phase, Map, all the input data have been divided into small blocks of data. Each of these is assigned to a node, which performs the function map () over an input block data, and the result is a set of key-value pairs. The set of all these key-value pairs generated by the Map phase become the input to the next phase, called Reduce. The paradigm states that the output of the Map phase is ordered for each key/value pair, which means that the function Reduce () receive as input a key, and the list of all values associated to the same key. Finally, the Reduce function () processes this list and as a result generates part of the application output. The figure below represents an example of the described behavior.

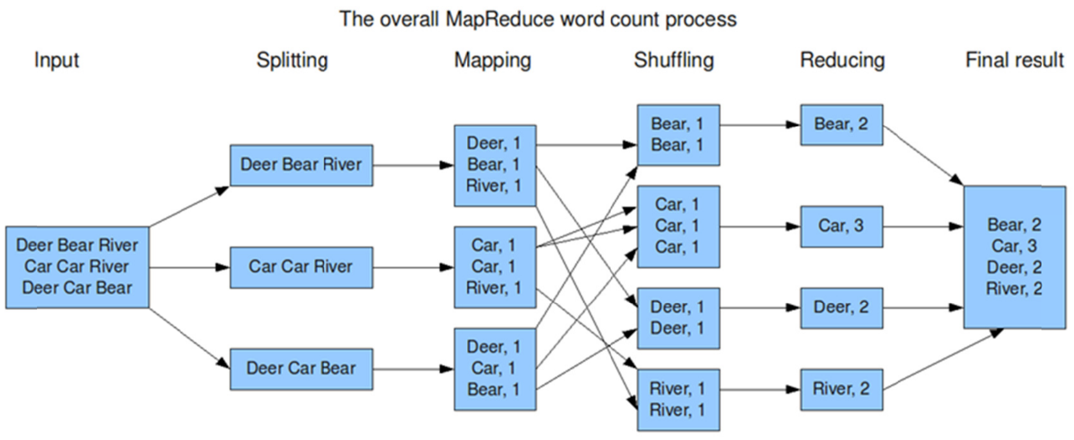


Figure 40: MapReduce Workflow

The major advantage of this paradigm is that the execution of all functions of the Map phase can be performed in parallel on different nodes simultaneously. This is because the input is divided into separate blocks, and there is no dependence between inputs and outputs of this phase. The same goes for the execution of all functions of the phase Reduce, with the only difference that must be completed entirely Map phase to start running Reduce phase.

HDFS: Hadoop Distributed File System

HDFS is the file system used in Hadoop to store the data to be processed, as well as the results of an application. The distributed filesystem is built on top of the native OS filesystems available in each node of the cluster. The whole filesystem is managed by a set of daemons being run across the cluster nodes. In particular, the process of Hadoop responsible for maintaining the HDFS structure is called NameNode and is unique throughout the cluster.

To store large files that would not fit on a single node, HDFS divides data into blocks that are distributed across the nodes in the cluster. The process DataNode, running in each slave node in the cluster, is responsible for transferring and storing data blocks allocated locally. The NameNode maintains the structure and organization of the system files and the location of all blocks.

Hadoop MapReduce

Hadoop MapReduce is a framework that implements the paradigm MapReduce. Similar to the architecture of HDFS, it consists of two different processes: the JobTracker, which is unique in the cluster runs on the master node, and TaskTracker, which runs on each slave node.

When you want to run an application on Hadoop MapReduce, it needs to be packaged as a Hadoop job that is submitted to the JobTracker process. The job is divided into smaller units of work, called tasks, which are assigned and execute TaskTrackers. The JobTracker is responsible for assigning the units of work (maps and reduce operations) to the TaskTrackers running on different slave nodes.

Each Tasktracker can be configured to provide a specific number of execution slots in parallel to accommodate map and reduce operations. A slot corresponds to a process Mapper (or Reducer) runs the Map tasks (Reduce or tasks) assigned to it. This allows for a better use of the resources of nodes by running several tasks in parallel.

3.3.2. Instrumentation and Tracing environment

The information usually available to monitor the performance of Big Data frameworks is high level, and provides limited hints on the root causes of performance degradation. It is usually limited because provides little information about how Hadoop is behaving internally. For this reason, in LIGHTNESS we decided to develop a deep instrumentation environment for Hadoop as a proof of concept for other Big Data frameworks.

Tools: Paraver and Extrae

To implement the monitoring environment, we used two tools previously developed at the BSC: Extrae and Paraver. These tools are used to analyze the performance of parallel HPC applications. Unlike HPC environments, most of the communications in Hadoop frameworks are initated by the frameworks themselves, and not usually by the application code developed by the users.

The monitoring environment achieves its goals by following three different phases of instrumentation. The first step is done during the execution of the application to be analysed. Extrae is used to intercept certain calls and thus know the status of the application runtime, providing detailed information of the progress being done by a job in execution. Once the execution of the application is completed, Extrae generates a binary tracefile per process that has been monitored. In a second step, all generated binary files are merged to create a single file, which contains all the information related to various processes and unified. Finally, the last step is the visualization and analysis of trace generated by Paraver.

The method used in this project to integrate Extrae with Hadoop is to load a shared library item (libseqtrace provided by the same Extrae) when starting the Hadoop framework, allowing the interception functions to perform the detailed instrumentation.

The Extrae API comprises several functions, the most important are:

• Extrae_init (): initializes the library and the tracing process.

• Extrae_fini (): shuts down the tracing process and initiates the post-processing.

• Extrae_event (type, value): generates a user event with the specified type and value.

• Extrae_nevent (count, types, values): function as above, but in this case can issue several synchronized events on the same call. The advantage is that all events share the same timestamp, and therefore considered to have been generated at the same instant of time.

The main problem when using Extrae with Hadoop is that while the first is written in C language, Hadoop is written in Java. This makes it impossible to call the functions explained directly from Hadoop code. To address this problem we used a JNI (Java Native Interface), wrapper that allows to call native code and libraries written in other languages (C, C ++, assembler, etc.) from the same Java virtual machine.

Several new Extrae events were defined to capture information about different aspects of the Hadoop framework in execution. As an example, below is reported the list of events used to register the different Hadoop daemons (JobTracker, NameNode, SecondaryNameNode, TaskTracker and DataNode) in Extrae. Calls to these methods were inserted in the Hadoop runtime.

```
package es.bsc.tools.extrae;
public class IDManager
{
  public static void registerJobTracker() {
    Wrapper.Event(88881, 1);
  }
  public static void registerNamenode() {
    Wrapper.Event(88881, 2);
  }
  public static void registerSecondaryNamenode() {
    Wrapper.Event(88881, 3);
  }
  public static void registerTaskTracker() {
    Wrapper.Event(88881, 4);
  }
  public static void registerDatanode() {
    Wrapper.Event(88881, 5);
  }
  public static void registerTask() {
    Wrapper.Event(88881, 6);
  }
}
```

Figure 41 shows a window Paraver viewing a running Hadoop instrumented with Extrae, without even custom events have been reported so far. Each horizontal line corresponds to a different Java process. When the line is light blue, it means that the process has not yet been created or has been destroyed. This information is automatically collected in libseqtrace library.

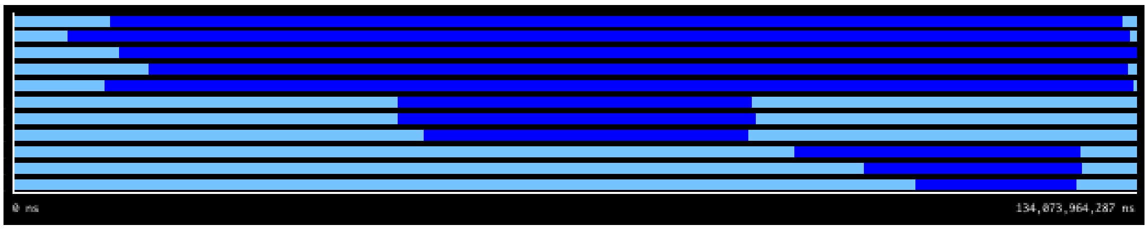


Figure 41: BigData activity trace

As the user events are not present, it is not possible to differentiate any details of the processing. Figure 42 shows another execution in which Extra events to identify the different daemons are already present. Each value of the event is assigned a different color, which identifies each daemon type.

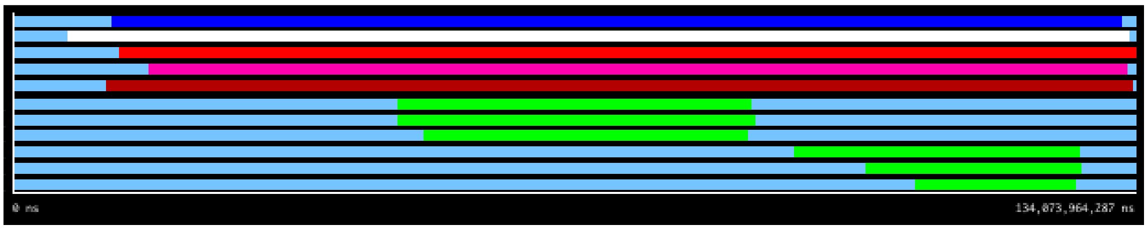


Figure 42: BigData activity trace, grouped by daemons

In this case the dark blue is JobTracker, the NameNode is white, red for SecondaryNameNode, TaskTracker is pink, the DataNode is dark red and finally green is associated to Map Reduce tasks.

To differentiate the stages in the Map and Reduce tasks, instead of using a single event to identify the type (as has been done with daemons) we used a different event for each state. This will allow to identify what the application is doing during the execution of each task. The following Figure 43 shows a scenario similar to the previous one but the inclusion of such events.

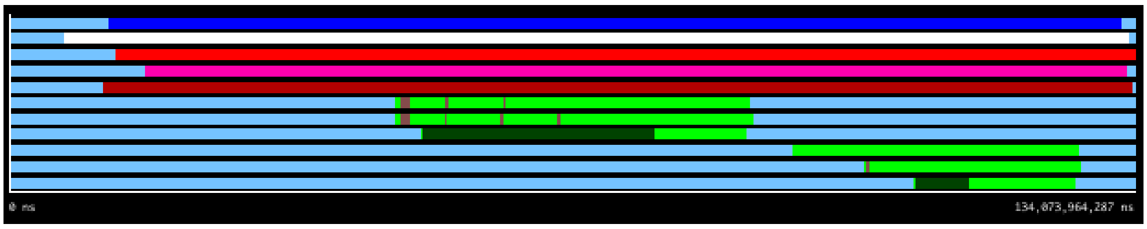


Figure 43: BigData trace, task distinction

As you can see now, more information has been added to the Map and Reduce phases. In particular, the brown color corresponds to different executions of Maps, and the dark green to Reducers. Thanks to the level of detail achieved with this instrumentation it is possible to observe how large are the runtime execution times compared to the user code execution, since the proportion of time running the application code (brown or dark green) with respect the total process (light green) is very low.

The Reduce phase is very long compared to Map, but this is because it actually consists of three phases: Copy, Sort and Reduce. Shortly after mappers began to run, he Reduce phases starts with the copy sub-phase, which is in charge for continuously copying data from mappers to reducers as the various maps are in execution. Only after all maps are finished, the reduce sub-phase can be really initiated. To identify the phases in the trace, more events were added to the runtime.

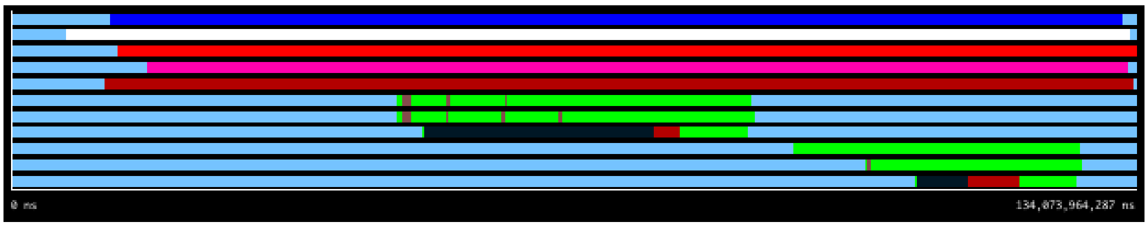


Figure 44: BigData trace, task activities

Unlike the previous capture, now the different stages of Reduce can be observed. Copy sub-phase corresponds to black, sort can be hardly seen because is very short, and finally the Reduce is shown.

Intercepting network communications

In order to capture communications between different processes, Extrae contains a set of API calls to store this information. Unfortunately, being mostly focused on MPI environments it is unhelpful for Hadoop mainly for two reasons. The first is that it should identify and modify all places where Hadoop initiates communications and add events there, what is unpractical. Unlike previously discussed events, which were modifications in specific locations with special significance, communications are widely spread across different places and using different code libraries (some are HTTP requests, some other RPC, etc.). The second reason is that Extrae assumes that all tasks in a job are static over time, but Hadoop creates tasks dynamically and therefore are unknown from the beginning.

To overcome these limitations, communications were tracked from outside of the runtime, and later on synchronized with other runtime events. For this purpose every node runs a sniffer task that is in charge to capture all TCP/IP packets sent and received through the NICs. On TaskTracker creation, the sniffer is automatically initiated.

The sniffer is implemented on top of libpcap, which is responsible for reading the packet headers at low level, and provides the following information:

- Source IP
- Source port
- Destination IP
- Destination port
- TCP Flags
- sequence number
- Number of ACK
- Payload (data content)

An important issue to address with this architecture is the fact that it is not obvious how to create effective yet efficient libpcap filters that shrink the amount of data to be capture to Hadoop traffic only. In practice the problem is to identify what open ports correspond to Hadoop services at the level of libpcap. To determine the relationship between open ports and PIDs we used the linux lsof tool. This command returns the list of currently open ports on the system and the PID of the process that has opened them. Because this tool introduces a significant overhead, it cannot be called for every packet capture to check the PID of the process associated to the communication. Instead, sampling was performed, in 1s intervals. Notice that since all processes created by Hadoop are registered, their PIDs are well known over execution. This information will be leveraged in a post-processing step to connect the captures with the source and destination processes in Hadoop. below is a Hadoop implementation of the captured communications between the various processes represented with yellow lines:

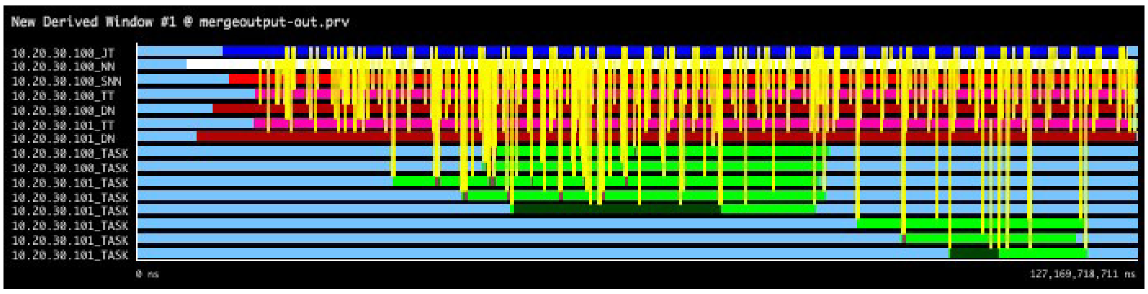


Figure 45: BigData trace, communication at high-level

The capture allows for detailed analysis of the communications between daemons, as the following Figure 46 of a TCP connection establishment sequence illustrates:

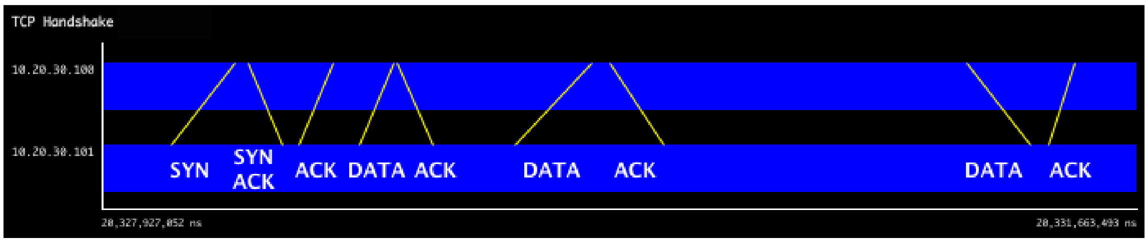


Figure 46: BigData trace, low-level view

3.3.3. Testing Environment

The tests have been run on 4 two-way 6-core Xeon E5-2620 0 @2.00GHz Linux boxes, for a total of 12 cores per node and 24 hardware threads because hyperthreading was enabled. Each machine was enabled with 64GB of RAM. All nodes were connected using GbE links to a non-blocking 48port Cisco 3750-X switch. For the software stack, all nodes were running Ubuntu 12.04LTS, and we used Hadoop 1.0.3 for the tests.

We picked Pagerank (500 pages), Sort (400MB) and Terasort (200MB) Benchmarks as described before to evaluate the projects developments as well as the network characteristics of some representative BigData workloads.

3.3.4. Workload characterization: PAGERANK

The PageRank application is composed of a set of jobs that are run serially until a given convergence condition is met. In this case, 500 pages were processed, looking for the creation of a reverse index to be used for searching content on the input dataset. The structure of the application can be seen in Figure 47, in which different waves of jobs can be seen (showing dark blue areas the time at which jobs are run). The execution of the different jobs happens at the same time across nodes, being initiated in each one of the 4 slave nodes. As it can be observed there is a large number of communications between the Hadoop daemons (top threads in the capture) and the actual tasks in execution.

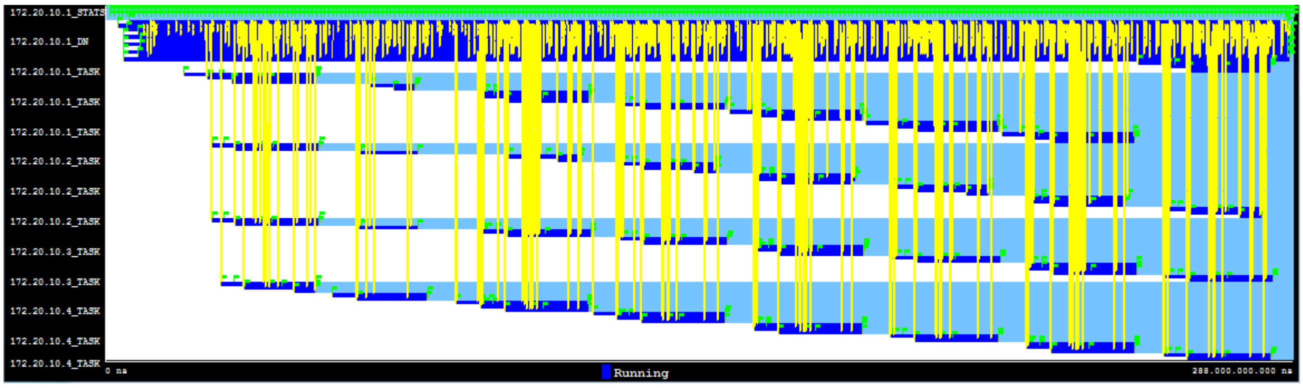


Figure 47: Pagerank, acitivity trace

Looking in more detail the trace, paying particular attention to the waves of map and reduce phases of all the jobs in Figure 48, it can be observed that in particular 7 jobs were run for this input dataset, with short map phases and a bit longer reduce phases. Notice also that both the map and reduce phase are completely parallel.

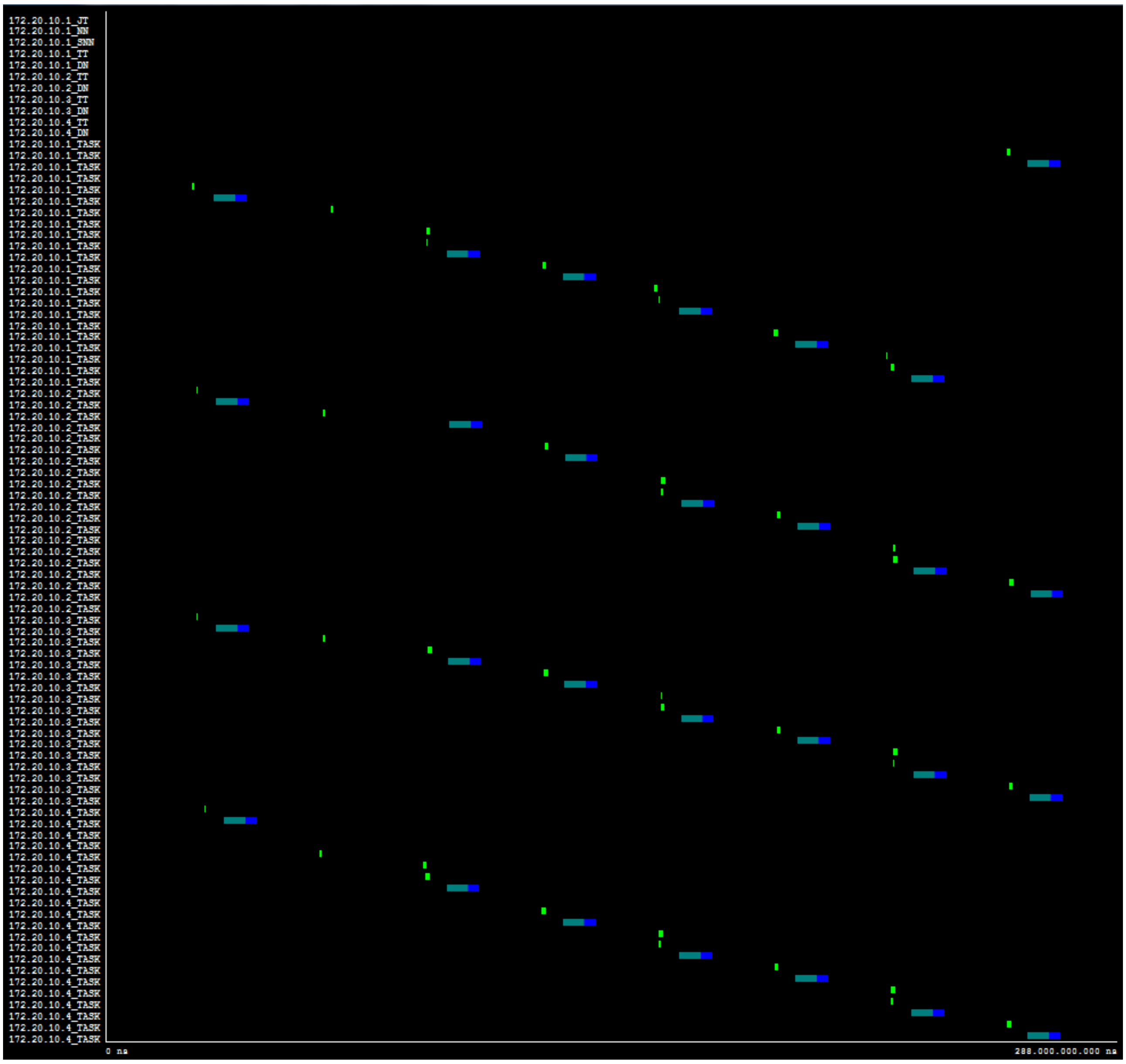


Figure 48: PageRank, Maps and Reduces

More interestingly, in the following Figure 49, we look at the distribution of data transfer sizes. The information is presented in two rows, the top row corresponds to the data produced for the set of Hadoop daemons, and the bottom row corresponds to the set of tasks. Darker colours represent higher frequencies, and lighter colours represent less frequent data sizes. Data transfer sizes are distributed from 0KB to 14KB, being the most frequent data transfers of less than 4KB. It is clear that the data transfers initiated by the slave processes are more uniform than the transfers initiated by the Hadoop daemons.

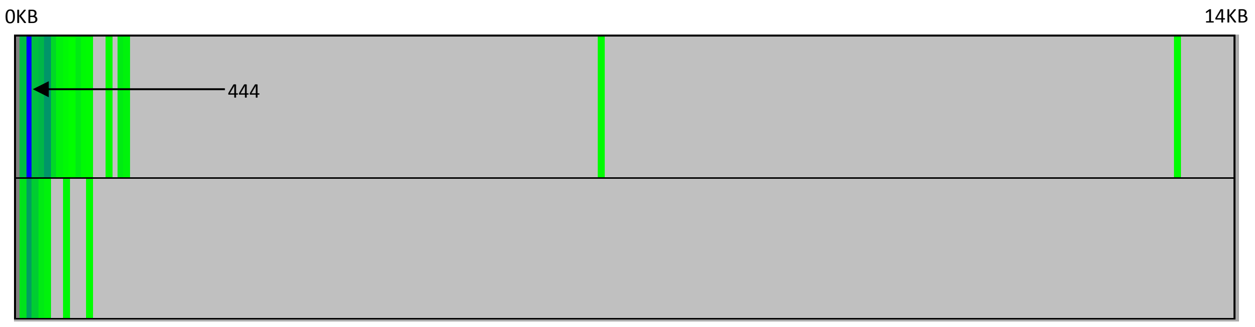


Figure 49: PageRank, histogram of data transfer sizes

Figure 50 represents a distribution of transfer latencies. Each row represents a process (either daemons or tasks) and each column represents a bin of latencies. The colour of each cell represents the frequency of data transfer latencies observed for the value represented by the cell. Darker colours represent higher frequencies, and lighter colours represent less frequent data sizes. Data transfer latencies are distributed from 0us to 410us, being the most frequent data transfers of less than 20us.

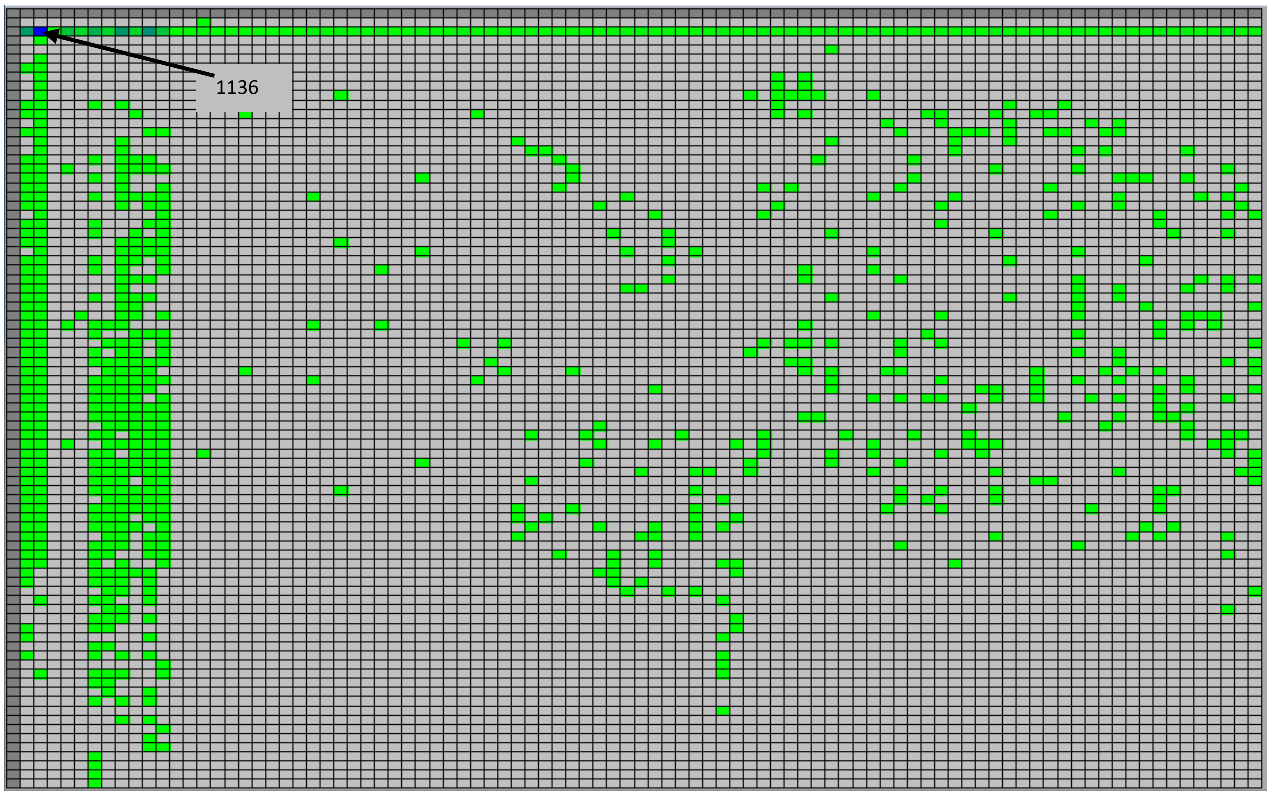


Figure 50: PageRank, histogram of data transfer latencies

3.3.5. Workload characterization: SORT

The Sort application is a single job application that works with the purpose to sort a large input text. For this experiment, we used a 200MB input that is sorted using the native Hadoop data ordering mechanisms, and therefore sorting is not done in user code. The structure of the application can be seen in the following Figure 51, different computing waves compute the provided input (showing dark blue areas the time at which activity takes place). The execution of all tasks happens at the same time across nodes, being initiated in each one of the 4 slave nodes. As it can be observed there is a large number of communications between the Hadoop daemons (top threads in the capture) and the actual tasks in execution.

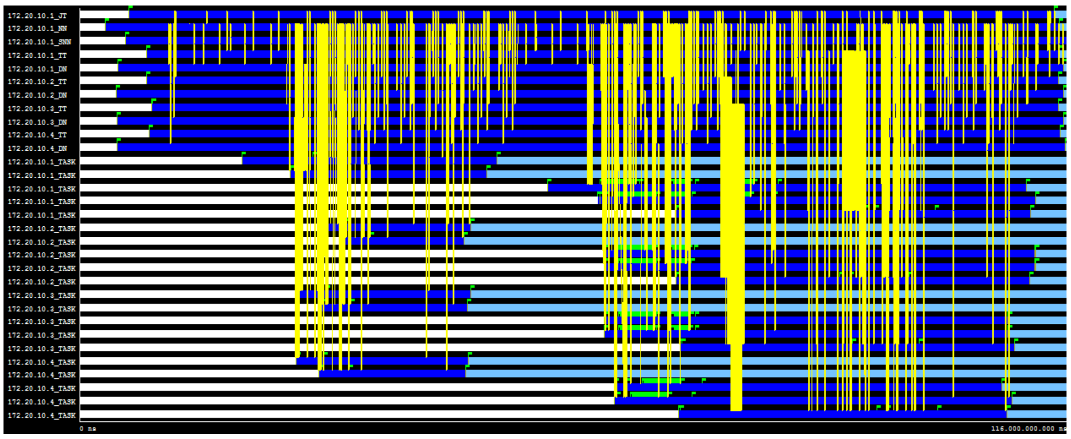


Figure 51: Sort, acitivity trace

Looking in more detail the trace as seen in Figure 52, it can be observed that this application has a simpler pattern than PageRank. It contains two waves of map phases (in green), which are basically identity assignments: that is, the input text is directly issued as the output of the map phase. Then the internal mechanisms of the Hadoop runtime perform the sorting, which is transparent for the user code. Finally, the reduce phase is run in parallel in all the nodes to produce the final output.

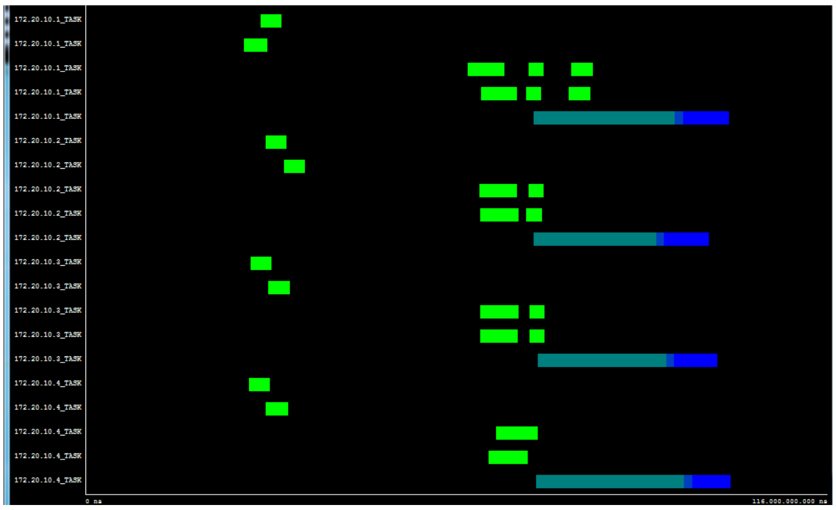


Figure 52: Sort, Maps and Reduces

Figure 53 shows the distribution of data transfer sizes. As previously, the information is presented in two rows, corresponding the top row to the data produced for the set of Hadoop daemons, and the bottom row to the set of tasks. Darker colours represent higher frequencies, and lighter colours represent less frequent data sizes. Data transfer sizes are distributed from 0KB to 18KB, being the most frequent data transfers of less than 4KB. It is clear that the data transfers initiated by the slave processes are more uniform than the transfers initiated by the Hadoop daemons.

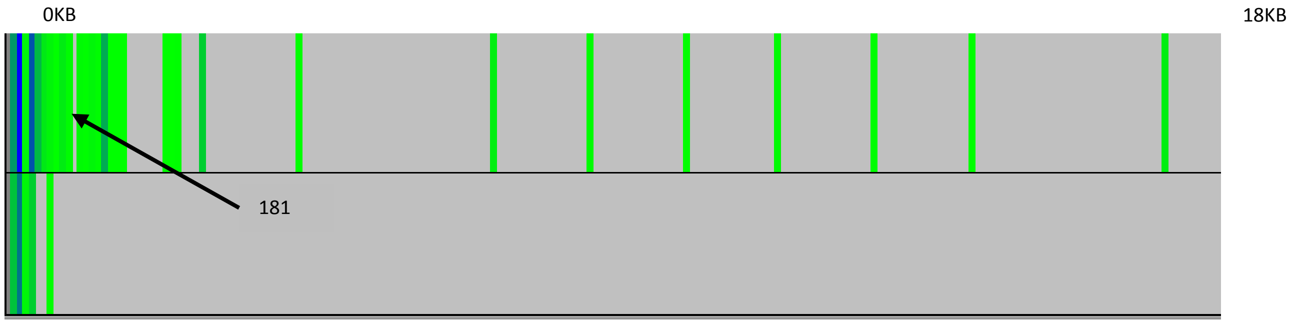


Figure 53: Sort, histogram of data transfer sizes

The following Figure 54 represents a distribution of transfer latencies. Each row represents a process (either daemons or tasks) and each column represents a bin of latencies. The colour of each cell represents the frequency of data transfer latencies observed for the value represented by the cell. Darker colours represent higher frequencies, and lighter colours represent less frequent data sizes. Data transfer latencies are distributed from 0us to 800us, being the most frequent data transfers of less than 100us.

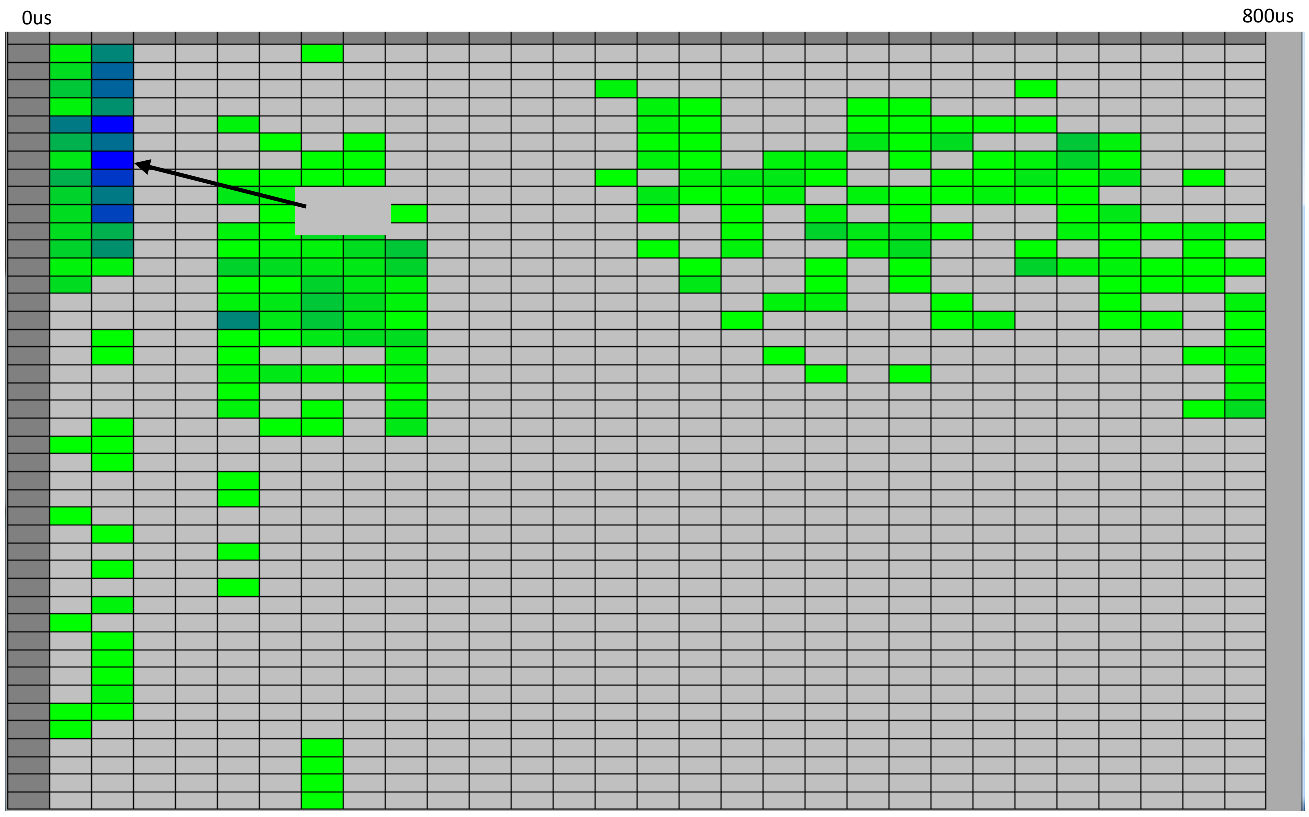


Figure 54: Sort, histogram of data transfer latencies

3.3.6. Workload characterization: TERASORT

The TeraSort application is a single job application that aims to sort a large number of key/value pairs. On its original form, it is used to sort 1TB of data, but for the purposes of this study we used a reduced input of 500MB of data. Data, like in the case of the Sort application, is sorted using the native Hadoop data ordering mechanisms, and therefore sorting is not done in user code. The structure of the application can be seen in Figure 55, in which different computing waves compute the provided input (showing dark blue areas the time at which activity takes place) can be observed. The execution of all tasks happens at the same time across nodes, being initiated in each one of the 4 slave nodes. As it can be observed there is a large number of communications between the Hadoop daemons (top threads in the capture) and the actual tasks in execution.

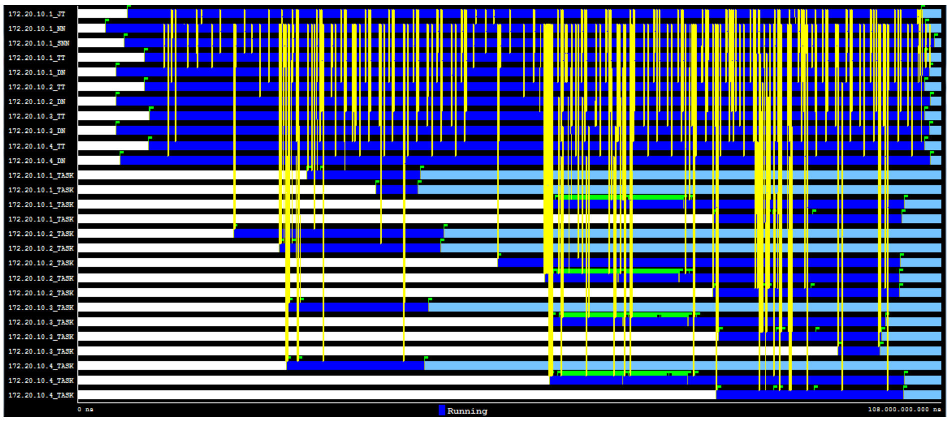


Figure 55: Terasort acitivity trace

As in the case of sort, it can be observed in Figure 56 that this application also has a simpler pattern than PageRank. It contains two waves of map phases (in green), which are basically identity assignments: that is, the input text is directly issued as the output of the map phase. Then the internal mechanisms of the Hadoop runtime perform the sorting, which is transparent for the user code. Finally, the reduce phase is run in parallel in all the nodes to produce the final output.

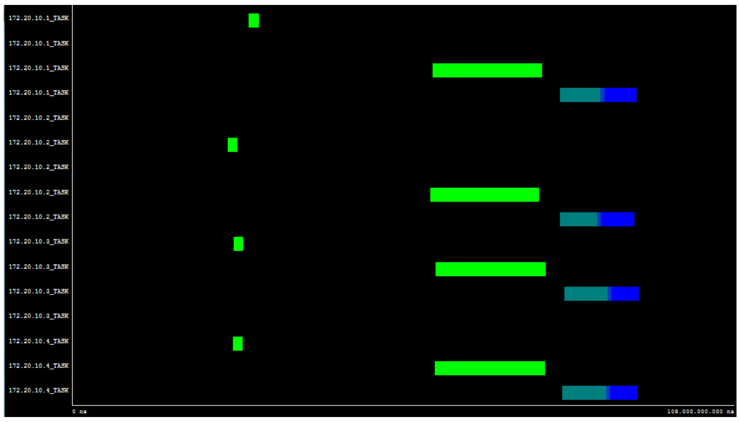


Figure 56: Terasort, Maps and Reduces

Figure 57 shows the distribution of data transfer sizes. As previously, the information is presented in two rows, corresponding the top row to the data produced for the set of Hadoop daemons, and the bottom row to the set of tasks. Darker colours represent higher frequencies, and lighter colours represent less frequent data sizes. Data transfer sizes are distributed from 0KB to 16KB, being again the most frequent data transfers of less than 4KB. It is clear that the data transfers initiated by the slave processes are more uniform than the transfers initiated by the Hadoop daemons.

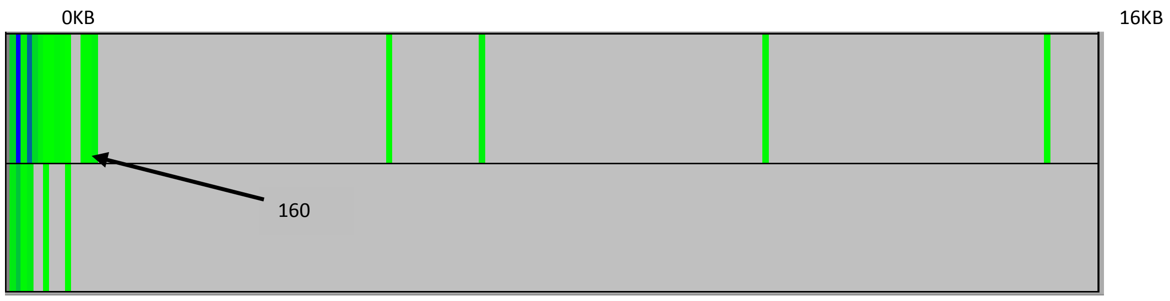


Figure 57: Terasort, histogram of data transfer sizes

Figure 58 represents a distribution of transfer latencies. Each row represents a process (either daemons or tasks) and each column represents a bin of latencies. The colour of each cell represents the frequency of data transfer latencies observed for the value represented by the cell. Darker colours represent higher frequencies, and lighter colours represent less frequent data sizes. Data transfer latencies are distributed from 0us to 600us, being the most frequent data transfers of less than 100us.

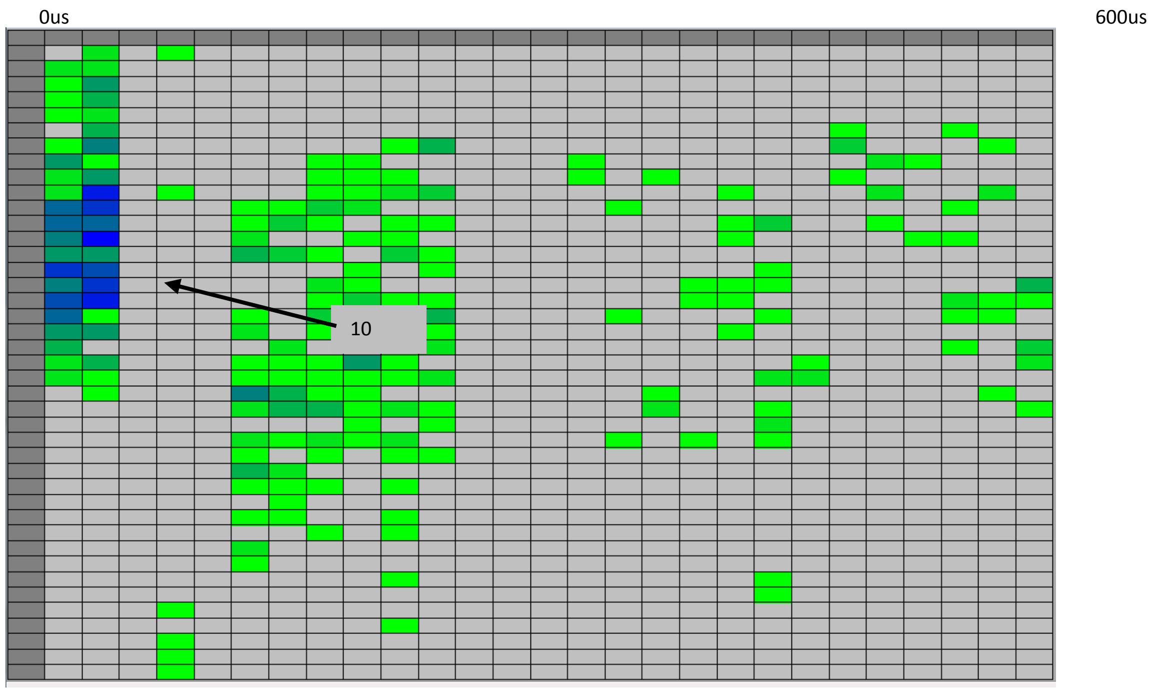


Figure 58: Terasort, histogram of data transfer latencies

4. Efficient Application Management Technique

As can be seen in chapter 3, OPS and OCS have advantages and disadvantages in data transfer of HPC and Big Data applications. The negative impact in performance is much lower when using OPS. Still, with proper combining of servers into racks this impact could be even lower when using OCS instead of using OPS. Figure 59 shows the methodology of choosing OPS or OCS for executing an application. The first and the most important step is the mapping which includes grouping servers in racks, number of used fibers and number of wavelengths per fiber.

The specific characteristic of OCS is shown in Figure 35. Grouping servers into two racks makes the execution possible with setting up just one OCS path. This way, 25ms would be added to the whole execution time, which is a cost of setting up the path. To group the servers into two racks makes sense for the number of processes that we analysed.

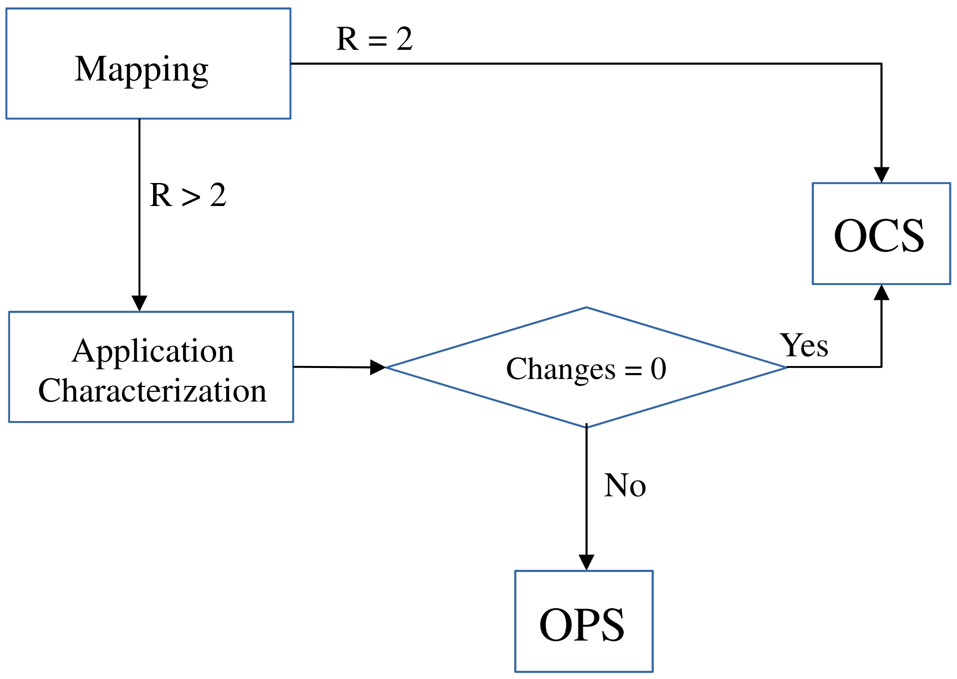


Figure 59: Methodology of choosing OPS or OCS

Methodology begins with mapping. The servers that will run analysed applications should be grouped in two racks, this way the advantage of OCS can be used. If servers have to be installed in more than two different racks, number of required OCS paths needs to be counted. If there are no changes of OCS paths once the first paths are set, the OCS switch should be used because it will not add any latency during the execution. If the changes exist, it is better to use OPS, because high number of OCS path changes would add significant latency.

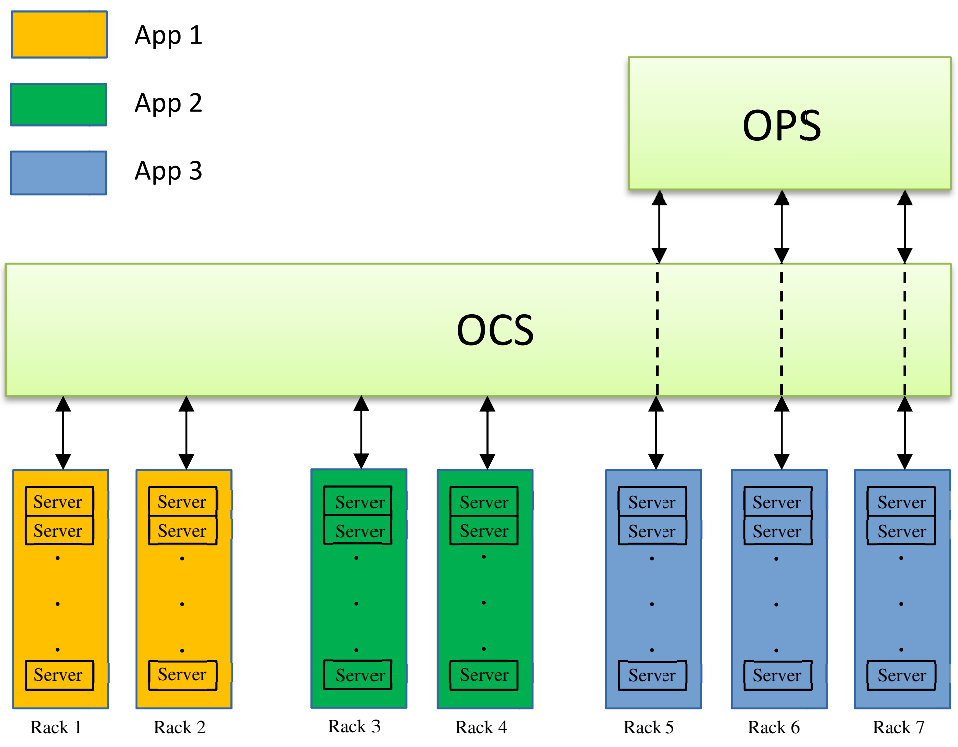


Figure 60: Methodology of choosing OPS or OCS

Figure 60 shows the usage of proposed approach. If application could be run on servers located in two racks (App 1 and App 2), then OCS should be used. If application is executed on servers which are installed in more than two racks and setting up of OCS paths repeats during the execution, then the best approach is to use OPS (App 3).

Even if we use two ports per rack, the paths in both directions will be set in less than 1ms. Regarding the time interval needed for setting one OCS path (25ms), these events can be observed as simultaneous. If we use two ports per rack, one port will be used for incoming and the other for outgoing traffic. With one port per rack, instead of two, resource costs will be lower. This port will be used in both directions, for outgoing and incoming traffic.

There are the applications that demand higher number of servers than 16 (for *SORT* application this number is 35). The problem with this approach could be the number of servers per rack, but there are some possibilities like virtual servers that can be installed instead of several physical devices. One wavelength could be used by several servers in the same rack, but this will be discussed in next deliverable which is a study of scalability.

Execution of one application trace used in previous tests lasts around 4ms. We observed the application execution comparing the time with and without any added latency that exist when using OPS or OCS device. Regarding OPS device we assume that application with longer execution time will introduce similar percentage of latency. This especially refers to HPC applications because of the iterative nature of these applications.

For traces that last 4ms, this latency is quite insignificant, but when the whole application needs to be run, the latency could be significant, depending on the execution time. However, with longer application execution time, the percentage of performance difference will not change notably, since latency is proportional with the execution time.

Using two racks and the OCS, it does not matter how long the execution lasts, at the end there will be no added latency after the first path is set. Figure 61 shows the improvement that use of OCS introduces comparing to default mapping of OPS.

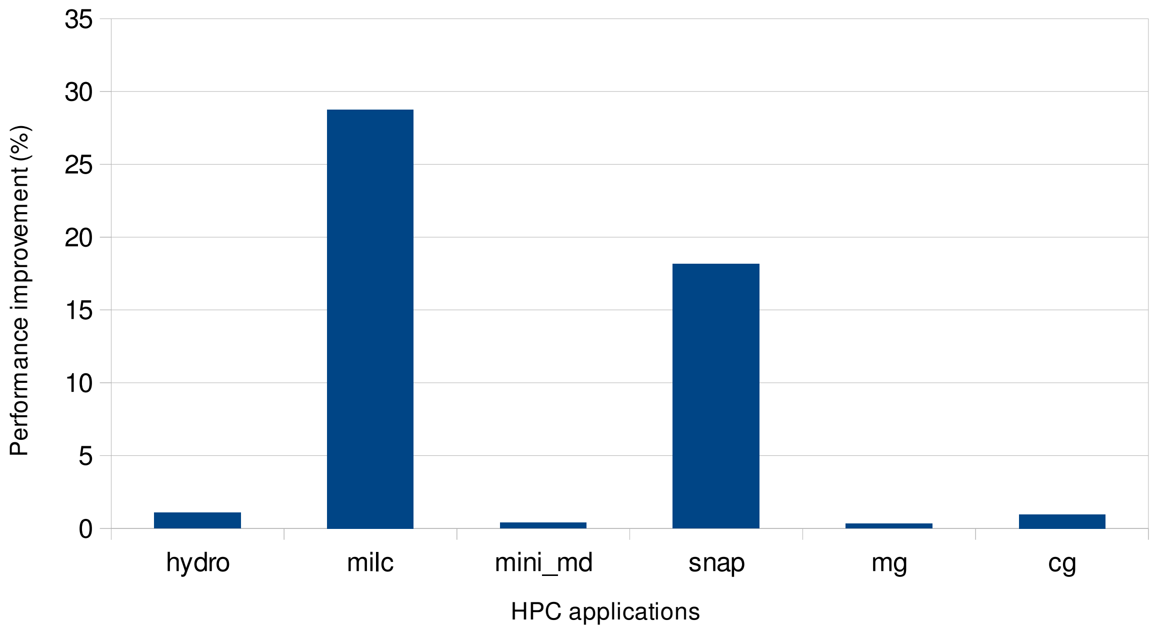
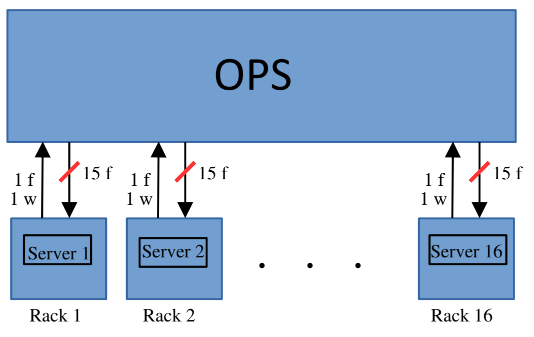
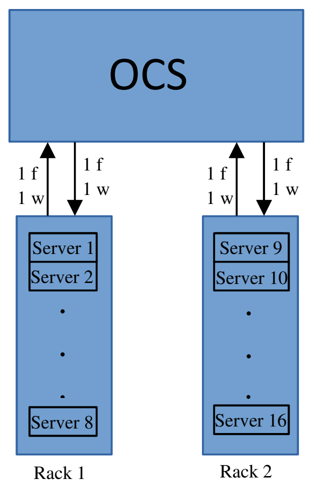


Figure 61: Simulation results – OCS proposed mapping

Default OPS mapping with no collisions is shown in Figure 62a. Using the OCS mapping that we propose (Figure 62b) improves the performance of every application comparing to default OPS mapping. For some applications this improvement is not noticeable but the performance is also not worse. For *MILC*, improvement of performance is around 30%. If we compare this mapping of OCS with different mappings of OPS (), the improvement with OPS could be seen for SNAP application (around 17%), which is similar to improvement with OCS proposed mapping (around 18%). When the OPS was used, we achieved the best result for *SNAP* application by grouping the servers also into two racks.





(a) (b)

Figure 62: OPS default mapping and OCS proposed mapping

5. Energy efficiency in data centres

5.1. Rationale and motivation

Data centres can be considered as the backbone of the modern economy, spanning from server rooms of small and medium-sized companies, to enterprise data centres of worldwide corporations and bigger server farms hosting cloud computing services from major cloud providers (Amazon, Google, etc.). Moreover, the recent explosion of digital content, big data, e-commerce, and Internet traffic in general is also turning data centres into one of the fastest growing consumers and seekers of energy and electricity.

Nowadays, data centres consume up to 3% of all global electricity production while emitting an equivalent quota of 200 million metric tons of CO2 [14]. In 2013, U.S. data centres consumed around 91 billion kilowatt-hours of electricity. Data centre electricity consumption is expected to increase to roughly 140 billion kilowatt-hours annually by 2020, costing around $13 billion annually in electricity bills and emitting nearly 100 million metric tons of CO2 per year [15].

In this huge power consumption environment, the hyper-scale cloud computing data centres represent only a small fraction of the whole energy consumption. The majority of energy is indeed consumed in small, medium, and large corporate data centres as well as in the multi-tenant data centres where lot of public and private companies are outsourcing their IT needs. These data centres have generally made much less progress in reducing and "greening" their energy consumption than their hyper-scale cloud ones due to persistent issues and market barriers.

When deploying a data centre, there is a set of key challenges to be addressed that has an impact on the requested investment and operational costs, including energy consumption and resource utilization: a) select an adequate physical environment, that take into account efficient power management, b) proper cooling infrastructure and network technologies, c) efficient support and maintenance strategies to address business continuity and disaster recovery.

A key technology that has emerged in recent years as a key enabler for reduction of power consumption and efficient resource utilization is the use of virtualization systems and services. By consolidating multiple physical servers into fewer virtualized machines, data centres are improving resource utilizations and reducing operational costs. Virtualization is currently a key technology implemented by data centre owners to replace the typical business customers' and end users' requirements to buy, manage and maintain physical infrastructures at their premises, while offering a scalable, secure, simple to operate and cost effective solution for computing, storage and integrated applications.

Beyond this innovation, data centre virtualization can reduce costs at different levels, including power, cooling and hardware, while simplifying administration and maintenance in a greener IT profile. The migration to hosted data centres (for customers) and the improvement of on-premise ones (for providers) can be achieved in a cost effective and scalable way through virtualization. For these reasons, data centre virtualization has been also adopted in LIGHTNESS, as explained in deliverable D4.6 [16].

5.2. Key factors and technology enablers

Data centre providers target to maximise the utilisation and the efficiency of all the resources employed in their infrastructures. As electric power is a must-have resource for all the operations of a data centre, the optimisation of energy consumption can have direct benefits on the overall data centre operation costs, as well as on the efficiency and performance of offered services.

Two main categories of data centre operation costs can benefit from proper energy consumption strategies and procedures: infrastructure costs and power costs. Infrastructure costs are those related to facilities dedicated to consistent power delivery and to evacuating heat, and consequently, these costs represent the main overhead of data centres. Cost items composing infrastructure include chillers, Universal Power Supply (UPS), Power Distribution unit (PDU), CRAC (Computer Room Air Conditioned). On the other hand, power costs represent one of the crucial items in the data centre cost breakdown for the data centre, mainly due to high cost of electricity and high load of data centres themselves.

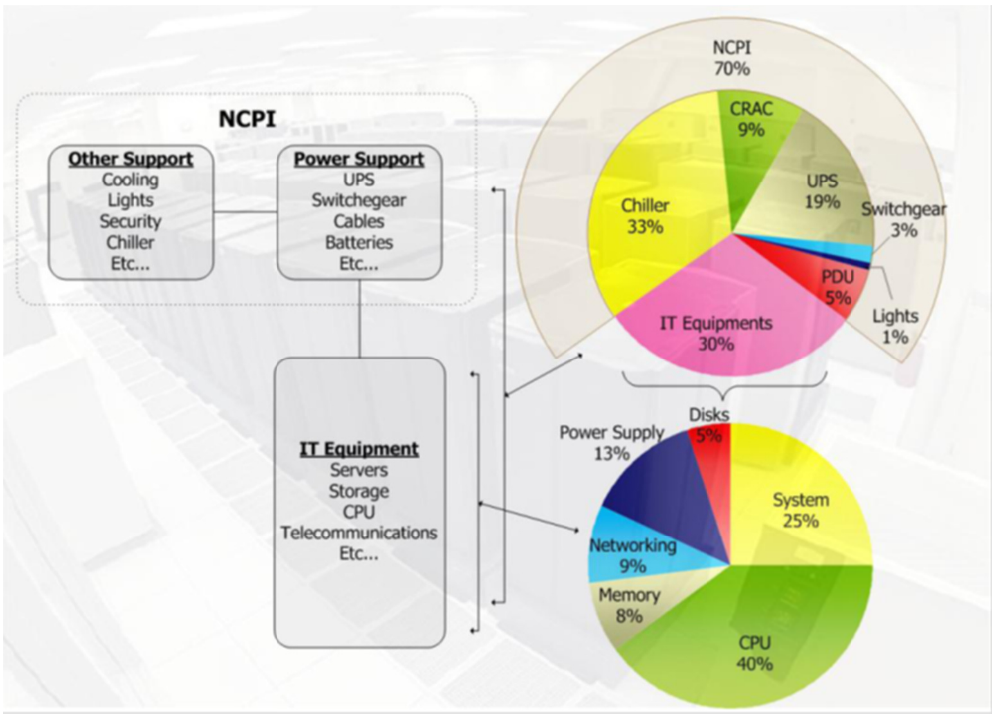


Figure 63: Typical breakdown for data centre power consumption (source Green Grid)

In the context of LIGHTNESS, and this document in particular, the relevant item to be properly analysed, managed, and mitigated where possible is the cost related to power usage. A typical breakdown for the power utilization, in the case of a generic medium size enterprise data centre identified by Green Grid [17], and that can help to understand how energy consumption is split across the different power items is presented in Figure 63. According to Green Grid, power is consumed by two major categories of equipment and facilities in the data centre operation:

- Network-Critical Physical Infrastructure (NCPI) equipment, that include cooling, lights, chiller infrastructures on the one hand, and all the power delivery units like switchgears and UPS on the other that basically represent the power support facilities within a data centre. Apart from the exact power consumption figures and portions shown in Figure 63, the NCPI typically consumes most of the energy needed to operate a data centre

- IT equipment, that include all those physical devices building the actual data centre, like bare metal servers, storage and networking devices, CPUs running services offered by providers.

Therefore, mechanisms, procedures and strategies targeting the optimization of power consumption within a data centre need to tackle the above two power consumption sources. While the NCPI component is mostly related to the data centre architecture, in terms of spaces, rooms, how racks are organized, cooled and supplied, the IT equipment component is the one that can be optimized and improved implementing innovative data centre technologies (at different layers, including network, control plane, SLA management, etc). It is also true that a proper optimization of IT equipment power consumption can have an indirect impact and benefit on the NCPI component.

The reduction of power consumption and the optimization of energy utilization within data centres is one of the top business objectives of data centre providers. A number of energy-aware strategies for data centre operation have been proposed including workload consolidation [18][19], optimal placement of workload [20][21], scheduling of applications [22][23], detection of power efficient servers and reduction of power consumption by cooling systems [24][25]. Also, at a higher level, data centre operators aiming to save energy costs and comply with environmental regulations (e.g. CO2 certificates) offer energy-aware SLAs to their customers, offering services with a “green” profile (and often with reduced performances) at lower price, opening opportunities for “green” marketing options.

In this LIGHTNESS data centre energy consumption analysis we focus on two technology enablers strictly related to novel approaches implemented in the project: data centre virtualization and multi-tenancy, and optical network technologies. The following two sub-sections briefly present benefits brought by these two technologies.

5.2.1. Data centre virtualization and multi-tenancy

Data centre virtualization and multi-tenancy has recently emerged as a key technology to enable and ease proper management, mitigation and optimization of energy consumption. In particular, a crucial innovation point is the coordinated virtualization of IT (i.e. servers) and network resources inside the data centre. Server domain virtualization is currently a consolidated approach by means of stable and scalable approaches based on hypervisors. It directly brings energy efficiency by sharing server resources (computation, storage) across different applications running in isolated Virtual Machines. Moreover, Virtual Machines can be easily provisioned on-demand and operated in a flexible way, thus achieving further energy savings by following the real needs of customers in terms of resources. Similarly, the creation of multiple co-existing and independent virtual networks in support of server virtualization allows achieving a shared and more flexible use of data centre network resources. The combination with server virtualization mechanism enables a dynamic and programmable coordinated virtualization of data centre resources. And most important, the concept of network virtualization has recently gained significant attention within the data centre and cloud communities as a key enabler for additional benefits for service providers and enterprises, such as reduced cost per application, improved resource utilization and energy consumption, rapid service delivery and mobility of applications within and across data centres.

Practically, data centre virtualization implemented in a coordinated way at both server and network level allows an efficient sharing of data centre resources across tenants and customers. This provides a significant reduction of power and energy consumption in two main directions. Directly, by reducing the amount of physical devices (i.e. servers or network switches and routers) needed to run a given amount of services for customers, or by increasing the amount of services and tenants running within a given fixed data centre infrastructure. Indirectly, by removing a portion of the power consumed by UPS and cooling systems for exceeding physical devices.

In addition to the considerations above, and to the direct and indirect power consumption improvements presented, data centre virtualization can be implemented by operators following energy-aware mechanisms. In particular, dedicated proactive and reactive procedures for deploying or moving Virtual Machines to servers in the same rack or same room may help in energy consumption reduction, as well as reorganizing workload for optimal energy efficiency with migration of Virtual Machines to prioritize the use of the most efficient servers. This requires collection of monitoring information concerning server status, consumption, performance to be gathered by energy-aware optimization processes that can dynamically reconfigure the virtual environment in the data centre, at both IT and network level.

5.2.2. Optical network technologies

Data centres are facing the rapid growth of cloud applications and hosting services with intensive workloads that are steadily boosting the internet traffic and putting great challenges and requirements on the underlying data centre interconnect networks. Current electronic switch fabrics built in a hierarchical structure could nominally support up to 10 Gb/s per port, but are becoming extremely costly and complex to operate at large-scale while keeping control of power consumption and heat dissipation. Photonic-based data centre network solutions can provide significant improvements over traditional electronic architecture in terms of power consumption [26]. Following this direction, the data centre network architecture proposed in LIGHTNESS relies on the integration in a full optical flat fabric of transparent Optical Packet Switching (OPS) and Optical Circuit Switching (OCS) technologies, that allow to avoid expensive Optical-Electrical-Optical (OEO) conversions, optical transceivers, and cables, reducing the energy consumption and cost of current electrical solutions.

Optical networks have inherent characteristics that support reduction of energy consumption and waste as well as lowering of operation cost for a data centre network through longer product lifecycles. One of the benefits of deploying an optical data centre network is the reduced energy consumption with respect to copper-based systems, not just initially, but also over the life of the network. Recent innovations in copper chipsets for 10GBase-T applications have brought power consumption for copper networks down to between 1 and 2 watts over shorter distances and about 3.5W at full 100-meter reach capability. On the other hand, optical networks, in comparison, may use less than 1W to transmit the 10-GbE signal over the IEEE specs of 300 meters [27]. Moreover, power savings from an optical network can be substantial over time, especially in data centre environments with thousands of network connections.

According to the Environmental Protection Agency (EPA) Green Power Equivalency Calculator Methodologies, the factor to calculate CO2 greenhouse gas (GHG) savings is 6.8956 x 10-4 metric tons CO2 / kWh [28]. Therefore, in a data centre environment where thousands of ports are deployed, a full optical architecture can save hundreds or thousands of kWh per year compared to the equivalent full copper data centre networks just considering port power consumption. Moreover, since optical networks use less energy to power the signal, they also generate less heat and therefore require less cooling. The EPA also evaluated that each KW of network power in a data centre requires a KW of power for the cooling facilities. Similar to virtualization, the indirect effect of deploying an optical data centre network is a reduction of energy needed to cool it. In addition to the energy savings, this also means you need less heating, ventilation and air-conditioning equipment, thus saving on materials and floor space.

Moreover, for data centre environments, optical fibre installations typically require fewer line cards than copper due to higher density of fibre line cards, leading to a potential deployment of fewer chassis. The optical fibre patch cords and cables that connect the equipment can also be themselves of high-density design (i.e. multi fibre), further condensing solutions.

As a last benefit, optical technologies typically have a longer infrastructure lifespan, and the inherently high bandwidth in optical networks means that once they are installed, they have a potential working life of 25-plus years without the need of re-cabling. This provides data centre operators a migration strategy that minimizes materials consumption and reduces the total operation cost of the network.

5.3. Interoute: an example of energy consumption evaluation

Interoute owns and operates one of the largest and most advanced unified ICT networks in Europe, that encompass 13 hosting data centres locally operated and centrally monitored (EU, USA, Asia), 32 purpose-built co-location centres and 60 owned Points of Presence (PoPs). This pan-European infrastructure is designed for the delivery of enterprise IaaS and virtualized services and is directly connected and interconnected through a network owned by Interoute. Each Interoute data centre is deployed with a network fabric deeply embedded in the company pan-European infrastructure to support effective access, geographic fallback and full product range. Major services offered by Interoute and running in this large data centre infrastructure include: managed hosting services, virtualization services (as a specialized offer of managed hosting, mostly Virtual Data Centre – VDC), colocation services (that provide companies with a range of flexible alternatives to housing their systems internally).

As any other data centre owner, Interoute has a strong interest in operating its data centre infrastructure in an efficient and cost effective way, trying to optimize the resource utilization while keeping under control and possibly reducing as much as possible the energy consumption. Interoute believes that data centre operation efficiency is key due to current constraints and drivers imposed by emerging cloud businesses and needs. Key benefits of efficiency will not only be saved energy, that is a crucial point to optimize data centre operation costs, but the ability to deploy more IT throughput towards customers while improving the service offer.

Interoute strategy for energy efficiency is built around four key sustainability actions: 1) measure energy consumption, 2) optimize resource utilization, 3) automate data centre operation, 4) monitor and improve the infrastructure. The key enabler for this approach is the energy consumption model implemented in Interoute, which is used to measure and monitor energy consumed with related costs against KPIs defined for each data centre. It is worth to mention that for sake of confidentiality, some figures, constraints, meters and details in general of the Interoute energy consumption model reported below are abstracted and hidden in this brief summary.

The Interoute data centre energy consumption model is based on measurement and monitoring of a constellation of five key efficiency parameters:

1. IT space usage, which in turn is computed as a function of two components:
   a. IT floor space usage
   b. IT rack space usage
2. Data centre efficiency, which in turn is computed as a function of three components:
   a. Server usage
   b. Network usage
   c. Cooling usage
3. IT cooling usage, which in turn is computed as a function of three components:
   a. CRAC
   b. Condensers and chillers
   c. Air flow
4. IT power usage, which in turn is computed as a function of three components:
   a. UPS
   b. Switchgears
   c. Transformers and distribution
5. Data centre infrastructure availability, which in turn is computed as a function of three components:
   a. Server uptime
   b. Network uptime
   c. Cooling uptime

Figure 64 shows an example of this data centre efficiency constellation for one of the Interoute data centres. For each parameter, three key values are reported: a) the Interoute goal, that is the target to be achieved for that parameter in the given data centre; b) the Interoute minimum goal, and when a given parameter is below this threshold some countermeasures or dedicated actions are needed to recover the energy inefficiency; c) the actual value measured for the parameter.

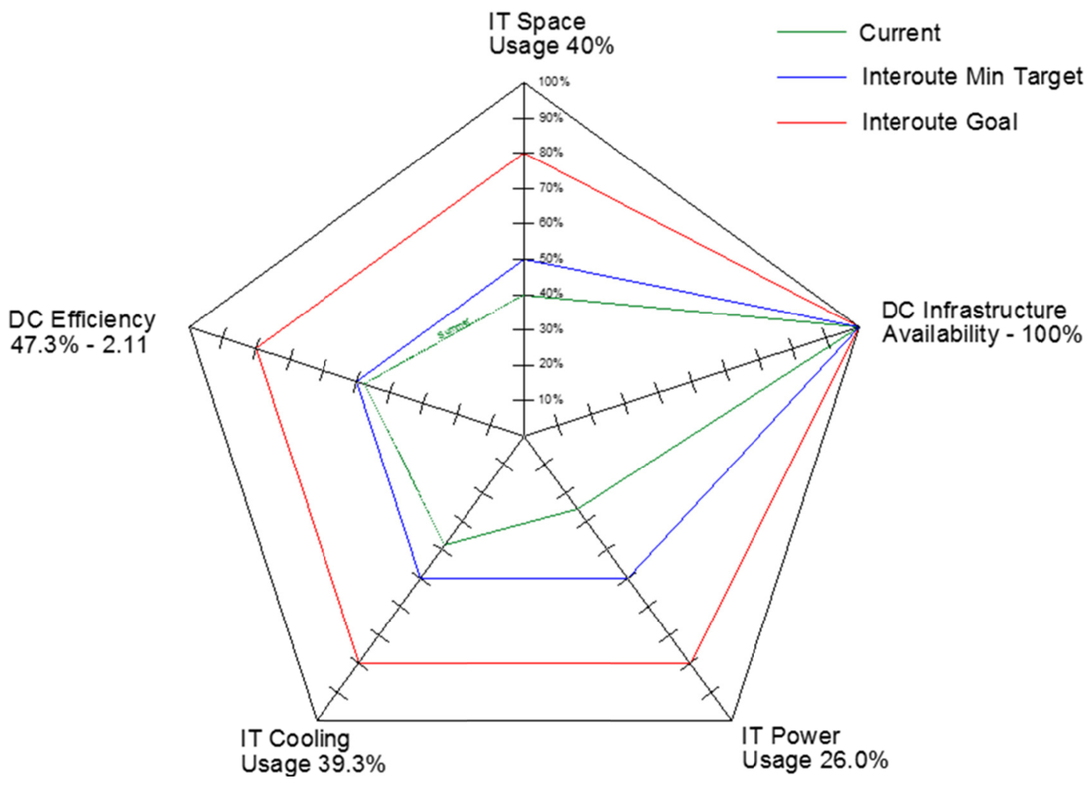


Figure 64: Interoute energy consumption KPIs in Berlin data centre

This constellation of key efficiency parameters is continuously monitored against the KPIs within each Interoute data centre, and combining these information, the data centre infrastructure usage is evaluated in terms of energy (in KW) consumed and yearly costs, as depicted in Figure 65. The energy consumed is evaluated at two levels of granularity. First, the total energy consumed by the data centre (Figure 65a) is split among power used for the data centre building only (e.g. lights, support services, surveillance, security, etc.) and the actual data centre operation, including IT, network and cooling facilities. Second, the data centre operation energy consumption (green portion in Figure 65a), is then split among three major components: 1) network equipment, 2) IT facilities (i.e. servers) dedicated to managed hosting and virtualized services, 3) IT facilities (i.e. servers) dedicated to colocation services. All these three components include power for facilities and cooling to heat down them. The network equipment part is evaluated as the composition of the power consumed by all the network devices employed in the data centre, taking into account both power supply and portion of cooling. The network devices deployed in Interoute data centres are mainly of two categories: devices needed for intra data centre networks, that in all 13 data centres are structured in legacy hierarchical architectures employing electronic switches; b) the PoP part which includes all those devices that interconnect the data centre to the Interoute pan-European network.

Starting from the split in Figure 65a, the data centre yearly cost for energy is also evaluated, to monitor the actual impact of energy consumed at different levels in the overall data centre operation costs (Figure 65b).

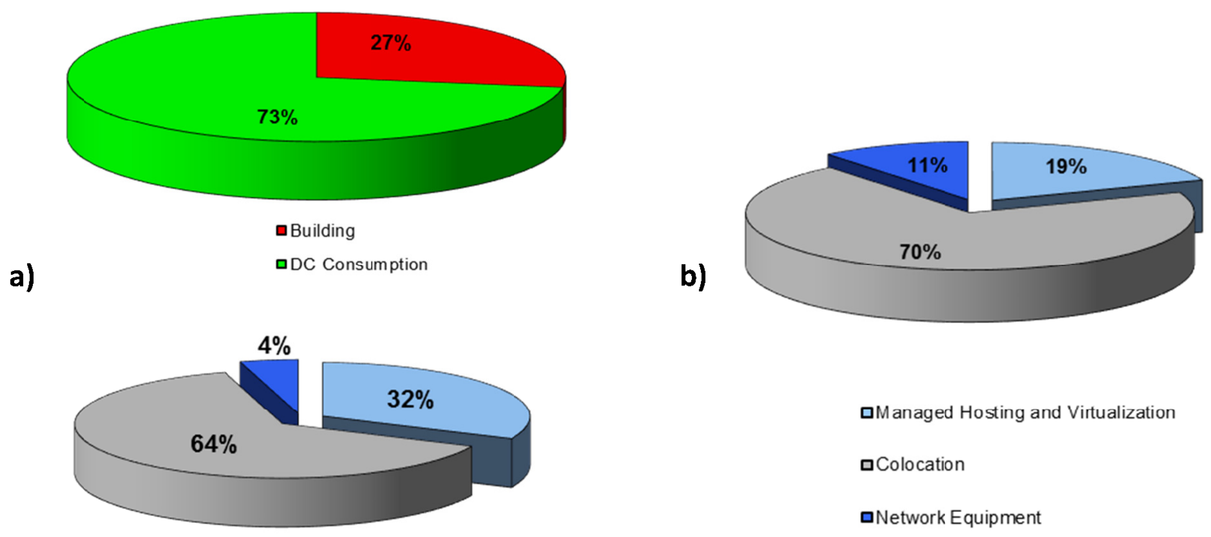


Figure 65: a) Berlin data centre power consumption split b) Berlin data centre yearly energy cost split

As said, as part of the management duties of Interoute infrastructure, all these energy efficiency and consumption evaluations are matched against data centre KPIs, and proper mitigation and optimization strategies are applied when inefficiencies are encountered, following the four actions sustainability approach described above.

As a key optimization approach for resource usage efficiency and reduction of energy consumption, Interoute has already deeply implemented data centre virtualization and multi-tenancy in its pan-European infrastructure. Among the available solutions, Virtual Data Centre (VDC) is the top product in the Interoute portfolio. It is a multi-tenant Infrastructure as a Service (IaaS) platform for on-demand computing and cloud hosting with integrated applications that enables either private or public cloud computing, offering public simplicity with private cloud security. VDC customers build their own virtual data centre through a VDC Control Centre graphical interface and they can provision virtualized servers, storage volumes and network segments within a specific zone (i.e., an Interoute data centre site). In the near term, Interoute intends to further consolidate its VDC offer by following its energy efficiency and sustainability strategy described above, in particular for what concern automation of data centre operation and improvement of infrastructure, by integrating SDN based control functions to dynamically and flexibly set up and reconfigure VDCs for customers.

Moreover, the adoption of optical technologies within its data centres is a key innovation that Interoute has planned in its evolution roadmap. The certain benefits of optical data centre networks in terms of energy consumption, in addition to unprecedented performance and scalability, are convincing Interoute that optics is the right technological approach to address and match the challenging requirements of emerging and future cloud applications. In this direction, the LIGHTNESS full optical and multi technology data centre fabric could be a good starting point and candidate to plan a partial migration of selected Interoute data centres to optical solutions.

6. Conclusions

This document presents the LIGHTNESS management traffic technique and key energy optimization strategies for data centres. LIGHTNESS network is based on the innovation of combining both OPS and OCS technologies in the same network. These optical technologies provide quite different performance. For instance, in OPS there is no need to setup connections but adds always minimum delay of several nanoseconds per packet transmission, whereas in OCS there is no delay but setting up OCS connections could take a million times higher than the OPS delay. The LIGHTNESS management traffic technique harnesses the advantages of both OPS and OCS technologies in order to boost the performance of applications. The LIGTHNESS management traffic technique is based on the insights collected during this study when analysing the communication traffic patterns of applications and the implications of the process mapping strategy. The study found that OCS technologies can be efficiently exploited by deploying two different strategies aimed to minimize the number of OCS connection setups. Firstly, by carefully mapping application processes to servers, and secondly by analysing application communication traffic to check the number of OCS connections required. Both techniques are required in order to distribute the traffic to OCS and OPS devices and exploit the benefits of zero latency in OCS technologies and the benefits of statistical multiplexing of OPS technologies. The application communication traffic characterization is conducted using executing tracing tools that have been developed for the LIGHTNESS project. In particular, it has been developed tools to trace BigData workloads. This communication traffic characterization methodology look at several features of the application traffic to quantify and asses the sensitivity of applications to different optical technologies. A performance validation of the LIGHTNESS management traffic technique has been provided showing that the technique efficiently harness the potential benefits of OCS and OPS achieving the best performance for all applications.

Regarding efficient energy optimization strategies, LIGHTNESS is proposing two key technology enablers to efficiently reduce the vast energy consumption of data centres. These techniques are devised to be deployed by Interoute data centres. The first technique is based on data centre virtualization technologies and the second one is based on deploying optical network technologies. Optical networks have inherent characteristics that support reduction of energy consumption and waste as well as lowering of operation cost for a data centre network through longer product lifecycles with respect to their copper-based counterpart data centres.

LIGHTNESS architecture is based on the Architecture on Demand (AoD) where the OPS is attached to the OCS in order to provide flexibility. The evaluation of this architecture will be provided in our coming next deliverable D2.5.

References

[1] http://www.bsc.es/marenostrum-support-services/mn3

[2] Extrae: http://www.bsc.es/computer-sciences/performance-tools/trace-generation

[3] Jeffrey Dean and Sanjay Ghemawat. “MapReduce:Simplied Data Processing on Large Clusters”.Google inc. 2004.

[4] Di Lucente, S., Calabretta, N., Resing, J.A.C. & Dorren, H.J.S. (2012). Scaling low-latency optical packet switches to a thousand ports. Journal of Optical Communications and Networking, 4(9), A17-A28. in Web of Science.

[5] Polatis OCS switches http://ww.polatis.com

[6] Paraver: http://www.bsc.es/computer-sciences/performance-tools/paraver

[7] Dimemas: http://www.bsc.es/computer-sciences/performance-tools/dimemas

[8] Partnership for Advanced Computing in Europe: http://www.prace-ri.eu/

[9] Department of Physics and Astronomy, The University of Utah: http://www.physics.utah.edu/~detar/milc/milcv7.html

[10] Mantevo application performance project: https://mantevo.org/

[11] Los Alamos National Lab: https://github.com/losalamos/SNAP

[12] NASA Advanced Supercomputing Division: http://www.nas.nasa.gov/publications/npb.html

[13] OMNEST: http://www.omnest.com/

[14] Data Centre Journal, “Industry Outlook: Data Center Energy Efficiency”, available on-line at: http://www.datacenterjournal.com/industry-outlook-data-center-energy-efficiency/

[15] Natural Resources Defense Council, “America's Data Centers Consuming and Wasting Growing Amounts of Energy”, available on-line at: http://www.nrdc.org/energy/data-center-efficiency-assessment.asp

[16] LIGHTNESS Deliverable D4.6 “SDN-based intra-DC network virtualisation mechanisms”, June 2014.

[17] The Green Grid – URL: http://www.thegreengrid.org/

[18] Banerjee, A., Mukherjee, T., Varsamopoulos, G., Gupta, S. K. S.. 2010. Cooling-aware and thermal-aware workload placement for green HPC Data Centres. In Proceedings of International Green Computing Conference (Chicago, IL, USA, August 15-18, 2010). 245-256. DOI=10.1109/DREENCOMP.2010.5598306.

[19] Meisner, D., Gold, B. T., and Wenisch, T. F. 2009. PowerNap: Eliminating server idle power. In proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (Washington, DC, USA, March 7-11, 2009). ASPLOS’09, ACM, New York, NY 205-216. DOI = 10.1145/1508244.1508269

[20] Carrol, R., Balasubramaniam, S., Donnelly, W., and D. Botvich. 2011. Dynamic optimization solution for green service migration in data centres. In Proceedings of IEEE International Conference on Communications (Kyoto, Japan, June 5-9, 2011). ICC'11.pp. 1-6, 2011. DOI =10.1109/icc.2011.5963030.

[21] C. Dupont, G. Giuliani, F. Hermenier, T. Schulze, and A. Somov. An Energy Aware Framework for Vir- tual Machine Placement in Cloud Federated Data Cen- tres, In Proceedings of the 3rd International Conference on Energy-Efficient Computing and Networking (e- Energy'12), ACM, Madrid, Spain, May 7–9, 2012.

[22] R., and Torres, J. 2010. Towards energy-aware scheduling in Data Centres using machine learning. In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (Passau, Germany, April 13-15, 2010). e- Energy'10. ACM, New York, NY 215-224. DOI=10.1145/1791314.1791349

[23] Garg, S. K., Yeo, C. S., Anandasivam, A., and Buyya, R. 2010. Environment-conscious scheduling of HPC applications on distributed cloud-oriented Data Centres. Journal of Parallel and Distributed Computing. 71 (2010), 732-749.

[24] Barbagallo, D., Nitto, E., Dubois, D. J., and Mirandola, R. 2010. A Bio-inspired algorithm for energy optimization in a self-organizing Data Centre. In Proceedings of the FirstSelforganizing architectures (Cambridge, UK). SOAR'09.Springer-Verlag Berlin, Heidelberg, 127-151. ISBN:3-642-14411-X 978-3-642-14411-0.

[25] Pakbaznia, E. and Pedram, M. 2009. Minimizing Data Centre cooling and server power costs. In Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design (San Francisco, CA, USA, August 19-21). ISPLED'09. ACM, New York, NY 145-150.DOI = http://doi.acm.org/10.1145/1594233.1594268

[26] C. Kachris, K. Bergman and I. Tomkos, Optical Interconnects for Future Data Centre Networks (Springer, 2013), Chap.1.

[27] Cabling Installation and Maintenance, How fiber can help make your network "greener", available on-line at: http://www.cablinginstall.com/articles/print/volume-21/issue-4/features/how-fiber-can-help-make-your-network-greener.html

[28] Green Power Equivalency Calculator Methodologies, April 2011, www.epa.gov/greenpower

Acronyms

ACK	Acknowledgement
AoD	Architecture on Demand
API	Application Programming Interface
CPU	Central Processing Unit
CRAC	Computer Room Air Conditioned
DC	Data Centre
DCN	Data Centre Network
DDR	Double Data Rate
EPA	Environmental Protection Agency
FDR	Fourteen Data Rate
GHG	Greenhouse Gas
HDFS	Hadoop Distributed File System
HPC	High Performance Computing
HTTP	Hypertext Transfer Protocol
ICT	Information and Communications Technology
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IT	Information Technology

JNI	Java Native Interface
KPI	Key Performance Indicator
MPI	Message Passing Interface
NACK	Negative Acknowledgement
NCPI	Network-Critical Physical Infrastructure
NIC	Network Interface Card
OCS	Optical Circuit Switching
OEO	Optical-Electrical-Optical
OPS	Optical Packet Switching
PDU	Power Distribution Unit
PID	Process Identifier
PoP	Point of Presence
RAM	Random-access memory
RPC	Remote Procedure Call
SRC	Source Server
SDN	Software Defined Networking
SLA	Service Level Agreement
TCP	Transmission Control Protocol
UPS	Universal Power Supply
VDC	Virtual Data Centre