UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ANDERSON MATTHEUS MALISZEWSKI

Impact of Network Interconnection in Cloud Computing Environments for High-Performance Computing Applications

Advisor: Prof. Dr. Philippe Olivier Alexandre Navaux
Coadvisor: Prof. Dr. Dalvan Griebler

Porto Alegre
October 2021

CIP — CATALOGING-IN-PUBLICATION

Maliszewski, Anderson Mattheus

Impact of Network Interconnection in Cloud Computing Environments for High-Performance Computing Applications / Anderson Mattheus Maliszewski. – Porto Alegre: PPGC da UFRGS, 2021.

100 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2021. Advisor: Philippe Olivier Alexandre Navaux; Coadvisor: Dalvan Griebler.

I. Navaux, Philippe Olivier Alexandre. II. Griebler, Dalvan. III. Título.

“Is this the end of the beginning?
Or the beginning of the end?”
— BLACK SABBATH

AGRADECIMENTOS

Este trabalho é dedicado ao meu Pai (*in memoriam*) Luiz, minha Mãe Márcia, meu Opa Ernesto e minha Oma Elzira. Obrigado por todo apoio, auxílio e luta que tiveram em suas vidas para proporcionar o melhor possível para a minha caminhada. À minha namorada, Maiara, por todo apoio, amor, paciência, carinho e dedicação. Sem vocês jamais teria conseguido.

Agradeço ao meu orientador Prof. Philippe O. A. Navaux e a meu co-orientador Prof. Dalvan Griebler (este último que já me acompanha desde a orientação no curso de Redes de Computadores - SETREM) pela oportunidade, apoio e aprendizados que me proporcionaram durante o período do meu mestrado.

Aos colegas do Grupo de Processamento Paralelo e Distribuído (GPPD), obrigado por todo aprendizado e momentos de descontração. Menção a ser feita aos colegas Eduardo Roloff e Matheus Serpa que tiveram fundamental importância em meu ingresso no mestrado, estadia em Porto Alegre, e ajuda em vários momentos. Estendo meus cumprimentos aos demais professores do INF e amizades que fiz nas cadeiras que frequentei.

Aos colegas do Laboratório de Pesquisas Avançadas para Computação em Nuvem (LARCC), por todo aprendizado e ajuda que obtive.

Por fim, aos familiares e amigos que não foram citados em específico aqui, meu muito obrigado.

ABSTRACT

The availability of computational resources has changed significantly due to the use of the cloud computing paradigm. Aiming at potential advantages, such as cost savings through the *pay-per-use* method and scalable/elastic resource allocation, we have witnessed efforts to execute high-performance computing (HPC) applications in the cloud. Due to the distributed nature of these environments, performance is highly dependent on two primary components of the system: processing power and network interconnection. If allocating more powerful hardware theoretically increases performance, it increases the allocation cost on the other hand. Allocation exclusivity guarantees space for memory, storage, and CPU. This is not the case for the network interconnection since several simultaneous instances (multi-tenants) share the same communication channel, making the network a bottleneck. Therefore, this dissertation aims to analyze the impact of network interconnection on the execution of workloads from the HPC domain. We carried out two different assessments. The first concentrates on different network interconnections (GbE and InfiniBand) in the Microsoft Azure public cloud and costs related to their use. The second focuses on different network configurations using NIC aggregation methodologies in a private cloud-controlled environment. The results obtained showed that network interconnection is a crucial aspect and can significantly impact the performance of HPC applications executed in the cloud. In the Azure public cloud, the accelerated networking approach, which allows the instance to have a high-performance interconnection without additional charges, allows significant performance improvements for HPC applications with better cost efficiency. Finally, in the private cloud environment, the NIC aggregation approach outperformed the baseline up to $\approx$98% of the executions with applications that make intensive use of the network. Also, Balance Round-Robin aggregation mode performed better than 802.3ad aggregation mode in the majority of the executions.

Keywords: Cloud Computing. Network Interconnection. High-Performance Computing. Performance Evaluation. Microsoft Azure. OpenNebula.

RESUMO

A disponibilidade de recursos computacionais mudou significativamente devido ao uso do paradigma de computação em nuvem. Visando vantagens potenciais, como economia de custos por meio do método de pagamento por uso e alocação de recursos escalável/elástica, testemunhamos esforços para executar aplicações de computação de alto desempenho (HPC) na nuvem. Devido à natureza distribuída desses ambientes, o desempenho é altamente dependente de dois componentes principais do sistema: potência de processamento e interconexão de rede. Se a alocação de um hardware mais poderoso teoricamente aumenta o desempenho, ele aumenta o custo de alocação, por outro lado. A exclusividade de alocação garante espaço para memória, armazenamento e CPU. Este não é o caso da interconexão de rede, pois várias instâncias simultâneas (multilocatários) compartilham o mesmo canal de comunicação, tornando a rede um gargalo. Portanto, esta dissertação tem como objetivo analisar o impacto da interconexão de redes na execução de cargas de trabalho do domínio HPC. Realizamos duas avaliações diferentes. O primeiro concentra-se em diferentes interconexões de rede (GbE e InfiniBand) na nuvem pública da Microsoft Azure e nos custos relacionados ao seu uso. O segundo se concentra em diferentes configurações de rede usando metodologias de agregação de NICs em um ambiente controlado por nuvem privada. Os resultados obtidos mostraram que a interconexão de rede é um aspecto crucial e pode impactar significativamente no desempenho das aplicações HPC executados na nuvem. Na nuvem pública do Azure, a abordagem de rede acelerada, que permite que a instância tenha uma interconexão de alto desempenho sem encargos adicionais, permite melhorias significativas de desempenho para aplicações HPC com melhor custo-benefício. Finalmente, no ambiente de nuvem privada, a abordagem de agregação NIC superou a linha de base em até 98% das execuções com aplicações que fazem uso intensivo da rede. Além disso, o modo de agregação Balance Round-Robin teve um desempenho melhor do que o modo de agregação 802.3ad na maioria das execuções.

Palavras-chave: Computação em Nuvem. Interconexão de Rede. Computação de Alto Desempenho. Avaliação de Desempenho. Microsoft Azure. OpenNebula.

LIST OF ABBREVIATIONS AND ACRONYMS

AWS	Amazon Web Services
BT	Block Tridiagonal
CC	Cloud Computing
CG	Conjugate Gradient
CPU	Central Process Unit
CRM	Customer Relationship Management
CSV	Comma Separated Values
EC2	Elastic Cloud 2
EP	Embarrassingly Parallel
ERP	Enterprise Resource Planning
FDR	Fourteen Data Rate
FT	Fourier Transform
GbE	Gigabit Ethernet
GCC	GNU Compiler Collection
GCE	Google Compute Engine
GNU	GNU's Not Unix
GPU	Graphics Processing Unit
HDFS	Hadoop Distributed File System
HPC	High-Performance Computing
HPL	High-Performance Linpack
IaaS	Infrastructure as a Service
IB	InfiniBand
IEEE	Institute of Electrical and Electronics Engineers
IPoIB	Internet Protocol over InfiniBand

IPv4	Internet Protocol version 4
IPv6	Internet Procotol version 6
IS	Integer Sort
KVM	Kernel-based Virtual Machine
LA	Link Aggregation
LAN	Local Area Network
LU	Lower-Upper
LXC	Linux Containers
MAC	Media Access Control
MG	Multi-Grid
MPI	Massage Passing Interface
MPTCP	MultiPath TCP
NFS	Network File System
NAS	NASA Advanced Supercomputing Division
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NPB	NAS Parallel Benchmarks
OS	Operating System
OTF2	Open Trace Format Version 2
PaaS	Platform as a Service
QDR	Quad Data Rate
RAM	Random Access Memory
RDMA	Remote Direct Memory Access
RoCE	RDMA overConverged Ethernet
RR	Round Robin
SaaS	Software as a Service

SLA	Service Level Agreement
SP	Scalar Penta-diagonal
SR-IOV	Single Root Input/Output Virtualization
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VM	Virtual Machine
XOR	eXclusive OR

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

With the increase in complexity and number of computational problems as well as in the acquisition value of private infrastructures, there has been a significant migration from traditional environments to those that provide resources in a fast, scalable, and pay-for-use manner, such as cloud computing (BHOWMIK, 2017).

Cloud environments have been developed over several technologies (e.g., virtualization), and characteristics from distributed, grid, and parallel computing available practically since 2010. Nowadays, it is a consolidated model capable of providing computing resources on-demand (e.g., CPU, GPU, memory, storage, network) without upfront investments through three service layers, known as IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) (MELL; GRANCE et al., 2011).

According to Gartner's forecasts and surveys[1], the migration from usual computation environments to public clouds, which was already considered significant before the pandemic, tends to increase by $\approx$18% in 2021, spending 304 billion dollars. This statistic mainly considers the complete "validation" of the cloud environment since during the COVID-19 crisis several jobs became remote or even needed greater flexibility.

High-Performance Computing (HPC), which is in turn supplied by clusters, grids, and cloud computing "as a service" models, has historically been used to speeding up data processing. Cloud Computing (CC) potential has increased due to the improvements on the technology stack, and it can provide an alternative to the usual computing methods, both in resource scalability and cost reduction.

Aiming for these benefits, we have witnessed efforts to execute HPC applications on clouds. These benefits typically came at the price of performance losses due to the negative impact of the virtualization layer (compared with the native environment) and the overhead of multi-tenants sharing/competing for resources (e.g., network interconnection) (EMERAS et al., 2019; GUPTA et al., 2016).

Moreover, as HPC applications executed in clusters or even in cloud computing environments are generally developed using Message Passing Interface (MPI), and the communication characteristics of applications vary due to their specific purpose, network interconnection may impact the overall performance. Therefore, the computing environment must ensure high-performance communication: high throughput and low latency to address the application's requirements and not become the entire system bottle-

[1] <https://www.gartner.com/en/newsroom/press-releases/2020-11-17-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-18-percent-in-2021>

neck (ESCUDERO-SAHUQUILLO et al., 2015; KAMBURUGAMUVE et al., 2017). However, as previous research studies have shown (ROLOFF et al., 2017; MOURA; HUTCHISON, 2016; SADOOGHI et al., 2017), network interconnection is still a considerable challenge for parallel applications executing in clouds.

1.1 Motivation

Cloud computing offers several benefits through its "as a Service" models, like pay-per-use, elasticity, and instant access to a pool of computational resources. Significant migration of entire environments is commonly seen in recent years. This migration also includes the execution of HPC applications in the cloud, seeking the benefits above.

In applications that demand a considerable amount of computational resources, the performance needs to be assured. However, two main problems well-known in the literature can limit the overall performance due to cloud characteristics.

The first problem refers to virtualization, which can induce performance losses compared to the native environment, used as a basis for CC. With joint efforts by academia and industry to reduce such losses, containers were created, which use light virtualization, known as virtualization at the level of OS with containers.

The second problem concerns an HPC premise, which seeks to extract as many resources as possible. This condition is only achieved theoretically with the guarantee of exclusivity of resources. Although the cloud can guarantee this priority allocation in memory, processor, and storage, it is not guaranteed to network interconnection since switching equipment is inevitably shared between several servers.

Also, it is observed that in cloud environments, the underlying network infrastructure is transparent to the user who allocates resources. Thus, the processing of several network flows originating from different instances can cause fluctuations in latency and throughput, making the network interconnection one of the main bottlenecks of cloud computing (ROLOFF et al., 2017; SADOOGHI et al., 2017).

Our objective is to evaluate the impact of network interconnection in cloud computing environments for HPC applications. For this, we performed three evaluations: We selected representative HPC applications and profiled them. We compare the usage of high-performance against traditional network interconnections regarding performance and cost efficiency in a public cloud. We employed the NIC aggregation configuration integrated with a private cloud and compared various scenarios with simultaneous users

(multi-tenants) executing HPC applications. With both performance evaluation results, we can observe the impact of the network in the private/public clouds.

1.2 Evaluations

In the following sections, we define the why and how our evaluations were made. Also, we shortly introduce the evaluations methodology.

1.2.1 Profiling HPC Applications

High-Performance Computing is a term used to describe the ability to process data and perform complex calculations at high speeds. Their solutions are mostly seen within giant computational infrastructures knew as supercomputers. These infrastructures are composed of thousands of computing nodes creating a cluster interconnected by network technologies, working together to complete one or more tasks.

The idealization of supercomputers was created based on the evolution of applications programming and the design of computers processor. Firstly, all processors were created with a single-core, and thus applications were executed serially. This concept has changed with the introduction of multi and many-core processors, which allowed applications to be executed using the parallel computing paradigm. With the rise of computer parallelism, complex problems that demand large amounts of calculation could be solved by decomposing the problem into smaller tasks and executing each task's instructions in parallel using several processes and computational cores.

With the advent of the multiprocessor, different parallel programming models have made it possible to improve the overall system performance. There are many classes of parallel hardware and, consequently, many different parallel programming models. Between them, we can highlight the shared memory and distributed memory models.

The shared memory model is used in environments with several processors that share the address space of a single memory; that is, processors can operate independently but share the same memory resources. The most common technique to create parallel programs using shared memory is the OpenMP.

On the other hand, the distributed memory model has several processors, each with its memory interconnected by a communication network. The tasks share data through the

communication of sending and receiving messages, and thus, multiple tasks are initiated and distributed by the processors of the environment, using their memory address. The most common technique to use distributed memory is MPI, and it is used in distributed environments with usually more than one node (e.g., clusters, grids, clouds).

This thesis used representative parallel applications from the NAS Parallel Benchmarks suite's Message Passing Interface (MPI) implementation. These benchmarks were chosen because of their high utilization in the academy and to cover a higher range of parallel patterns. Thus, we start profiling the applications to explore the parallel patterns and corroborate our performance evaluation on the clouds. We expect the profiling results to obtain the application characteristics and determine if a specific application is driven by communication or computation.

1.2.2 Cloud Computing

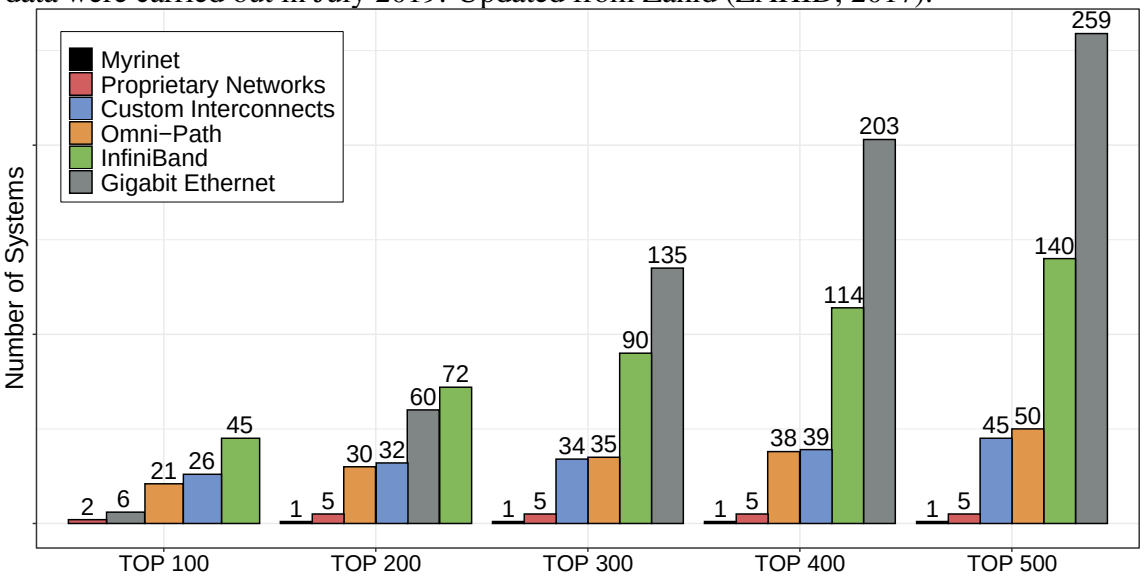
Historically, large computational clusters with thousands of servers have satisfied hardware requirements for executing High-Performance Computing (HPC) applications. However, as these applications usually require a significant amount of resources to obtain performance, this model has identified several problems. The main one is about costs since the hardware is inevitably constantly updated, in addition to its maintenance and energy cost. Performance in this environment depends heavily upon two main system components: processing power and network interconnection.

Focusing on network interconnection, over the last decade, we have seen a growth in the popularity of InfiniBand as a network interconnection for HPC systems and data centers. Viewing the interconnections used in *clusters* inserted in the Top 500[2] list (Figure 1.1) it is clear that 140 of the 500 supercomputers use InfiniBand (IB) as the primary interconnect, being surpassed only by the Ethernet interconnect family (e.g., 1GbE, 10GbE) with 259. However, when the Top 500 list is seen only by its first 100 systems, IB has a more significant presence than Ethernet, representing 45% of the total interconnections used, suggesting that its performance characteristics make it more suitable for large supercomputer installations.

As clouds use clusters to provide computing resources to their clients, the same performance issues are inherited, such as network interconnection. The performance disparity between HPC systems and cloud infrastructures is because many current cloud

[2]<https://www.top500.org/>

Figure 1.1: Family of network interconnects used by Top 500 systems. The assessment data were carried out in July 2019. Updated from Zahid (ZAHID, 2017).



systems use low-cost commodity Ethernet networks providing relatively low bandwidth and high latency between nodes. On the other hand, HPC interconnects technologies like InfiniBand are increasing their usage in clouds/instances focused on HPC. However, if, on the one hand, allocating more powerful hardware theoretically boosts performance, on the other hand, it increases the allocation cost.

1.2.3 Public Clouds

Public clouds are generally created in large computational infrastructures (also known as clusters or supercomputers), which a cloud provider owns. It provides on-demand computation resources to its clients, paying for their use (pay-per-use premise) according to the utilization. The usage of public clouds removes the idealization of maintaining and supporting the hardware infrastructure, as the provider does this. Thus, the user can focus on their utilization.

However, this transparent provision also has its drawbacks. For instance, the user does not know the infrastructure's internal network topology and can allocate hardware from different pods to create a cluster. Also, the user is limited to using the options provided by the cloud provider and can not manage the low-level configuration of the servers. In addition, the network infrastructure is not allocated by a single user, so streams from multiple simultaneous instances (multi-tenant) tend to share them.

We evaluated how the network interconnection impacts performance and cost efficiency in a public cloud-based on those drawbacks. We deployed three individual clusters in Microsoft Azure public cloud provider, each with eight instances and different size/network interconnections. For example, one cluster with A10 instance size and 10GbE, another cluster with DS4_v2 instance size and 40GbE IB, and finally, the last cluster with F8 instance size and 50GbE IB interconnection.

Considering that a faster interconnection theoretically improves performance, we also verified if it could be more cost efficient. We executed representative parallel applications from the NAS Parallel Benchmarks suite and the Alya HPC application in those clusters. With the obtained results, we expect to answer the following question: *Considering that a faster interconnection theoretically results in improved application performance, can it also lead to a higher cost efficient?*

1.2.4 Private Clouds

As its name suggests, private clouds are computational infrastructures managed and owned by a private identity, for instance, a company or a research laboratory. Contrary to public clouds, private cloud environments are not fully adherent to the essential characteristics of CC defined by NIST. This happens because the number of resources, the elasticity, and the pay-per-use billing model are inconsistent. Also, this is the cloud model with a higher price involved since the organization needs to maintain and buy the computation infrastructure. On the other hand, it also provides more security to the organization and higher low-level management to perform specific upgrades or changes.

In this work, we deployed a private cloud using the OpenNebula cloud manager. Then we explore low-level configurations of the network interconnection to perform our evaluation. We used a private cloud because this was the only cloud model accessible, allowing us to perform low-level configurations. We created clusters with four identical physical hosts using LXD containers and integrated them with the NIC aggregation technique. We developed an evaluation methodology considering different numbers of aggregated NICS (up to four), different modes of aggregation (802.3ad and Balanced-RR), and up to three simultaneous LXD instances running specific applications to create noise in the network concerning the fourth and central LXD instance. In this way, we experience a real multi-tenant environment and evaluate the type of interference that happens on it.

With this evaluation, we aim to answer the following questions: *Can the NIC*

aggregation approach improve HPC applications' performance on the cloud? Which is the number of NICs aggregated and aggregation mode that provides better performance?

1.3 Goal and Contributions

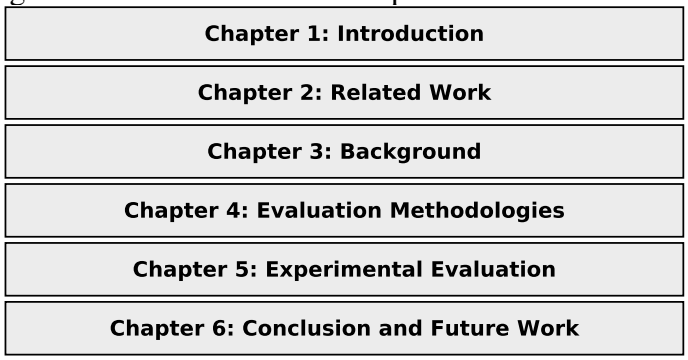
The main goal of our research is to evaluate the impact of network interconnection in cloud computing environments for high-performance computing applications. For this goal, our main contributions are the following:

- Performance and cost assessment for public cloud environment on instances with different network interconnections.
- Performance assessment for private cloud environment under different network configurations based on NIC aggregation methodologies.

1.4 Text Organization

The remainder of this dissertation consists of the following chapters. **Chapter 2** presents the related work. **Chapter 3** briefly presents the background of the dissertation, including cloud computing, and NIC aggregation. **Chapter 4** presents the evaluation methodologies of this dissertation. **Chapter 5** shows the experimental evaluation, including the benchmark profiling, the high-performance interconnects in public clouds, and NIC aggregation in private clouds. **Chapter 6** summarizes our conclusions and outlines ideas for future work that can be based on this dissertation. An overview of the structure of this work is shown in Figure 1.2.

Figure 1.2: Overview of the chapters of this dissertation.



2 RELATED WORK

In this chapter, we grouped state of the art regarding evaluations in network performance for cloud environments. We considered as related work those that tackle network performance optimizations/evaluations in public/private clouds and clusters. The selected related work are described below.

2.1 High-Performance Networks Evaluations

Since the introduction of high-performance interconnects, several evaluations have been done comparing them in terms of performance. For instance, Vienne et al. (VIENNE et al., 2012) made a comprehensive assessment of several high-performance interconnections, including 10/40 GbE, InfiniBand 32 Gbps Quad Data Rate (QDR), InfiniBand 54 Gbps Fourteen Data Rate (FDR), and 10/40 GigE RDMA over Converged Ethernet (RoCE). The experiments were conducted in HPC and cloud computing environments, using NAS Parallel Benchmarks, TestDFSIO, and HBase benchmarks to assess the impact of interconnections on HPC performance, primarily HDFS, and cloud computing benchmarks. They concluded that the latest InfiniBand FDR interconnect offers the best performance in terms of latency and bandwidth in HPC and cloud computing systems. RoCE 40 GigE delivered better performance when compared to IB QDR in network-level assessments and for HPC applications, and IPoIB QDR provided better performance than 40 GigE Sockets for cloud computing middleware.

On the other hand, we have legacy works like Liu et al. (LIU; WU; PANDA, 2004) that evaluated MPI over InfiniBand, proposing a new design to achieve better scalability, exploiting application communication patterns using an RDMA-Based implementation with MVAPICH, which benefits not only large but also small and control messages. They provide measures of latency and throughput. Also, to corroborate its implementation, they used the benchmarks of NAS Parallel Benchmarks using classes A and B. In the same trend, Liu et al. (LIU et al., 2003) performs a detailed performance comparison of MPI over InfiniBand Myrinet, and Quadrics. They first characterized the MPI implementations with micro-benchmarks, measuring both latency and throughput, and after using applications like NAS Parallel Benchmarks with class B and Sweep3D to evaluate the performance. Also, they correlated the overall application performance results with the information acquired from the micro-benchmarks.

2.2 Public Cloud Evaluations

More recently, with the evolving of cluster/grid environments to cloud computing, relevant studies performed evaluations regarding network interconnection in public/private clouds. For instance, Gupta et al. (GUPTA et al., 2016) carried out an evaluation with several platforms and applications, including public cloud providers and clouds optimized for HPC. Considering the scalability of the cloud, different classes of applications were identified, differentiated by their communication patterns and the proportion between the number and size of messages. Scalability differences were found due to network virtualization, multi-tenant, and hardware heterogeneity. Based on these findings, the authors devised two general strategies to combat the performance limitations of HPC clouds. The first is about the decomposition of the work units, configuring the ideal size of the problem, and adjusting the network parameters to improve the computation and communication rate. The second is about using lightweight virtualization, configuring CPU affinity, and NIC aggregation to reduce the overhead of the underlying virtualization platform. The study also found that the more CPU cores an application requires, the more likely an HPC platform will offer the best cost/benefit ratio.

Mauch et al. (MAUCH; KUNZE; HILLENBRAND, 2013) described virtualization techniques, as well as management methods, such as multi-tenancy, dynamic provisioning, and Service Level Agreements (SLAs). The authors presented an approach for using high-performance network interconnections, in this case, InfiniBand (IB), in a private cloud environment deployed with the OpenNebula cloud manager framework. They implement a virtual cluster environment to verify the integration of the IB interconnect and deploy the private cloud as a proof of concept and compare it with Amazon's Elastic Compute Cloud (EC2) instances, which had interconnections from 1 to 10 Gigabit Ethernet. To verify the performance of the instances, the High-Performance Linpack (HPL) benchmark was used, and this showed that the performance penalty introduced by the virtualization layer is quite bearable in single-node operation. When nodes are combined into virtual clusters, communication latency plays a crucial role. In this way, the evaluation compared the private cloud with the AWS instances using the AkaMPI benchmark, which verified the latency and jitter of the network. The results emphasized that the InfiniBand interconnect provides better network performance for parallel applications that require low latency than a solution using Ethernet and is a better choice for HPC clouds.

Expósito et al. (EXPÓSITO et al., 2013) studied the performance of HPC applica-

tions on Amazon EC2 and focused mainly on aspects of I/O and scalability. In particular, they compared the CC1 with the CC2 instances launched in 2011 using the benchmarks from the NAS Parallel Benchmarks suite with up to 512 cores. They also investigated the cost-benefit of using these instances to execute HPC applications. They concluded that, although CC2 instances provide more communication performance, applications that make intensive use of collective communication performed worse than CC1 instances. In addition, they also concluded that the use of multilevel parallelism generated a scalable and economical alternative for applications on Amazon EC2 instances.

2.3 Private Cloud Evaluations

With relation to private clouds, Ruivo et al. (RUIVO et al., 2014) integrated OpenNebula with InfiniBand using the Single Root Input/Output Virtualization (SR-IOV) and Kernel-based Virtual Machine (KVM). Their approach includes evaluations on the worst-case scenario (small messages) in latency and bandwidth with micro-benchmarks and Linpack. Similarly, Chakthranont et al. (CHAKTHRANONT et al., 2014) integrated CloudStack with InfiniBand and conducted a performance evaluation in virtual and a physical cluster using Intel MPI benchmarks, HPC Challenge, OpenMX, and Graph500.

Vogel et al. (VOGEL et al., 2017) conducted a network performance assessment using the CloudStack manager, deploying clouds based on KVM and LXC. They measured network throughput and latency and indicated alternatives for improvements in network performance using the vhost-net module. The results showed that the KVM achieves fair yield rates but performance degradation in latency. On the other hand, LXC performed better in latency but lacked support and compatibility.

Zhang et al. (ZHANG et al., 2015) provided an efficient approach to build HPC clouds using OpenStack private cloud manager using SR-IOV enabled with InfiniBand clusters. They introduced a design to take advantage of SR-IOV for inter-node and intra-node communication and integrated it with OpenStack. Finally, they performed several performance evaluations with micro-benchmarks like OSU Micro-Benchmarks (OMB) and NPB, comparing their approach with Amazon EC2.

2.4 NIC Aggregation Evaluations

Other works have focused their investigations on network/link aggregation approaches. For instance, Watanabe et al. (WATANABE et al., 2008) investigated the impact of topology and link aggregation on a large-scale PC cluster with Ethernet. They performed several experiments with High-Performance LINPACK Benchmark (HPL) using 4-6 NICs aggregated using a torus topology. Their results have shown that the performance can be significantly improved in overall HPC applications up to 650%. This would allow cloud infrastructure using commodity hardware to improve network performance without significant additional investments in the hardware side.

Chaufournier et al. (CHAUFOURNIER et al., 2019) created a comprehensive assessment of the feasibility of using MPTCP to improve the performance of data center and cloud applications. Their results showed that while MPTCP provides useful bandwidth aggregation, congestion prevention, and improved resiliency for some cloud applications, these benefits do not apply uniformly across all applications. Similarly, Wang et al. (WANG et al., 2016) evaluated the applicability of MultiPath TCP (MPTCP) to improve the performance of the MapReduce application. Its scenario explored the capabilities of GPUs and showed the impact of network bottlenecks on applications performance. As a result, it demonstrated that aggregation of network links reduced the data transfer time and improved the overall performance.

Rista et al. (RISTA et al., 2017) created a methodology for evaluating performance measures such as bandwidth, throughput, latency, and execution times for Hadoop applications. The assessment also employed the Network Bonding 4 (IEEE 802.3ad) mode but mainly explored the benefits that aggregation brings with up to 3 instances simultaneously in LXC containers. As a result, they achieved performance improvements by reducing application times in $\approx$33%. Although the results obtained are promising, the use of simultaneous instances, also known as multi-tenant, does not apply to HPC applications, as these require no competition for computational resources.

2.5 Concluding Remarks

This chapter analyzed the related work on high-performance interconnects, public and private clouds, and NIC aggregation evaluations. The related work shown several different evaluations according to their characteristics. We summarized them in Table 2.1.

Table 2.1: Summary of related work. Each line represent a related work. Each column represents a desired property.

	Network Perfomance			
	HPC Network	**Public Cloud**	**Private Clouds**	**NIC Aggregation**
Vienne et al.	X			
Liu et al.	X			
Gupta et al.		X		
Expósito et al.		X		
Mauch et al.	X	X		
Vogel et al.			X	
Ruivo et al.	X		X	
Zhang et al.	X	X	X	
Watanabe et al.				X
Chaufournier et al.				X
Wang et al.				X
Rista et al.				X
This Work	X	X	X	X

There are several techniques to evaluate and create new solutions to improve the network interconnections in clouds. However, the majority typically involves upgrading the communication equipment, requiring a significant amount of financial resources. Due to the lack of comprehensive studies that estimate the impact of network interconnection in clouds, this situation was identified as a research opportunity. Our work goes beyond and looks for a comparison between HPC interconnections in terms of performance and cost and evaluates a NIC aggregation technique to improve performance cost effectively.

3 BACKGROUND

In this section, some important concepts used in this work are explained. We first review cloud computing, including its definition, characteristics, service models, and deployment models. Then we review some NIC Aggregation background, including its definition, aggregation modes, and hash policies.

3.1 Cloud Computing

Cloud computing is nowadays a consolidated model to provide computation resources on-demand. It was developed combining several characteristics from distributed, grid, and parallel computing and also consolidated technologies such as virtualization, which dynamically abstract hardware resources (BUYYA et al., 2009).

A definition of cloud computing well adopted by the community and used as a base in this document is given by the National Institute of Standards and Technology (NIST) as "a model which allows access to a shared pool of computing resources (e.g., networks, servers, storage, applications, and services) on-demand through the network, that can be quickly provisioned and released with minimal efforts or interactions by the service provider" (MELL; GRANCE et al., 2011). Also, it includes five essential characteristics, three service models, and four deployment models, which will be explained below.

3.1.1 Essential Characteristics

According to NIST (MELL; GRANCE et al., 2011), cloud computing has five essential characteristics.

- **On-demand self-service:** This first characteristic concerns the allocation and deallocation of computing resources without interaction with the provider's staff.
- **Broad network access:** All the on-demand services offered by the providers must be accessible over a network through standard mechanisms. For instance, over a local area network (LAN) or the Internet itself.
- **Resource pooling:** The providers are responsible for owning and managing the physical and virtual computing resources and providing them to multiple users (multi-tenant) according to their demand. These users have no control or knowl-

edge of the location of the resources but may specify location at a higher level (e.g., country, state, or datacenter).

- **Rapid elasticity:** The cloud providers offer the user's resources in any quantity at any time. From the customer's view, the resources seem to be unlimited.
- **Measured service:** The cloud systems need to provide transparency for both provider and final users. It is allowed for the provider to use techniques to measure resource usage and availability, such as monitoring, which will also be used to guarantee the service level agreement (SLA) and billing purposes.

3.1.2 Service Models

In cloud computing, the service models divide different clouds according to the computation abstraction capacity provided to the end-user. This definition is categorized in three service models (MELL; GRANCE et al., 2011; BADGER et al., 2012), known as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These models are substitutes for traditional infrastructure, but we pay for the volume of use instead of paying for licenses. They are described below:

- **Infrastructures as a Service (IaaS)** This model abstracts servers, storage components, physical space, and the network. This means that the company does not have to buy servers, routers, racks, and other hardware "boxes". However, operating systems, databases, and applications are the responsibility of the clients. The user takes care of the middleware and execution, while the heavier parts related to servers, processing, and memory are up to the provider. Thus, companies can develop their applications and platforms. It requires a more technical framework and specialized professionals. Here, the services are charged for factors such as the number of virtual servers, amount of data transferred, stored data, and other items, depending on the supplier. In this type of service, suppliers offer load balancing and security issues to allow some functions such as high-performance computing, Big Data analysis, artificial intelligence, and other needs that involve more robust computing. The great advantage is having everything flexible, scalable, with competitive prices. IaaS services facilitate the consolidation of external data center projects and allow companies that do not fully virtualize their operations to reduce costs with their servers, data storage, and supporting infrastructure. Examples of

this type of service are Amazon Web Services (AWS), Google Compute Engine (GCE), and Microsoft Azure.

- **Platform as a Service (PaaS)** It offers a development environment for the end-users to create their applications. In other words, they present the infrastructure, servers, tools, libraries, and databases so that companies only focus on their systems. Data and applications are the sole responsibility of the user. The use of PaaS eliminates the need to purchase, configure and manage hardware and software resources. The infrastructure is invisible to the developer, but he can configure the applications and, eventually, aspects related to their environment. It is a strategy that supports agile software development, allowing applications to be created with the abstraction of OS, middleware, data centers, and other resources. For this reason, it is commonly adopted by startups that do not have much money or personnel and wish to manage and modernize their creations. Mainly, having a development platform in the cloud is crucial for remote work, an overall strategy in organizations that are still starting or impacted by the COVID-19 pandemic. Examples of PaaS are the Google App Engine, Heroku, and Microsoft Azure Cloud Services.
- **Software as a Service (SaaS)** This is the most common model. As the term suggests, it is an application offered in an accessible way via the Internet and consumed for a specific price, which varies according to use. Generally, payments are monthly, depending on the package chosen. SaaS is the simplest model of all, as it abstracts practically everything for the user and allows him only to use the functionalities of a system to fulfill a particular objective. The benefit is that it does not require installation, an environment for execution, maintenance, and upgrades. On the part of the customer, it involves only one registration to use. In other words, in a strategy of this type, the company that hires does not have to worry about databases, operating systems, security, servers, or even issues involving the physical data center. If instability or any problem affects any of these factors, the responsibility lies with the provider. Examples of such systems are common CRMs (Customer Relationship Management) and ERPs (Enterprise Resource Planning) on the market, human resource management systems, business intelligence applications, storage, and e-mailm services. These solutions are generally highly customizable.

3.1.3 Comparing the Service Models

SaaS is the one that most abstracts structural aspects, so, as already mentioned, it is the simplest. The main difference is the type of user the service is intended for: ordinary users or corporations who want to use fast resources without worrying about advanced issues. In general, it is ideal for those with little technical knowledge.

PaaS is more expansive and specialized, aimed at developers. If SaaS is focused on using ready-made features, PaaS is the basis for developing the solutions themselves. IaaS, in turn, is aimed at infrastructure managers and IT administrators. It requires a much more theoretical framework since it brings a lesser abstraction of technical issues.

PaaS and IaaS are crucial for more specific activities, in which companies need to dimension the number of components needed very well. In this sense, specialized knowledge is needed to choose the most advantageous and ideal options, as the descriptions translate into specific and complex terms. For SaaS, the company does not need an in-depth study of their needs, whereas, for other models, this is essential.

3.1.4 Deployment Models

Deployment models address the access and availability of cloud computing environments. The restriction or opening of access depends on the business process, the type of information, and the level of vision. Some companies may not want all users to access and use certain features in their cloud computing environment. In this sense, there is a need for more restricted environments, where only a few duly authorized users can use the services provided. This definition is categorized in four deployments models (MELL; GRANCE et al., 2011; BADGER et al., 2012), known as Public, Private, Community, and Hybrid Clouds. They are described below:

- **Public Clouds**: In this model, resources are shared, usually located at a third-party provider that provides cloud computing solutions to diverse companies and individuals. The main advantage of the public cloud is that as resources are shared between several companies, this can help lower costs, especially if users are small businesses or individuals with little demand. Usually, a public cloud is accompanied by an authorization system.
- **Private Clouds**: In this model, resources are not shared. That is, dedicated servers

are used for the same company or individual. It can be hosted remotely (at a third-party provider) or locally (at the company itself). The main advantage is that so the user knows how much he can grow. The significant disadvantage appears for users with little load or load that varies a lot with time.

- **Community Clouds**: In this model, a group of people or companies, usually with common interests, create a cloud shared only between the community members. The resources are located with one or more community participants.
- **Hybrid Clouds**: It is just the model in which part of the infrastructure is private, and part is public. An example is Amazon, which uses part of its infrastructure as a Private Cloud for itself, and another part is made available to be rented by third parties, such as a Public Cloud.

3.2 NIC Aggregation

Also known as Link Aggregation (LA) or Bonding, it is a technique that combines several NICs into a logical link. It is commonly used to interconnect pairs of network devices (i.e., switches, routers.) to improve bandwidth and resilience cost-effectively by adding new links and existing ones instead of replacing equipment (IEEE, 2000; DAVIS et al., 2011). The specific behavior of connected interfaces is based on the choice of a usage mode among seven existing modes. These modes are configured directly in the Linux network interface configuration file and are expressed by both name and number. Another equally important use of NIC aggregation is to failover transparently. This is preferred for deployments where high availability is critical. The same idea can be further extended to provide a combination of increased bandwidth and transparent failover with degraded performance in a NIC failure event.

3.2.1 Aggregation Modes

Aggregation modes are responsible for specifying which policies will be used during the NIC aggregation. By default, mode 0 or balance-rr is used. The seven existing modes are listed and described below.

- **Balance-rr or 0:** It implements a Round-robin policy, transmitting all packets in sequence from the first node to the last, providing load balancing and fault tolerance.

- **Active-backup or 1:** It implements an Active-backup policy, where only one slave (network card) remains active. The only possibility for another slave to become active is in case of failure. In this mode, the MAC address is visible on only one network adapter port to avoid getting confused between the slaves. This mode offers fault tolerances.
- **Balance-xor or 2:** It implements an XOR policy, in which it selects an interface for the transmission of packets based on the result of an XOR operation in the count of slave network interfaces of the source and destination MAC address module. This calculation ensures that the same interface is selected for each destination MAC address used. This mode provides fault tolerance and load balancing.
- **Broadcast or 3:** It implements a Broadcast policy, allowing data traffic to occur on all slave interfaces, which provides fault tolerance.
- **802.3ad or 4:** It implements the IEEE 802.3ad Dynamic link aggregation protocol, creating aggregation groups with the same speeds and duplex configurations. It uses all slaves in the active aggregator, according to the 802.3ad specification. This mode has two prerequisites, which are: Ethtool support in the drivers, in addition to a switch that supports IEEE 802.3ad link aggregation.
- **Balance-tlb or 5:** It implements an adaptive load-balancing transmission, where it makes the connection between channels that do not require any exceptional support in the switch. Outgoing traffic is distributed according to the current network load on each slave, taking into account the traffic speed. The current slave receives incoming traffic. If the receiving slave fails, another slave will assume the MAC address of the failed slave. This mode requires the system to have Ethtool.
- **Balance-alb or 6:** It implements adaptive load balancing, including the balance-tlb, and receives load balancing modes for IPv4 traffic, requiring no exceptional support on the switch. Receiving load balancing is achieved through ARP negotiation.

3.2.2 Hash Policies

As its name suggests, hash policies are attributes that define which algorithm will be used in the network slave selection. They are valid in modes balance-xor, 802.3ad, and tlb modes and divided into layer2, layer2+3, and layer3+4. They are described below.

- **Layer 2:** This policy uses XOR of hardware MAC addresses and packet type ID

field to generate the hash. This algorithm will place all traffic to a particular network peer on the same slave.

- **Layer 2+3:** This policy uses a combination of layer2 and layer3 protocol information to generate the hash. Uses XOR of hardware MAC addresses and IP addresses to create the hash. If the protocol is IPv6, then the source and destination addresses are first hashed using ipv6_addr_hash. This algorithm will place all traffic to a particular network peer on the same slave. This policy is intended to provide a more balanced distribution of traffic than layer2 alone, especially in environments where a layer3 gateway device reaches most destinations.
- **Layer 3+4:** This policy uses upper layer protocol information, when available, to generate the hash. This allows traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves. If the protocol is IPv6, then the source and destination addresses are first hashed using ipv6_addr_hash. The source and destination port information are omitted for fragmented TCP or UDP packets and all other IPv4 and IPv6 protocol traffic. This algorithm is not fully 802.3ad compliant. A single TCP or UDP conversation containing fragmented and unfragmented packets will see packets striped across two interfaces. This process may result in out-of-order delivery. Most traffic types will not meet these criteria, as TCP rarely fragments traffic, and most UDP traffic is not involved in extended conversations. Other implementations of 802.3ad may or may not tolerate this noncompliance.

3.3 Concluding Remarks

This chapter introduces the main concepts of cloud computing based on the NIST definition and NIC aggregation based on the IEEE normative. Several studies have performed evaluations with cloud computing. On the other hand, NIC aggregation is not so known in the literature. In the following chapter, we will detail the evaluation methodologies used in this work.

4 EVALUATION METHODOLOGIES

This chapter is structured as follows. In Section 4.1 we detail the methodology used in our first evaluation, which profiles HPC applications from NAS Parallel Benchmark concerning the fractions of time that each MPI process spends in Computing and MPI Communication. In Section 4.2 we present the methodology of the evaluation in Azure public cloud concerning the application's performance and cost efficiency. Finally, the methodology of the private cloud evaluation concerning the NIC aggregation configurations with many different scenarios is presented in Section 4.3.

4.1 Benchmark Profiling

We perform a tracing procedure to determine the application's execution behavior, which allows us to create an execution profile of each application. Below we describe how this process was done.

This evaluation was made using the Message Passing Interface (MPI) parallel implementation of the Numerical Aerodynamic Simulation Parallel Benchmarks (NPB) (BAILEY et al., 1991) suite. We carry out a profile computing the fractions of time that each MPI process spends in Computing and MPI Communication of the applications. With this data, we classified the applications into four groups: the highly network-dependent (network-intensive utilization) to the group without network dependency.

4.1.1 Computational Infrastructure/Experiments Setup

Our experiments were performed with four identical nodes, each one composed with two Intel®Xeon®E5-2650 v3 (Q3'14) Haswell 2,3 GHz, 20-cores (10 by CPU) with Hyper-Threading enabled, resulting in 40 threads and 128GB DDR4 RAM. Each core has L1 (32KB instruction and 32KB data) and L2 (256KB) caches. L3 (256MB) cache is shared between all cores. Nodes are interconnected via a generic 1 Gbps switch. The software has Ubuntu Server 18.04 64-bit (kernel 4.15.0-48) as the operating system (OS), MPI library Open MPI 2.1.1, and GCC/GNU Fortran compiler 7.4.0.

To trace the applications and expose the MPI and Computing behavior, we used the Score-P (KNÜPFER et al., 2012) version 6.0, which was responsible for introduc-

ing the code instrumentation in the applications during its compilation. After that, the application tracing execution was made as usual. By the end of this step, we made a post-mortem trace analysis by first converting the original traces created in the Open Trace Format Version 2 (OTF2).

To convert the current OTF2 format traces, we used the Akypuera tools[1], more specifically the `otf22paje` tool[2]. After completing this conversion, we obtain a file with trace format, which in our case needs to be converted to a CSV with `pj_dump` tool[3]. Finally, the CSV file containing all the trace information was parsed in the R statistical language. The applications were executed with the largest number of possible/supported processes by the cluster, in this way, BT and SP used 144 processes (the number of processes must be a square root), and the rest of the applications (CG, EP, FT, IS, LU and MG) used 128 processes (the number of processes must be a power of 2).

We employ a reproducible research methodology (STANISIC; LEGRAND; DANJEAN, 2015), using R, Git, Zenodo, and a laboratory notebook. All the data that has been collected in this work is publicly available[4]. The source codes and project methodology are also available in a Git repository[5].

4.2 High-Performance Interconnects on Public Clouds

As a new important step towards the characterization of network interconnection impact on HPC application performance, in this assessment (MALISZEWSKI et al., 2020a), we are evaluating instances of the Microsoft Azure public cloud provider, which have different network interconnection options. More specifically, we carry out the analysis with *A10*, *DS4_v2*, and *F8* instance sizes using 10GbE, 40GbE InfiniBand, and 50GbE InfiniBand, respectively. Our evaluation uses three individual clusters with eight instance sizes - A10, DS4_v2, or F8. We execute synthetic representative benchmarks of the NAS Parallel Benchmarks (BAILEY et al., 1991) suite and the real Alya HPC application (VÁZQUEZ et al., 2016) with 64 processes (8 processes per instance). Following a reproducible research methodology, we analyze these applications regarding performance and cost efficiency.

[1] <https://github.com/schnorr/akypuera>
[2] <https://github.com/schnorr/akypuera/wiki/OTF2WithAkypuera>
[3] <https://github.com/schnorr/pajeng/wiki/pj_dump>
[4] <https://doi.org/10.5281/zenodo.3581280>
[5] <https://github.com/andermm/CMP223>

4.2.1 Computation Infrastructure/Experimental Setup

This Section describes the experimental setup regarding hardware/software specifications alongside the cost efficiency details used in our evaluation. We compare the sizes *A10*, *DS4_v2*, and *F8* because they have 8 vCPUs, and support different interconnections[6]; *A10* with 10 GbE, *DS4_v2* with 40GbE IB (Mellanox Technologies MT27500/MT27520 Family [ConnectX-3/ConnectX-3 Pro Virtual Function]), and *F8* with 50 GbE IB (Mellanox Technologies MT27710 Family [ConnectX-4 Lx Virtual Function] (rev 80)), and different allocation costs per hour; *A10* US$ 0.78/instance, *DS4_v2* US$ 1.008/instance, and *F8* US$ 0.792/instance).

Our experiments are performed with eight instances with sizes *A10*, *DS4_v2*, and *F8* from Microsoft Azure Public Cloud. Table 4.1 contains an overview of their hardware specification. The software specification for the instances has 64-bit Ubuntu Server 18.04 (Kernel 5.0.0-1032-azure) as OS, MPI Open MPI 2.1.1 library, GCC/GNU Fortran compiler version 7.4.0. *DS4_v2* and *F8* instance sizes used the Accelerated Networking approach of Microsoft Azure. It enables single root I/O virtualization (SR-IOV) to the instance, improving its networking performance. This high-performance path bypasses the host from the datapath, reducing latency, jitter, and CPU utilization. Besides, enabling Accelerated Network has no extra cost.

Figure 4.1 depicts the comparison of the communication between two VMs[7] with and without accelerated networking. All network traffic in and out of the VM must traverse the host and the virtual switch without accelerated networking. The virtual switch provides all policy enforcement, such as network security groups, access control lists, isolation, and other network virtualized services to network traffic. With accelerated networking, network traffic arrives at the virtual machine's network interface (NIC) and is then forwarded to the VM. All network policies that the virtual switch applies are now offloaded and implemented in hardware. Using the policy in hardware enables the NIC to forward network traffic directly to the VM, bypassing the host and the virtual switch while maintaining all the policies applied in the host (SILVA et al., 2019).

We execute ten applications covering several parallel patterns. They include the NAS Parallel Benchmarks set (IS, EP, CG, FT, MG, BT, SP, and LU) as synthetic applica-

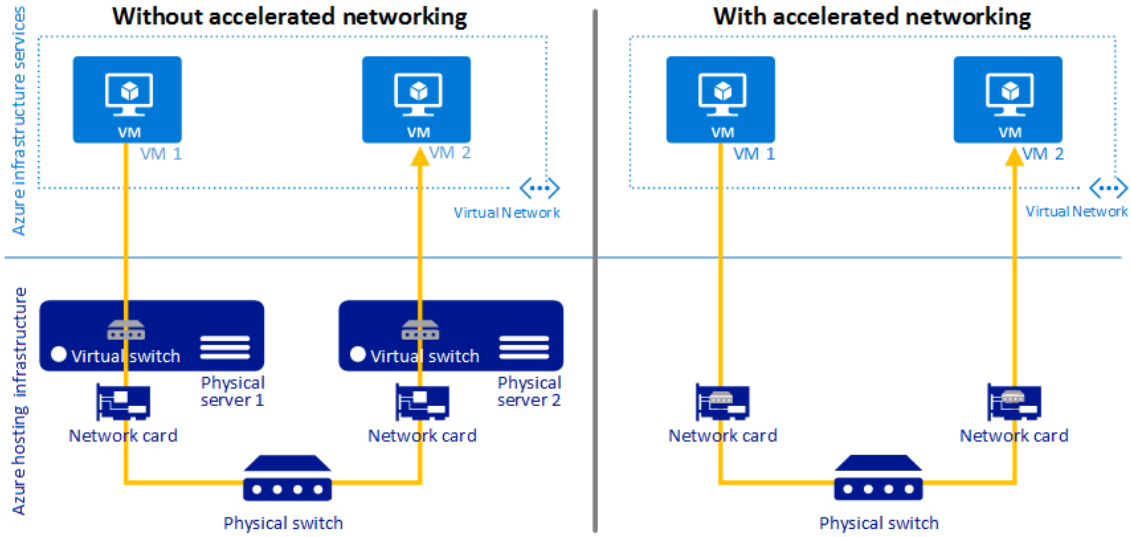
[6] Although main memory sizes vary between different instance sizes, all amounts were sufficient for our experiments and are therefore not mentioned.

[7] In this work, when we mention VMs or instances, both are referred to the same virtualized machine hosted in the cloud.

Table 4.1: Azure instances hardware specification.

Inst Size	Processor	vCPUs	Network	Inst N	Price/Hour
A10	E5-2670, 2.60GHz	8	10 GbE	8	US$ 0.78
DS4_v2	E5-2673 v3 2.40GHz	8	40 GbE IB	8	US$ 1.008
F8	8171M, 2.60GHz	8	50 GbE IB	8	US$ 0.792

Figure 4.1: Network diagram of Azure VMs, both with and without Accelerated Networking approach. Extracted from (SILVA et al., 2019).



tions, Alya as a real HPC application, and Intel MPI Benchmarks (Ping-Pong) to obtain network performance indicators. The **Ping-Pong** application (INTEL, 2014) is used for a first view of the network latency and throughput, using two processes between two nodes (one process per node), increasing message sizes varying from 1 Byte up to 4 MBytes. Both NPB applications and Alya are executed using 8 instances with up to 64 processes.

We employed a reproducible research methodology (STANISIC; LEGRAND; DANJEAN, 2015), using R, GIT[8], and a laboratory notebook, making publicly available all the data of this work. We followed a randomized full factorial experiment design (JAIN, 1991) to guide the execution of the experiments. The design has 30 replications with two factors (ten applications and three instances). The reported execution times and Ping-Pong measurements are averages of the replications, and the error bars were calculated considering a confidence level of 99.7%, assuming a Gaussian distribution.

The cost efficiency metric introduced in Roloff et al., (ROLOFF et al., 2012) was used in this evaluation to verify which cloud offers the higher performance for the price paid. It is represented by the number of executions of a determined application that could be made in an hour, divided by the hourly cost of the selected instance, in this case,

[8]https://github.com/andermm/ISCC-2020.git

the sum of the eight *A10*, *DS4_v2* or *F8* instances sizes. Equation 4.1 formalizes this definition. A higher result of this equation indicates higher efficiency.

$$CostEfficiency = \frac{\frac{1\ hour}{execution\ time[hours]}}{price\ per\ hour} \tag{4.1}$$

4.3 NIC Aggregation in Private Clouds

In this assessment, we used four synthetic applications (BT, SP, FT, IS) from the NAS Parallel Benchmarks, executed in clusters created with LXD containers in the OpenNebula manager (VOGEL et al., 2016; MALISZEWSKI et al., 2021). We make progress on the state-of-the-art by evaluating combinations of different NIC aggregation modes (802.3ad mode 4 and Balanced Round-Robin mode 0), number of NICs aggregated (four, two, and one), as well as, introducing the impact of parallel instances (four, two and baseline) executing applications.

The computational resources are equally divided between the instances. For example, with 4 simultaneous LXD instances deployed, each one had 25% of the total computation resources available. We created a methodology (see Table 4.2) which divides the executions into three environments regarding the network utilization by the applications used: High, Medium/High, and Low. This methodology was created based on previous researches (MALISZEWSKI et al., 2019; MALISZEWSKI et al., 2020b) and on its profiling (Section 4.1), in which we consider the applications BT and SP as having medium/low network utilization and the application FT and IS as having high network utilization.

4.3.1 Computational Infrastructure/Experimental Setup

The computational environment that supported the experiments was composed of four HP ProLiant servers with identical hardware resources. Each has two six-core AMD Opteron processors 2425 HE, 32GB of RAM, 4 Intel Gigabit network interface cards (NICs) interconnected by a Gigabit Switch. The software specification has Ubuntu Server 18.04 64-bit (kernel 4.15.0-99) as the operating system (OS), MPI Open MPI 2.1.1 library, GCC/GNU Fortran compiler version 7.5.0. Besides, OpenNebula cloud manager was used with version 5.10.1 and the Ethernet Channel Bonding Driver with version 3.7.1. All software involved in the evaluation process was used with their last stable available

Table 4.2: Methodology for NIC aggregation experiments.

One VM - Full Computational Resources				
Network Utilization	**Main**	**N of Parallel VMs**	**Parallel Apps**	**Result**
Base	IS	0	None	(IS)
	FT	0	None	(FT)
	BT	0	None	(BT)
	SP	0	None	(IS)
Two VMs - Computational Resources divided between the two VMs				
Network Utilization	**Main**	**N of Parallel VMs**	**Parallel Apps**	**Result**
High	IS	1	IS	(IS) + Parallel (IS)
	FT	1	FT	(FT) + Parallel (FT)
	FT	1	IS	(FT) + Parallel (IS)
	IS	1	FT	(IS) + Parallel (FT)
Medium/High	BT	1	IS	(BT) + Parallel (IS)
	SP	1	IS	(SP) + Parallel (IS)
	BT	1	FT	(BT) + Parallel (FT)
	SP	1	FT	(SP) + Parallel (FT)
Low	BT	1	BT	(BT) + Parallel (BT)
	SP	1	SP	(SP) + Parallel (SP)
	BT	1	SP	(BT) + Parallel (SP)
	SP	1	BT	(SP) + Parallel (BT)
Four VMs - Computational Resources divided between the four VMs				
Network Utilization	**Main**	**N of Parallel VMs**	**Parallel Apps**	**Result**
High	IS	3	IS	(IS) + Parallel (IS + IS + IS)
	FT	3	FT	(FT) + Parallel (FT + FT + FT)
	FT	3	IS	(FT) + Parallel (IS + IS + IS)
	IS	3	FT	(IS) + Parallel (FT +FT + FT)
Medium/High	BT	3	IS	(BT) + Parallel (IS + IS + IS)
	SP	3	IS	(SP) + Parallel (IS + IS + IS)
	BT	3	FT	(BT) + Parallel (FT + FT + FT)
	SP	3	FT	(SP) + Parallel (FT + FT + FT)
Low	BT	3	BT	(BT) + Parallel (BT + BT + BT)
	SP	3	SP	(SP) + Parallel (SP + SP + SP)
	BT	3	SP	(BT) + Parallel (SP + SP + SP)
	SP	3	BT	(SP) + Parallel (BT + BT + BT)

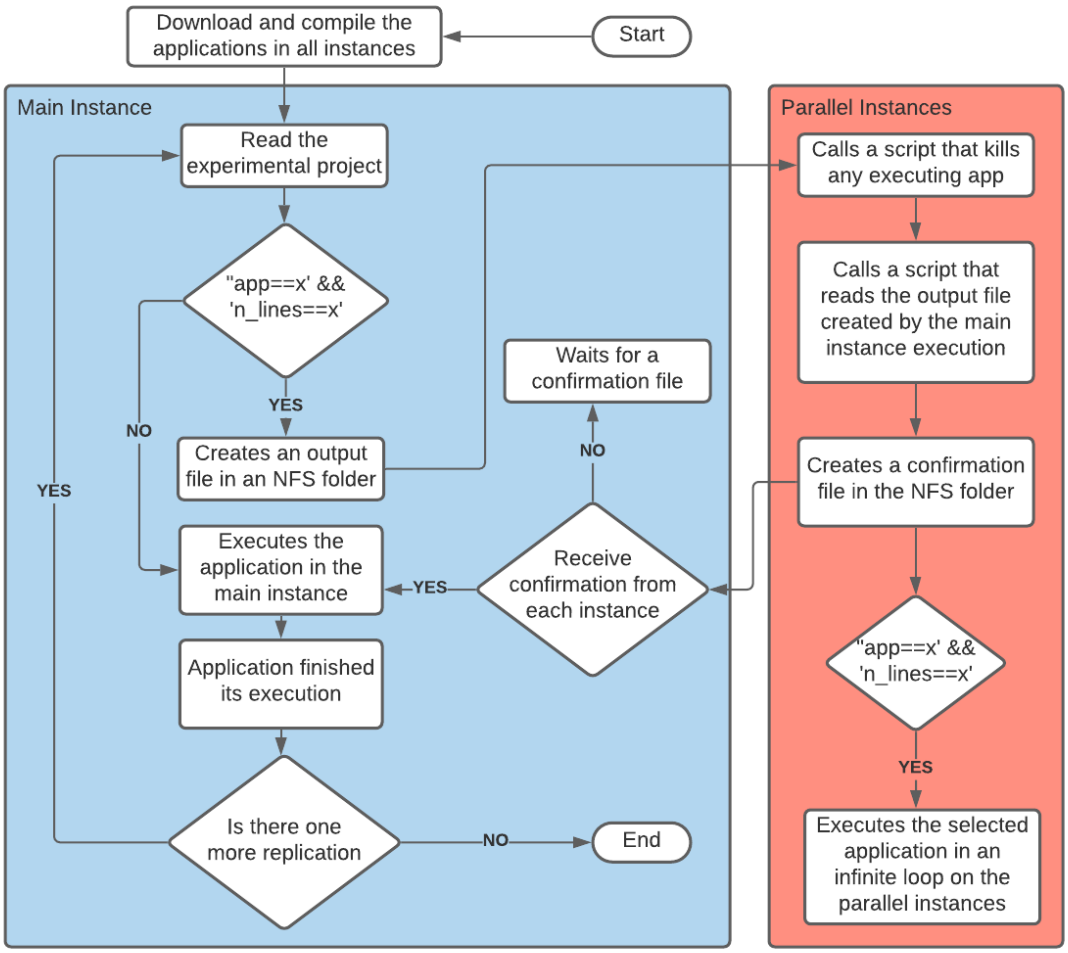
version. The LXD instances were created using the LXC version 3.0.3 and used the same OS, MPI wrapper, and GCC version as the physical servers.

To guide the execution of the experiments, we created a program to start it automatically in all nodes (main and parallel). For instance, when we pretend to evaluate the interference caused by three instances executing BT application, against the execution of BT in the main instance, our program first downloads and compile the benchmarks in all instances, then in the main instance, it reads the experiment project, which contains the applications execution order, and creates an output file in an NFS folder. Next, in the parallel nodes, it calls a script that kills any executing applications, calls a script the read the output created by the main instance in the NFS folder, creates a confirmation file in the same folder saying it is ready to cause the noise, and finally, select and execute an application in an infinite loop. After confirming that the parallel instances are executing their applications, the application will execute on the main instance. Each time it execution in the main instance finishes and changes to another, a message is sent through NFS, and the infinite execution loop in the parallel instances is killed. Finally, the program re-reads the experimental project and restart the previous steps with the correct applications. In Figure 4.2 we depict this process using a flowchart, considering the colors blue for the processes in the main instance and red for the process in the parallel instances.

Every time we ended the execution of the experiments for the baseline, each number of NICs aggregated (0, 2, and 4), and different aggregation modes (802.3ad and Balanced Round-Robin) we needed to restart the underlying servers. With the reboot process, we also ensured that there was no interference in the experiments related to various levels of cache (e.g., memory, processor instructions). The designs have 10 replications, and the reported execution times measurements are averages of the replications with the error bars calculated considering a confidence level of 95%, assuming a Gaussian distribution. We considered the results of the applications executed in the main instance. The parallel instances' results were discarded once they were used to cause the noise in the network and consequently affect or not the main instance performance.

We conduct our evaluation using four HPC benchmarks (IS, FT, BT, and SP) from the Numerical Aerodynamic Simulation Parallel Benchmarks (NPB) suite (BAILEY et al., 1991). The NPB set, used with version 3.4.1, was designed to evaluate different hardware and software performance in HPC systems. All NAS benchmarks were compiled with size C with -O3 flag, mpifort, and mpicc for Fortran and C codes. We executed the applications with two variations in the number of processes. With 16 MPI processes

Figure 4.2: Execution program flowchart.



(4 per instance) because we want to evaluate only the network concurrency, once the instances have their hardware equally divided (parallel), and with 64 MPI processes (16 per instance) because we want to add one more factor of concurrency, resulting in more processes per instance than the physical server has.

5 EXPERIMENTAL EVALUATION

In this chapter, we expose the results of three different experimental evaluations. The first one (Section 5.1), create the NPB application's profile, which will be used to understand the subsequent evaluation results. In Section 5.2 we evaluate the performance of clusters from a public cloud with three different instance sizes and different network interconnections. In addition, we also estimate if the performance could be improved coupled with cost efficiency. By the end of this chapter, in Section 5.3, to improve the performance of applications executed in clouds, we evaluate the approach of different NIC aggregation modes, which can improve the network performance, with various scenarios of parallel VMs and application combinations.

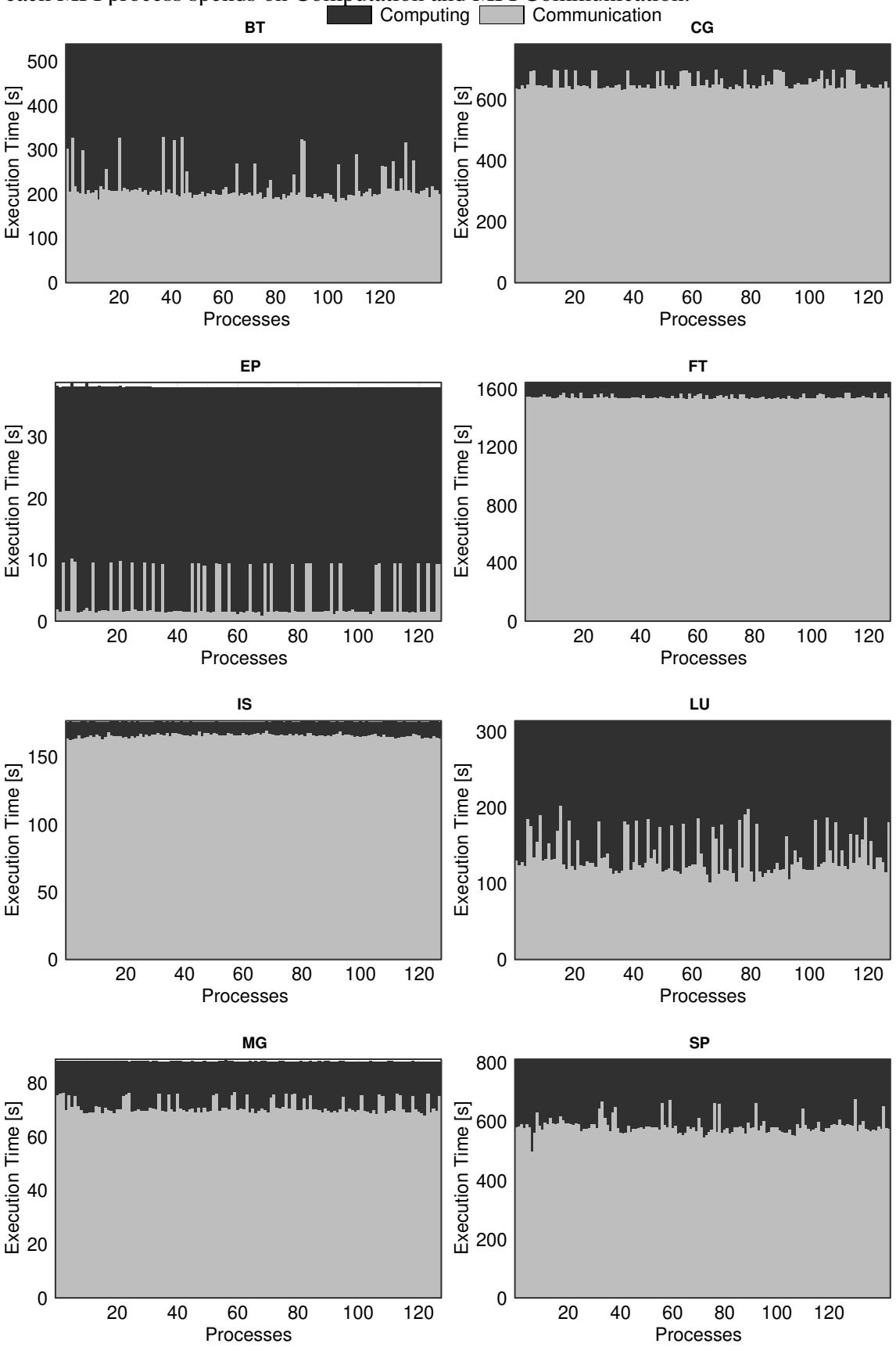
5.1 Benchmark Profiling

The results show the trace of the applications in which their MPI patterns are exposed. Below, we depict the results of each application in Figure 5.1, showing the fractions of time each MPI Rank/Process spends on Computation (in black), and MPI Communication (in gray). The MPI communication portion in each graph represents the exchange of information between hosts (over the network) and the communication between processes (at localhost). Both are due to the use of four interconnected hosts. For example, if the application is executed on only one host, there will be no network communication, and only inter-process communication will occur.

Furthermore, we cannot say which process communicates with another process. For instance, the BT application was executed with 144 processes (36 for each host). We have defined in the machine file that processes 1-36 will be placed on host X and processes 37-72 on host Y. However, we cannot say whether process 1 communicates with process 36 on host X (communication between processes) or with process 40 on host Y (communication between processes over the network). Thus, the fractions of time that each MPI Rank/Process spends in MPI Communication are generalized for communication between processes and through the network.

In the y-axis, we have the execution time in seconds, and in the x-axis, we have the number of MPI Ranks. The application's execution time can not be considered natural because the trace execution used modified the application's binaries with routines to record specific events information during the subsequent execution. Thus the execution

Figure 5.1: NPB set profiling. Rank-by-rank profiling computing the fractions of time each MPI process spends on Computation and MPI Communication.



time experience the probe effect or instrumentation overhead[1].

The first group consist of applications highly network dependent, which include IS and FT. FT is known to be Communication-Bound, and as one can see in our profiling, this characteristic is corroborated. Looking at the MPI ranks, we can observe that almost the entire execution of FT is spent doing MPI operations, with a slight portion of Computing. This application sends a considerable amount of small messages, which turns it latency-sensitive. We can highlight the Communication-Bound pattern, with ≈6% spent on Computing and ≈94% of time spent on MPI communication operations.

Similarly, IS shows a dominant behavior of having its execution almost entirely spent on MPI routines. It has the second shortest execution time of all these applications and approximately the same values in percentage as FT (MPI communication operations - ≈94%, and Computing - ≈6%). This behavior emphasizes that the performance of this application is driven by network performance (latency and throughput).

The second group illustrates the behavior of network-dependent applications. In the profile, it was identified that the CG and MG share this pattern. In CG, it is perceptible the unbalance between the ranks, with some MPI spikes. It has the usage pattern of the routines `MPI_Send` (blocking send) and `MPI_Wait` (wait for an MPI request to complete). Thus, the spikes seen in the profiling are justified. Also, it has its execution time with a high proportion of MPI routines (≈84%) in comparison to Computing (≈16%).

MG, by its time, shows a similar profile with CG. However, its execution time is considerably shorter. We also can identify some unbalance, resulting in spikes depicted in the graph. These spikes are mostly caused by the utilization of routines like `MPI_Send` (performs a standard-mode blocking send) and `MPI_Wait` (waiting for an MPI request), performing blocking operations. In this application, MPI Communication is predominant, with (≈80%) to (≈20%) of Computation.

The third group consists of BT, SP, and LU applications, which are classified as having low network dependency according to the profile created. BT and SP have similar MPI patterns, represented by the routines `MPI_Wait` (wait for an MPI request) and `MPI_Waitall` (wait for all MPI requests). In other words, in BT and SP the ranks are locked, waiting for the result of some calculation to follow its execution. This behavior can be seen in the graphs by the spikes it creates. The Computing and MPI routines division correspond (≈60%) to (≈40%) in BT, respectively. In SP the percentage of Computing represents (≈28%) to (≈72%) in MPI Communication routines. In LU, as its

[1]This overhead typically stays below 4% (KNÜPFER et al., 2012).

name suggests (**L**ower **U**pper), has its entire execution done with a predominant unbalance of processes, causing several spikes. The most usual routine in this application is the `MPI_Recv` (Standard-mode blocking receive). Finally, the division of time spent on Computing and MPI Communication corresponds to (≈56%) to (≈44%), respectively.

The last group is composed only by the EP application, described as not dependent on the network. As its name suggests (**E**mbarassingly **P**arallel), EP application profiling is dominated by computing. Also, some spikes of MPI operations are perceptible, in which EP mostly makes the MPI_Bcast and MPI_Allreduce routines to broadcast the message from the process with rank "root" to the others, and combine these values, distributing the result, respectively. EP has ≈91% of its time spent in computing and ≈9% spent in MPI communication operations. Thus, this application is considered a CPU-Bound.

5.2 High-Performance Interconnects on Public Clouds

This section presents the results starting with a first overview of the network capacity concerning latency and throughput. After, we present the application execution times (with different instance sizes and interconnections) and the cost efficiency results.

5.2.1 Network Performance

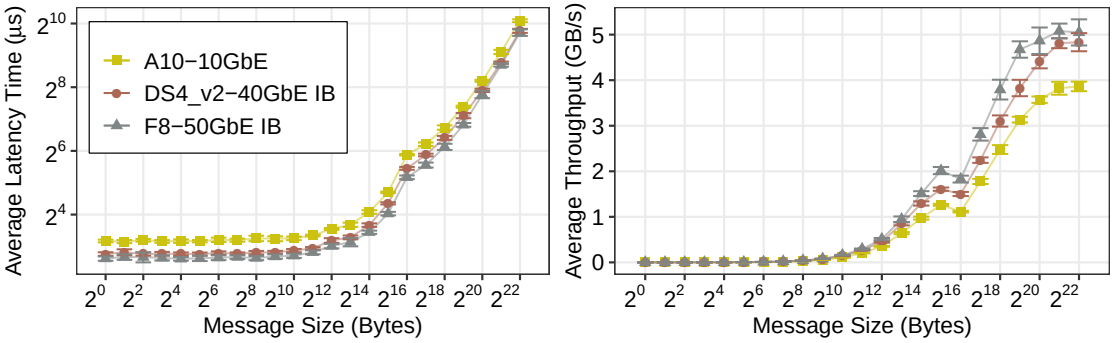
The results of interconnection latency and throughput using a Ping-Pong application are depicted in Figure 5.2(a) and Figure 5.2(b), respectively. We used these results to establish a baseline for the performance of the interconnection. Figure 5.2(a) shows the average latency (on the Y log scale) between two nodes for the three different instances/interconnections (differentiated by color and points) as a function of message size (also on the X log scale).

From these results, we can observe that the *F8* instance with 50GbE IB exhibits the best performance. With small messages (64 Bytes - 2^6) *F8* overcomes *A10* with 10GbE in ≈47%. We can not state a difference between *F8* and DS4_v2 instances because their results overlap the error bars. Also, DS4_v2 overcomes *A10* in ≈33%. With respect to larger messages (4 MBytes - 2^{22}), *F8* overcomes *A10* in ≈29%. Again, we can not state a difference between *F8* and DS4_v2 because their results overlap the error bars. Finally, DS4_v2 overcomes *A10* in ≈24%. This evidence confirms the expected results *i.e.*, In-

finiBand instances are reaching the best latency performance compared to Ethernet. This is because when we use Infiniband interconnection, we experience the impact of RDMA (Remote Direct Memory Access), which reads data directly from the main memory of one server and writes it directly to the main memory of another server, without involving processor, cache or either operating system (OS).

Figure 5.2(b) depicts the average throughput (on Y), as a function of message size (with X on log scale). We can highlight that IB instances achieve higher throughput starting with message sizes of 2^{14} Bytes. Smaller messages have their error bars overlap. With the larger message sizes (4 MBytes - 2^{22}), *F8* instance overcomes *A10* in ≈30%. DS4_v2 overcomes *A10* in ≈25%. Finally, we can not state a difference between *F8* and DS4_v2 instance because their error bars overlap. This result follows the same obtained for the latency, in which IB instances reach higher throughput.

Figure 5.2: Network performance results.



(a) Average network latency (on the Y log scale) measured with the Ping-Pong application for the three instance sizes/interconnections as a function of message sizes (also with the X log scale). In terms of latency, lower results are better.

(b) Average network throughput (on the Y) measured with the Ping-Pong for the three instance sizes/interconnections (with the X log scale). In terms of throughput, higher results are better.

5.2.2 HPC Applications Performance

Our performance evaluation results are presented in Figure 5.3 concerning Alya, BT, CG, EP, FT, IS, LU, MG, and SP applications. In Y-axis, we set the logarithm 10 scale because the execution times' absolute values vary widely between the benchmarks (e.g., IS and MG with mean execution times are < 20 and SP and FT are > 1000 seconds). On the X-axis is the name of the application. To a better overview of the absolute values of this graph, we represented them in Table 5.1.

We classified the results into four groups according to their performance gains.

Group 1 has the IS and FT applications. IS shows the higher gain using the *F8* instance (up to ≈491% compared to the *A10* instance and ≈28% compared to the DS4_v2). FT shows a significant gain (up to ≈373% using 50GbE IB compared to the 10 GbE. These applications are executed based on the routines `MPI_Alltoallv` and `MPI_Alltoall` (sends data from all to all processes). Thus, a low latency obtained through a high-performance interconnect has an extreme impact on their performance.

Group 2 has CG, MG, and SP applications. MG application shows a significant performance improvement of ≈256% executing in the *F8* compared to the *A10*, and of ≈155% compared to the DS4_v2. CG executing in the *F8* instance improved its performance in ≈252% compared to the *A10* instance, and in ≈33% compared to the DS4_v2. SP improved its performance in ≈245% using the 50 GbE IB compared to the 10 GbE and in ≈185% compared to the 40 GbE IB. These applications have in common the use of `MPI_Send` (performs a blocking send) and `MPI_Wait` (wait for an MPI request to complete) routines. For this reason, even if they manage to obtain performance improvements, the impact of the interconnection is lower when compared to the first group.

The third group is composed of Alya, BT, and LU. Alya shows a performance improvement of ≈74.3% executing in the *F8* instance compared to the *A10* instance and of ≈12.3% compared to the DS4_v2. BT improved its performance in ≈92.3% executing in the *F8* instance compared to the *A10* instance, and in ≈39.4% compared to the DS4_v2. LU improved its performance in ≈96% executing in the *F8* compared to the *A10* instance and in ≈44% compared to the DS4_v2. These applications have similar MPI routines, represented by the `MPI_Wait` (wait for an MPI request) and `MPI_Waitall` (wait for all MPI requests). In LU, the most representative MPI routine is `MPI_Recv` (blocking receive for a message). This is the last group of applications for which we observe some gain executed with an instance that has an HPC interconnect.

Group 4 is composed of EP, which is the only application that decreases its performance executing in the *F8* instance. Slightly better performance can be seen when EP is executed in the DS4_v2 instance, of ≈7% compared to the *A10* and of ≈9.2% to the *F8*. This result is expected because these applications are CPU-Bound, and a faster interconnection may not provide considerable gains in performance.

Figure 5.3: Performance results in seconds (less is better) of the Alya and NPB set using Class D executed on the three different instances size (*A10*, *DS4_v2* and *F8*)/interconnections (10GbE, 40GbE IB and 50GbE IB).

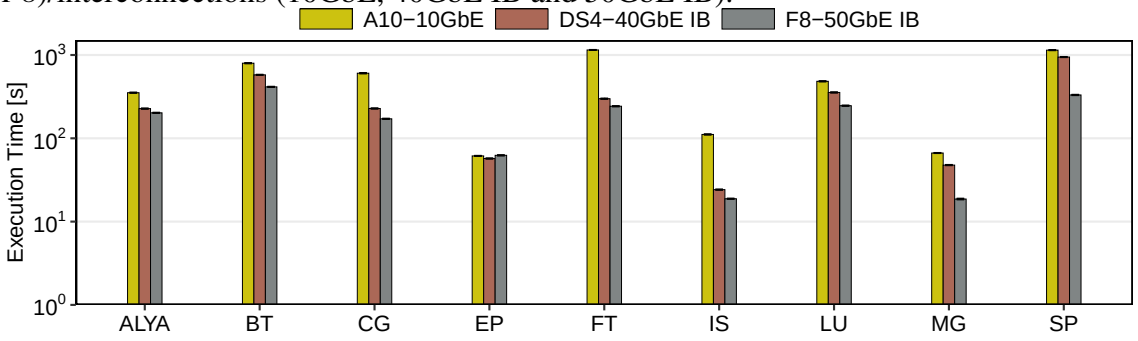


Table 5.1: Absolute values of the performance results are shown in Figure 5.3. The first column has the applications followed by its execution time mean (in seconds) and confidence interval.

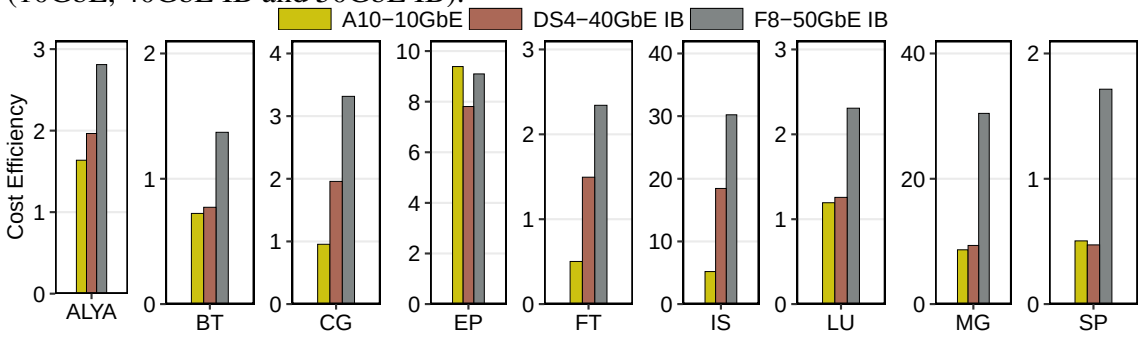
Apps	A10	DS4_v2	F8
ALYA	352.47 (±0.45)	227.19 (±0.80)	202.19 (±0.50)
BT	797.36 (±0.80)	577.84 (±0.44)	414.45 (±0.62)
CG	604.41 (±4.07)	227.96 (±0.18)	171.38 (±0.46)
EP	61.46 (±0.08)	57.18 (±0.13)	62.46 (±0.22)
FT	1,149.06 (±1.55)	298.66 (±0.04)	242.62 (±0.24)
IS	111.23 (±0.23)	24.19 (±0.05)	18.81 (±0.02)
LU	483.26 (±0.47)	355.10 (±0.39)	246.22 (±0.57)
MG	66.59 (±0.11)	47.71 (±0.20)	18.67 (±0.20)
SP	1,144.02 (±2.28)	945.12 (±2.65)	331.33 (±1.56)

5.2.3 Cost Efficiency

As previously described (see Section 4.2.1), we used *A10* (10 GbE), *DS4_v2* (40 GbE IB), and *F8* (50 GbE IB) Microsoft instances sizes. The hourly cost of one *A10*, *DS4_v2*, and *F8* instance is US$ 0.78, US$ 1.008, US$ 0.792. Our experiments were conducted by using eight instances, then we have the total costs of US$ 6.33/hour (*F8*), US$ 8.064/hour (*DS4_v2*) , and US$ 6.24/hour (*A10*). In Figure 5.4 we present the application cost efficiency results.

As one can see, eight (Alya, BT, CG, FT, IS, LU, MG, and SP) of nine applications have a significantly better cost efficiency when executed in the *F8* instance. The only application that does not have better cost efficiency in the *F8* instance is the EP, mainly because this application is CPU-Bound. The EP application is the best cost efficient when executed in the *A10* instance. We can highlight the best cost efficiency in *F8* are observed for MG, IS, and CG. Finally, *DS4_v2* instance has no better cost efficient applications that are executed on it (when compared to *F8*) and can also be slightly worst for the FT case. This result is due to the *DS4_v2* higher cost when compared to the other two instances.

Figure 5.4: Cost efficiency (higher is better) of the Alya and NPB set using Class D executed on the three different instance sizes (*A10*, *DS4_v2* and *F8*)/interconnections (10GbE, 40GbE IB and 50GbE IB).



5.3 NIC Aggregation in Private Clouds

Below we depicted the results of the experiment's methodology in line graphs. There are three line types, representing the number of VMs (One VM and just the execution of the application, Two VMs with one main VM and one parallel VM execution, and Four VMs with one main VM and three VMs each one executing an application).

In the y-axis, we have the execution time in seconds, and in the x-axis, we have

the aggregation mode, and the number of NICs aggregated, being the first number the aggregation representation (0 for Balanced Round-Robin and 4 for 802.3ad), and the second number, the quantity of NICs aggregated. The “None” bar legend represents the baseline (executed in the LXD without NICs aggregated). The shape points are individual for each line type and indicate the execution time of the application on each environment (Aggregation Mode + Number of NICs Aggregated). Finally, each figure is divided by two graphs: the first (left) the execution of the application with 16 processes and the second (right) the execution of the application with 64 processes.

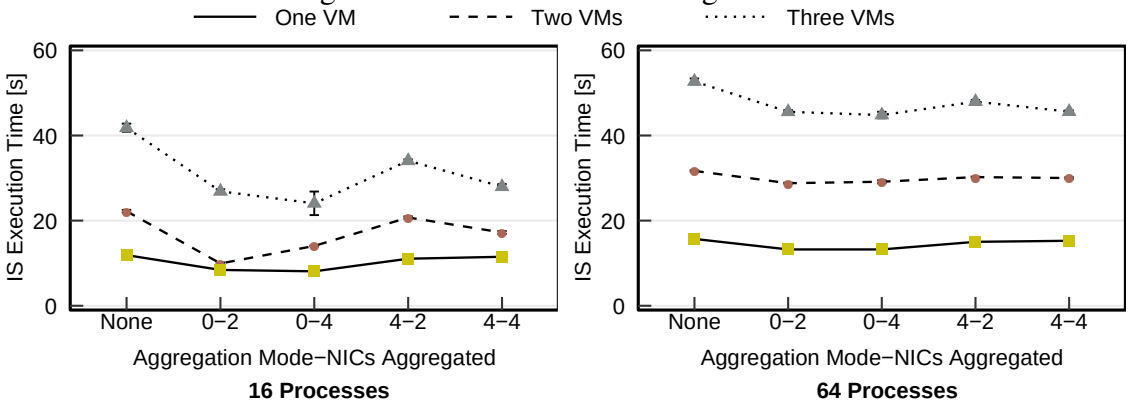
5.3.1 High Network Utilization

The first group is the one that makes high network usage. Between them are four combinations executing the applications IS and FT with 16 (in left) and 64 (in the right) processes. Due to the characteristics of this group, it is expected that the utilization of the NICs aggregation approach, increases the performance if compared to the baseline, with no aggregation, due to the improvement of network performance. The first combination (seen in Figure 5.5) uses the IS application as the main execution (which times are represented in the graph), against the noise caused by the same application executed in parallel VMs. After analyzing the executions with 16 processes, we have the first points of the three environments starting from the “None” configuration, representing the baseline (without NIC aggregation) for each of the three environments (One VM, Two VMs, Four VMs). Using the Balanced Round-Robin aggregation mode, it is significant perceptible improvements compared to the baseline. With two and four NICs aggregated, the performance gains that stand out is the environment with four VMs, which have gained in ≈55%, and ≈73%, respectively, compared to the baseline. RR with one VM shows some performance improvements compared to the RR mode with 2 NICs. However, the outstanding result is the loss of performance with two VMs compared to the aggregation with 2 NICs, which performed in ≈41% better.

In the 802.3ad mode (mode 4) with 2 NICs, all three environments have gained performance compared to the baseline. Similarly to RR, the environment that presents the higher gains is with four VMs (in ≈22% compared against baseline). Finally, the same mode but with 4 NICs aggregated, present improvements with four and two VMs, compared to 802-3ad 2 NICs (≈21% and ≈20%, respectively). Also, the result with just a single VM presents a slight loss of performance compared against mode 4 - 2 NICs.

In the same evaluation but with 64 processes, we see the execution times more linear and fewer representative gains than with 16 processes. For instance, in both environments, with one and two VMs, the performance gains are in general less than 3 seconds compared to the baseline. Also, this improvement is made on the RR mode, which again overcomes the 802.3ad mode. The environment with the most representative gains is the one with four VMs. Compared to the baseline, mode 0 with 2 and 4 NICs have performed better in ≈15%, and ≈17%, respectively. Mode 4 with 2 and 4 NICs overcomes the baseline in ≈9% and ≈15%, respectively. When we look at the full picture, both aggregation modes present better results than baseline, and also, the RR aggregation mode presents better results than 802.3ad in all environments.

Figure 5.5: IS execution times against IS.



The second combination, with results represented in Figure 5.6, executes IS as the main application, and in the parallel instance is executed FT. When it was executed with 16 processes and using the RR mode and 2 NICs, we had significant gains compared to the baseline, mainly in the environment with 4 VMs (gain of ≈73%). Also, with two VMs, it reached a gain of ≈25% and got close to the result with one VM. With 4 NICs and the same mode, the only environment with expressive gains is four VMs that improved ≈24% compared to the RR-2 NICs. With one VM, it has slight improvements. However, in the environment with two VMs, the performance has decreased ≈34% compared with the same aggregation mode but with 2 NICs aggregated.

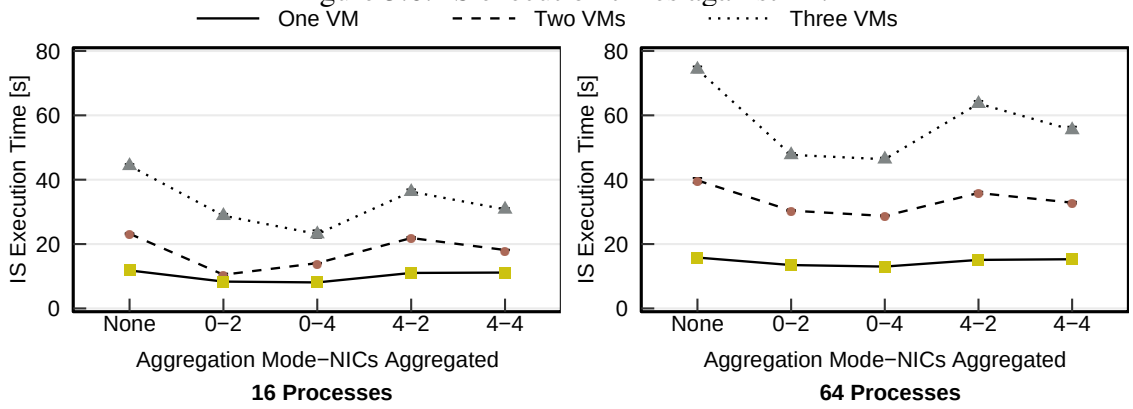
The execution of IS with 802.3ad mode and two NICs has overcome the baseline in the three environments (≈7% with one VM, ≈6% with two VMs, and ≈22% with four VMs). With 4 NICs and the same mode, both parallel environments have increased their performance, in ≈17% with 4 VMs, and in ≈20% with 2 VMs. On the other hand, the performance has slightly reduced with one VM compared to the 802-3ad-4 NICs.

Looking at the execution results with 64 processes, we see more significant gains

than the 16 processes execution, mainly in the environments with parallel VMs. For instance, the gain of using RR mode with two NICs reaches ≈55% with four VMs, and ≈30% with two VMs, while there is a lower gain of ≈17% with one VM compared to the baseline. In the usage of 4 NICs and RR mode, all three environments presented slightly gains of performance, in ≈2% with four VMs, in ≈5% with two VMs, and ≈3% with one VM, compared to the RR - 2 NICs.

With 802.3ad mode and 2 NICs, the applications performed better than the baseline in the three environments (gain of ≈4% with one VM, ≈10% with two VMs, and ≈16% with four VMs) compared to the baseline. Furthermore, with 4 NICs aggregated, the performance of with 4 and two VMs have overcome the aggregation with 2 NICs and the same mode in ≈14% and ≈9%, respectively. The performance with one VM has slightly decreased (less than 15 milliseconds). When we compare the whole picture, IS has performed better with both aggregation modes than the baseline in all environments. Also, the best aggregation mode (in terms of obtained performance) is the Balanced Round-Robin.

Figure 5.6: IS execution times against FT.



The third combination of the high network usage methodology executed FT as the main application against the same application executed in parallel VMs. As can be seen in Figure 5.7, when using the Balanced Round Robin mode with 2 NICs aggregated all three environments presented performance improvements (compared to the baseline), in ≈40% with one VM, in ≈114% with two VMs, and in ≈48% with four VMs. These performance improvements are still increasing with four VMs when using mode 0 with 4 NICs compared with 2 NICs (in ≈13%). On the other hand, with two VMs, the performance decreases in ≈15%, while with one VM it slightly increases in ≈2%.

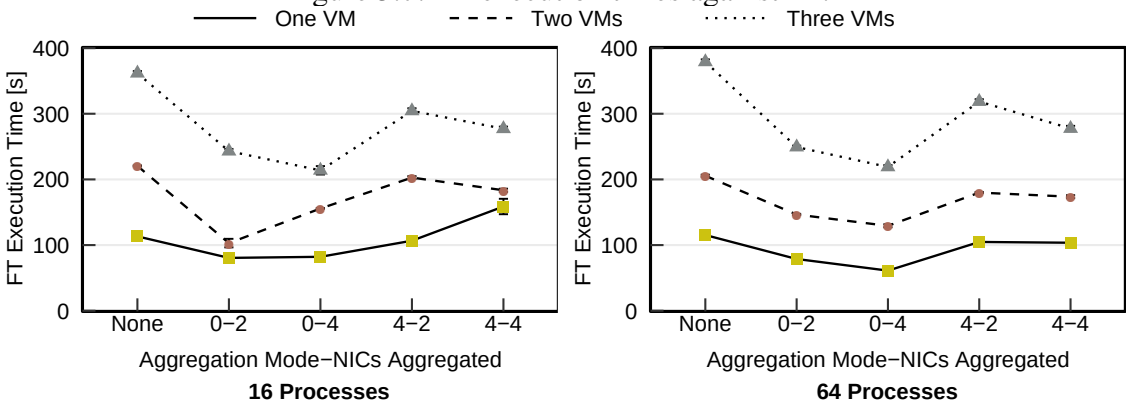
The usage of 802.3ad mode with two NICs aggregated improved the performance of the applications, compared to the baseline, mainly with four and two VMs environment

(in ≈18% and in ≈8%, respectively), while with one VM we had slightly performance improvements in ≈6%. With the last configuration (mode 4 and 4 NICs), the performance with one VM has significantly decreased in ≈48%, while the performance of both 4 and two VMs environments had increased its performance in ≈9%, and ≈10% (concerning the mode 4 with 2 NICs). With 16 processes, only the execution with 802.3ad and 4 NICs aggregated not overcame the baseline between both aggregation modes.

FT execution with 64 processes (right side), the environment with RR-2 NICs presented significant gains in all three environments compared to the baseline (≈46% with one VM, ≈40% with two VMs, and ≈52% with four VMs). These performance gains continued when we used the RR mode with four NICs aggregated (≈29% with one VM, ≈13% with two VMs, and ≈13% with four VMs), compared to the RR with 2 NICs.

Following the gain of RR mode, the 802.3ad mode with 2 NICs presented performance improvements in all three environments (≈9% with one VM, ≈14% with two VMs, and ≈18% with four VMs) against the baseline. Finally, when we used this same aggregation mode with four NICs, the execution times were better in all three environments than mode 4-2 NICs. The higher performance gains are obtained in more parallel environments. In both aggregation modes with 2 and 4 NICs aggregated, the performance overcomes the baseline results in ≈96% of the cases, and the Balanced Round Robin aggregation mode overcomes 802.3ad mode in all three environments.

Figure 5.7: FT execution times against FT.



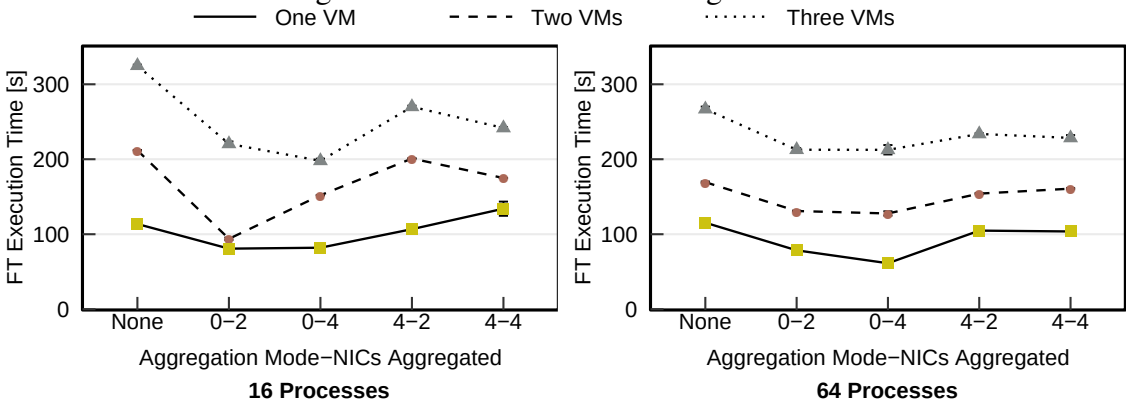
The last combination (depicted in Figure 5.8) of this methodology is the execution of FT as the main application, experimenting with the network noise caused by the execution of IS in the parallel instances. With 16 processes, the execution of FT with RR mode was better than the baseline in all three environments (in ≈40% - one VM, ≈125% - two VMs, ≈47% - four VMs). When we increased the number of NICs to 4 with this mode, performance with four VMs (in ≈11%) environment still having performance im-

provements, compared to the RR-2 NICs. On the other hand, with one VM and with two VMs, the performance decreased in ≈1%, and ≈61%, respectively.

Comparing the 802.3ad mode against the baseline, we perceive that the three environments increased their performance with the aggregation (in ≈6% - one VM, ≈5% - two VMs, and ≈20% four VMs). These results were improved when four NICs were used for both parallel environments (≈14% to two VMs and ≈11% to four VMs). Contrary, the environment with a single execution decreased its performance against the previously used number of NICs, and also compared to the baseline.

With 64 processes, the performance has been improved using mode 0 with 2 NICS compared to the baseline (in ≈46%). With mode 4 using 2 NICs, the performance also improved compared to the baseline in all environments (≈9% - one VM, ≈9% - two VMs, and ≈14% - four VMs). However, with four NICs, the performance has slightly increased with one VM (in ≈1%) and with four VMs (in ≈2%), and also decreased with two VMs (in ≈4%) compared with two NICs aggregation. For both 16 and 64 processes execution, the only aggregation mode which has decreased the performance against native was mode 802.3ad with 4 NICs in the environments with one VM and 16 processes. Comparing the aggregation's modes, RR overcomes 802.3ad in all three environments.

Figure 5.8: FT execution times against IS.



5.3.2 Medium/High Network Utilization

According to our profile, the second group is the one that makes medium/high network usage. Four combinations are executing the applications BT and SP in the front VM and IS or FT in the parallel VMs, with 16 (in left) and 64 (in right) processes. Based on the previously verified characteristics, it is expected that the application can slightly

improve or decrease its performance using NIC aggregation compared to the baseline. The first combination of this group is composed by the execution of BT in the front VM against IS in the parallel VMs, depicted in Figure 5.9. As can be seen, BT executed with 16 processes has slightly improved its performance when used mode 0 with 2 NICs in the environment with one VM ((in ≈3%)) and with four VMs (in ≈5%).

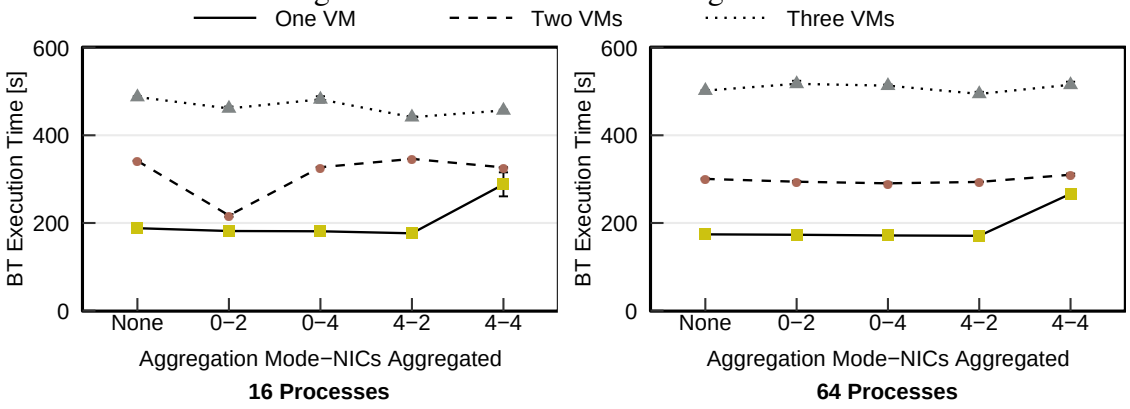
On the other hand, with two VMs, the application's performance improved (in ≈57%), compared to the baseline. With four NICs and the same aggregation mode, the only environment with slight performance improvements is with one VM (in ≈0.3%), compared with 2 NICs aggregated. Both environments with two and four VMs had decreased their performance in ≈33% and in ≈4%, respectively.

BT was executed in environments with 802.3ad aggregation mode, and it has performed better than the baseline with one VM in ≈6% and with four VMs in ≈10%. However, the environment with two VMs has slightly decreased its performance in ≈1%. Finally, when using mode 4 with four NICs, the performance decreased significantly with one VM (in ≈38%) and slightly decreased with four VMs. On the other hand, with two VMs, performance slightly overcomes the same mode with two NICs.

In the execution with 64 processes, the linearity of the results stands out. Slight differences are seen when using RR mode with two and four NICs with one VM and with two VMs, while with 4, a loss of performance is noticeable (in ≈3% compared to baseline and in ≈2% with RR-4 (compared to RR-2). With the other aggregation mode, almost no differences are seen with one VM when using two NICs. However, with two four VMs, the performance has slightly improved (in ≈2%, and ≈1%) compared to the baseline. In the end, using mode 4 with 4 NICs, the performance of BT has decreased in all three environments, mostly seen with one VM (in ≈35% compared to the aggregation with mode 4 and 2 NICs). In both 16 and 64 process executions, the execution times are better in ≈87% of the cases when NIC aggregation was used. Also, in comparing the two aggregation modes, 802.3ad has better results than RR in ≈58% of the cases (comparing the modes in the same environment and number of NICS).

In Figure 5.10 is depicted the execution of BT in the front VM against FT in the parallel VMs. As can be seen, when using mode 0 with two NICs aggregated and 16 processes, the performance has improved in all three environments, emphasizing the two VMs result (≈56% better than the baseline). On the other hand, when we add two more NICs (mode 0 with four NICs), the performance has decreases in both environments with parallel VMs (≈32% with 2 and ≈9% with 4) compared with mode 0 and 2 NICs.

Figure 5.9: BT execution times against IS.



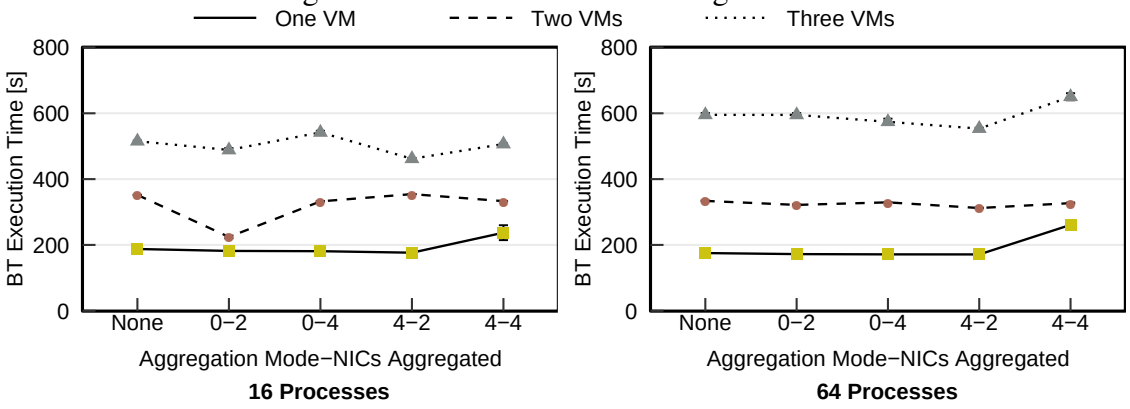
Using the 802.3ad mode with 2 NICs (4-2) makes the performance improves with one VM (≈6%), and with four VMs (≈11%. On the contrary, with two, the performance has slightly decreased in ≈0.4% (compared against the baseline). In the same aggregation mode but with four NICs, performance compared against with two NICs has decreased with one VM (≈25%) and four VMs (≈8%). Finally, with two VMs, performance overcame mode 4 with two NICs in ≈6%.

Executing the same application with 64 processes, performance improved when using mode 0 with 2 NICs in all environments, mostly seen with one ((≈1%), and with two VMs (≈3%) compared against the baseline. With mode 0 and 4 NICs compared against mode 0 with 2 NICs, two environments improve performance mainly with four VMs (in ≈3%). On the other hand, with two VMs, performance decreased in ≈2%.

Mode 4 with two NICs has improved its performance compared to the baseline in all three environments (≈2% with one VM, ≈6% with two VMs, and ≈7% with four VMs). On the other hand, when we applied mode 802.3ad with four NICs, the results experienced performance losses in the three environments in comparison with the same mode and two NICs (≈34% with one VM, ≈4% with two VMs, and ≈14% with four VMs). In general, when we used aggregation to improve performance, it presents better results than the baseline in ≈79% of the cases. Moreover, comparing aggregation modes 802.3ad overcomes RR in ≈58% of the results in the same environments.

The third combination is presented in Figure 5.11, which shows the execution time of SP against IS executed in parallel VMs. Using the aggregation mode 0 with 2 NICs (16 processes execution) has improved its performance in all three environments in comparison to baseline (in ≈4% with one VM, ≈36% with two VMs, and ≈19% with four VMs). Increasing the number of NICs to four and using the same aggregation mode, turns the performance losses with two (in ≈19%) with two and four (in ≈2%) parallel

Figure 5.10: BT execution times against FT.



VMs compared against the same mode with two NICs.

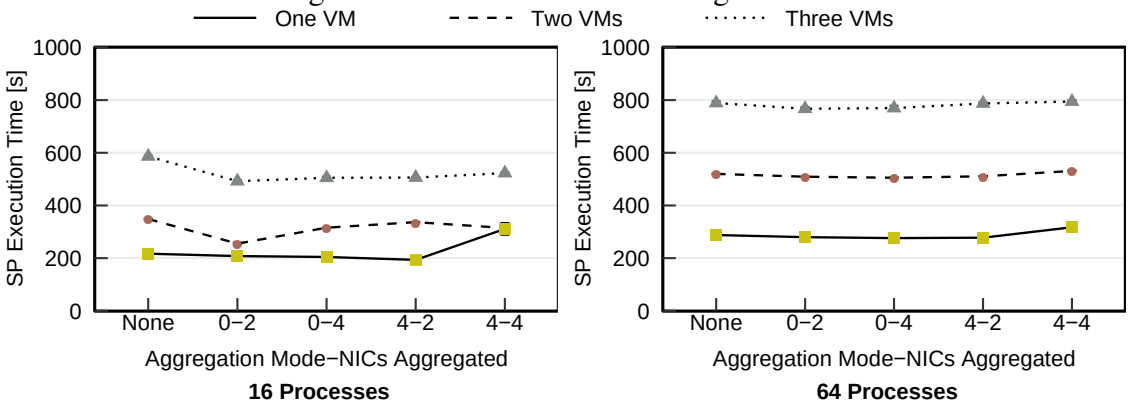
The performance improved in all environments (in ≈12% with a single VMs, ≈3% with two VMs, and ≈15% with four VMs) when we used the 802.3ad mode with two NICs in comparison to the baseline. Using mode 4 with 4 NICs, the performance has decreased with one (in ≈37%), and with four VMs (in ≈3%) in comparison against mode 4 and 2 NICs, while with two VMs, the performance has been improved (≈6%).

In the execution with 64 processes, performance increased when using Balance Round Robin mode with two NICs in comparison with the baseline in all three environments (≈2% with a single VM, ≈2% with two VMs, and ≈2% with four VMs). With four NICs aggregated and the same mode, performance slightly increased with one VM (in ≈1%) and with two VMs (≈1%) compared to the same mode but with two NICs. On the other hand, with four VMs, the performance slightly decreases.

With mode 802.3ad and 2 NICs, the performance of all three environments overcome the baseline (in ≈3% with one VM, ≈1% with two VMs, and ≈0.2% with four VMs). On the contrary, using the same mode but with 4 NICs aggregated, the performance has decreased on all three environments (in ≈12% with one VM, ≈3% with two VMs, and ≈0.9% with four VMs) in comparison to the same mode with two NICs. In general, performance has increased in ≈84% of the cases that we applied an aggregation mode against the baseline. Also, the Balance Round Robin overcomes 802.3ad mode in ≈60% of the cases, comparing in the same environment.

The last combination of this group is seen in Figure 5.12. In it, we executed SP application in the front VM against FT in parallel VMs. With 16 processes, using the first aggregation mode (mode 0) and 2 NICs, SP performance was improved in all three environments (≈4% with one VM, ≈34% with two VMs, ≈21% with four VMs) in comparison to the baseline. Following the results with the same aggregation mode but

Figure 5.11: SP execution times against IS.



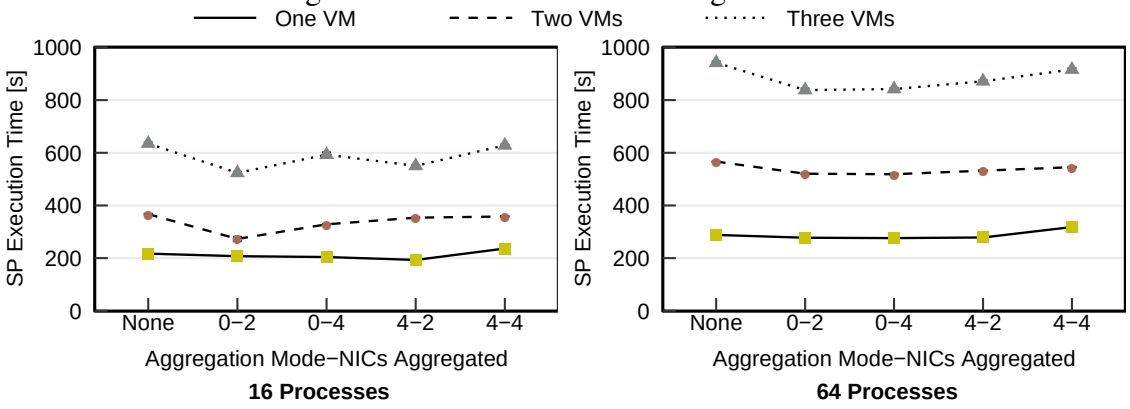
with four NICs, performance decreased in two (in ≈16%) and four (in ≈11%) parallel VMs compared to the same mode but with two NICs. On the other hand, with one VM, performance increased in ≈1%.

In turn, using mode 802.3ad with two NICs aggregated increased the performance in all three environments (in ≈12% with a single VM, ≈3% with two VMs, and ≈15% with four VMs) compared against the baseline. Next, with the same mode and four NICs, all environments decreased their performance in comparison against the 4 and 4 NICs (in ≈18% with one VM, ≈1% with two VMs, and ≈12% with four VMs).

Looking at the right side graph with 64 processes used in the execution, we see performance improvements in all three environments with mode 0 and 2 NICs (in ≈3% with a single VM, ≈8% with two VMs, and ≈12% with four VMs) in comparison against the baseline. In the execution of the same mode with four NICs compared to mode 0 with 2 NICs, performance has been slightly improved with one VM and two VMs. On the contrary, with four VMs, performance has slightly decreased.

In the execution with mode 802.3ad and two NICs, performance overcomes baseline in all environments (≈3% with one VM, ≈6% with two VMs, and ≈8% with four VMs). On the contrary, with mode 4 and four VMs, performance decreased in all environments (≈12% with a single VM, ≈2% with two VMs, and ≈4% with four VMs). In general, the performance obtained using aggregation modes is better in ≈96% of the cases than the baseline. Finally, comparing both aggregation modes (with the same number of NICs and environment), Balance Round Robin overcomes 802.3ad in ≈91% of the cases.

Figure 5.12: SP execution times against FT.



5.3.3 Low Network Utilization

According to the application's profile, the third group is the one that makes low network usage. The combinations with BT and SP are executed with 16 (in left) and 64 (in right) processes. It is expected that in this group, the utilization of aggregation modes can not improve the performance in the majority of the executions in comparison to baseline. The first combination is depicted in Figure 5.13, in which BT application is executed in the front VM against the same application executed in parallel VMs.

As can be seen in the execution with 16 processes, using the mode 0 with 2 NICs slightly improved the performance with one VM (in ≈3%) and with a higher gain with two VMs (in ≈67%), compared against the baseline. Adding two more NICs to this aggregation mode (mode 0 with 4 NICs) slightly improved the performance with one VM (in ≈0.3%). However, with two and four VMs, the performance decreased in ≈39%, and in ≈1%, respectively, compared to the same aggregation mode but with two NICs.

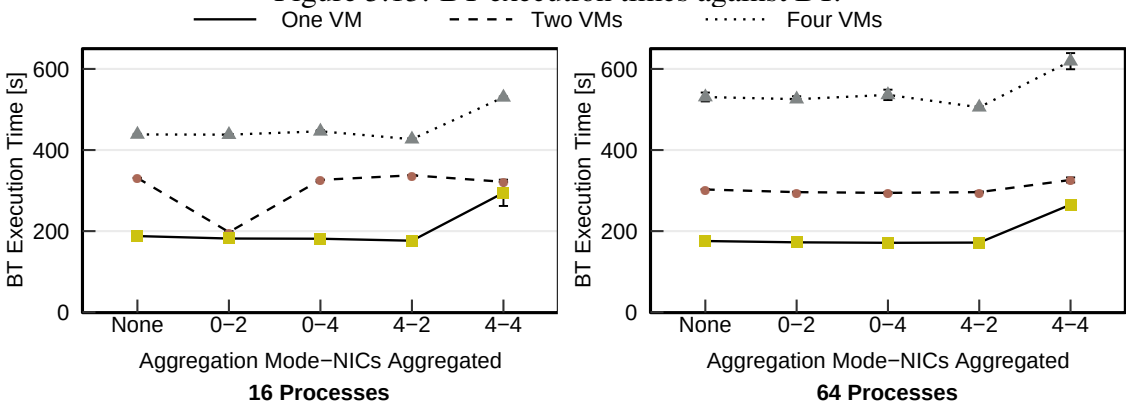
With the same number of processes, but using mode 4 with 2 NICs aggregated, performance has increased with one VM in ≈6%, and with four VMs in ≈2%. With two VMs, it has slightly decreased (≈1%) in comparison to the baseline. With the same mode and four NICs aggregated, compared with mode 4 with 2 NICs, performance has significantly decreased with one VM (in ≈40%) and four simultaneous VMs (in ≈19%). On the other hand, with two VMs, performance slightly improved in ≈5%.

In the execution with 64 processes with mode 0 and 2 NICs, performance has slightly improved in the three environments (with a single VM (in ≈1%), with two VMs (in ≈2%), and with three VMs (in ≈1%)) in comparison to the baseline. Increasing the number of NICs aggregated to four and keeping the same aggregation mode has slightly

increased the performance with one VM and two VMs. On the contrary, with four VMs, performance has decreased (in ≈1%) compared to the same mode and 2 NICs.

In the execution with mode 4 and 2 NICs, the performance of all three environments overcomes the baseline (with one VM in ≈2%, with two VMs in ≈2%, and with four VMs in ≈5%). Finally, with mode 4 and 4 NICs, significant performance decrease in all environments (with a single VMs in ≈35%, with two VMs in ≈9%, and with four VMs in ≈18%) in comparison to the same mode with two NICs aggregated. With of both aggregation modes, performance has improved compared to the baseline, in ≈66% of the cases. Also, 802.3ad aggregation mode performed better than Balance RR mode in ≈58% of the cases compared to the same environments and number of NICs aggregated.

Figure 5.13: BT execution times against BT.



The second combination uses BT executed in the main VM and SP in the parallel VMs, with results depicted in Figure 5.14. In the execution of these applications with 16 processes and mode 0 with 2 NICs, performance has increased in all three environments (≈3% with a single VM, ≈65% with 2 VMs, and ≈1% with four VMs) in comparison to the baseline. Using the same mode but with four NICs, performance has decreased in ≈37% with two VMs, and ≈5% with four VMs. With a single VM, it has slightly increased in comparison to the same mode with 2 NICs.

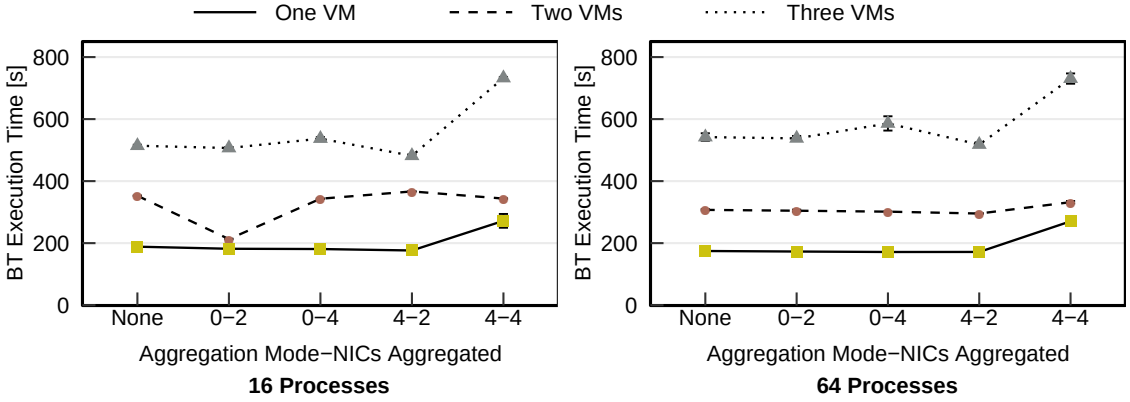
With 802.3ad mode and two NICs compared to the baseline, performance has improved with a one VM in ≈6%, and with four VMs in ≈6%. On the other hand, with two VMs, it has decreased in ≈3%. Finally, using the same mode with four NICs, makes performance increase in ≈6% only with two VMs. With one and four VMs, it has decreased in ≈35%, and ≈34%, respectively, compared with mode 4 - 2 NICs.

In this combination with 64 processes, performance has slightly increased in all three environments (less than 1%) in comparison to the baseline using RR mode with 2 NICs. Increasing the number of NICs to four with the same mode makes performance

slightly improve with a single VM and two VMs, compared to mode 0 with two NICs. On the other hand, with four VMs, performance decreased in ≈8%.

In the execution with mode 4 and 2 NICs, performance overcome baseline in all three environments (in ≈1% with a single VM, in ≈4% with two VMs, and in ≈4% with four VMs). Finally, using the same mode but with four NICs aggregated, performance has decreased in all three environments (in ≈36% with one VM, ≈11% with two VMs, and ≈29% with four VMs). In general, the performance obtained by using an aggregation mode has outperformed baseline in ≈66% of the cases. Also, Balance Round Robin mode has overcome 802.3ad mode in ≈58% of the cases.

Figure 5.14: BT execution times against SP.



The third combination is the execution of SP against the execution of the same application in parallel VMs. As can be seen in Figure 5.15, with 16 processes and mode 0 - 2 NICs, performance has increased with one VM (in ≈4%) and with two VMs (in ≈33%), while with four VMs performance has slightly decreased in comparison to the baseline. Increasing the number of aggregated NICs to four in the same mode, performance has increased only with a single VM (in ≈1%). With two and performance has decreased in ≈22%, and with four parallel, it slightly decreased compared to RR and 2 NICs.

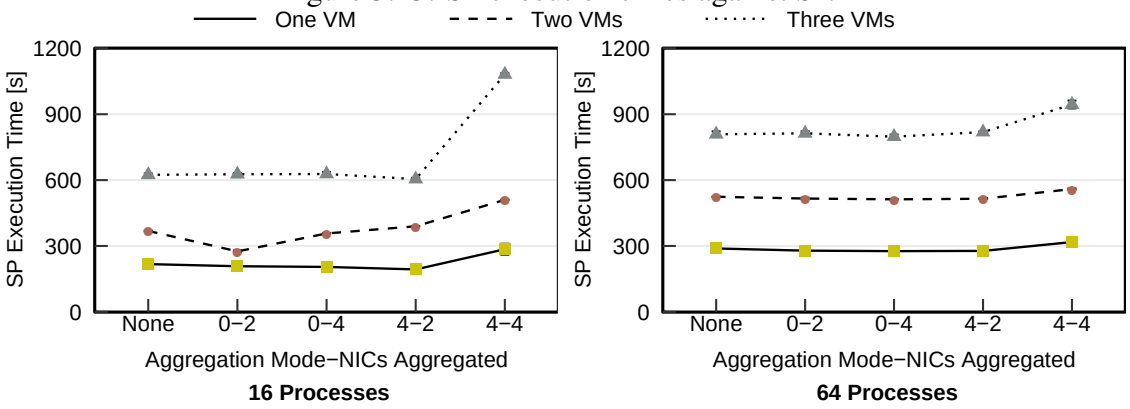
Mode four with 2 NICs outperforms baseline with one VM (in ≈12%) and four VMs (in ≈3%). On the other hand, with two VMs, performance decreased in ≈5%. Using the same mode but with four VMs in comparison with mode 4 and two NICs, performance has significantly decreased in all three environments (in ≈32% with a single VM, ≈23% with two VMs, and ≈44% with four VMs).

In the same combination of applications, but with 64 processes (right graph), RR with 2 NICs aggregated has outperformed baseline with a single VM (in ≈3%), and with two VMs VM (in ≈1%). On the other hand, with four VMs, performance has slightly decreased. Using the same mode with four NICs, turns the performance to slightly increase

in all three environments compared to mode 0 and 2 NICs.

In the execution with mode four and two NICs, performance has overcome baseline with one VM in ≈4%, and with two VMs in ≈1%. On the contrary, with four VMs, it has decreased in ≈1%. Finally, using mode 4 with 4 NICs aggregated has turned the performance to decrease in all three environments compared to the same mode with two NICs (in ≈12% with no parallel VM, ≈7% with two VMs, and ≈13% with four VMs). In this combination, the performance obtained through the usage of NICs aggregation has overcome baseline in ≈54% of the cases. Moreover, RR mode outperforms 802.3ad mode in ≈66% of the cases (comparisons made in the same environments and number of NICs).

Figure 5.15: SP execution times against SP.



In the last combination of this group, SP was executed in the front VM, and BT was executed in parallel VMs. This combination is seen in Figure 5.16. In the execution with 16 processes, using RR mode with two NICs, performance has increased in all three environments (≈4% with a single VM, ≈33% with two VMs, and ≈4% with four VMs) in comparison to the baseline. Increasing the number of aggregated NICs to four and using the same mode increases the performance with one VM (in ≈1%) and slightly with four VMs. On the other hand, with two VMs, the performance has decreased in ≈22%, compared with the same mode and two NICs.

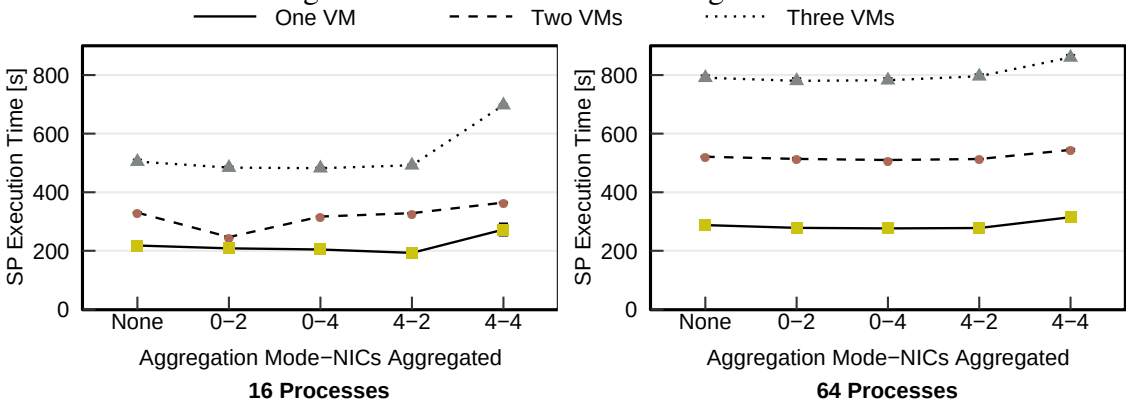
In the execution of mode 4 and 2 NICs, performance has outperformed baseline in all three environments (in ≈12% with a single VM, ≈0.5% with two VMs, and ≈2% with four VMs). However, using the same mode but with four NICs aggregated turns the performance to significantly decrease in all environments (≈29% with a single VM, ≈9% with two VMs, and ≈29% with four VMs).

In the same combination but with 64 processes, using the RR mode with 2 NICs aggregated turns the performance to overcome baseline in all three environments (≈3% with one VM, ≈1% with two VMs, and ≈1% with four VMs). Increasing the number

of NICs to four makes the performance slightly increase with one VM and two VMs, comparing it to the same mode but with two NICs. On the other hand, with four VMs, performance has slightly decreased.

Mode four with two NICs outperforms baseline with one (in ≈3%) and two VMs (in ≈1%). However, with four VMs, performance slightly decrease. Using the same mode with four NICs turn the performance to decrease in all environments (≈11% with a single VM, ≈5% with two VMs, and ≈7% with four VMs). When we used an aggregation mode for this combination, performance has outperformed baseline in ≈71% of the cases. Finally, RR overcomes 802.3ad in ≈75% of the cases comparing both aggregation modes.

Figure 5.16: SP execution times against BT.



5.3.4 Important Findings

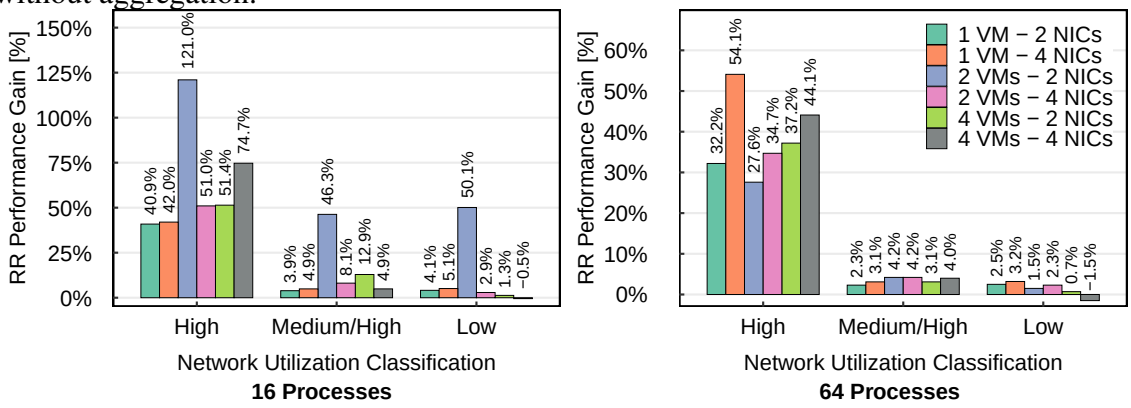
Here we summarize all the results using a global view of them and point out our findings of the aggregation experiments with the private cloud. Below are the graphs with the comparison results. In Y-axis, we present the aggregation mode performance gain. On the X-axis is the name of the applications group considering its network utilization. The higher the percentage, the greater the gain of execution with aggregation. These results are an average of the four cases used by each group. For instance, in the high network utilization group, the results represent the average gain of the cases IS x IS, FT x FT, FT x IS, and IS x FT compared against the baseline.

We start analyzing RR aggregation mode against the baseline environment results with the applications combinations from the higher to the lowest network utilization executed with 16 and 64 processes. These results are depicted in Figure 5.17. As we can see, with 16 processes (left side graph), the higher gains happen with the execution of 2

VM and 2 NICs aggregated, even with high, medium/high, and low network utilization, reaching up to ≈121% of gain against the baseline. These results are mainly seen in the executions of IS x IS (Figure 5.5, and FT x IS (Figure 5.8 for the high network utilization, BT x IS (Figure 5.9, and BT x FT (Figure 5.10 for the medium/high network utilization, and BT x BT (Figure 5.13, and BT x SP (Figure 5.10 for the low network utilization. Also, on average, the higher gains are in the high network utilization group. As expected, in the groups with less network utilization, the performance gains are smaller. Finally, the only average with a slight performance loss (less than ≈1%) is seen in the low network utilization group with 4 VMs, and 4 NICs aggregated.

The performance gains are more singular in the executions with 64 processes (right side graph) than with 16 processes execution. For instance, the higher gains are up to ≈54% with 64 processes and ≈121% with 16 processes. In general, the results with 64 processes obtained lower gains than the results with 16 processes because of the concurrency, in which there is a bigger dispute of CPU. Again, the higher gains are in the high network utilization group, mostly with 1 VM and 4 NICs aggregated. This result is seen in the executions of FT x FT (Figure 5.7) and FT x IS (Figure 5.8). Also, we can see that the executions with 4 aggregated NICs usually have better results than with 2 aggregated NICs. This previous affirmative is true in the high and medium/high network utilization groups in all execution and the low with one and two VMs. Compared to the baseline, the only results of loss performance are in the low group with 4 VMs and 4 NICs. With both executions using 16 and 64 processes, we can see that the Round Robin aggregation mode can provide better results in most of the results.

Figure 5.17: Results from the comparison between RR aggregation mode and baseline without aggregation.



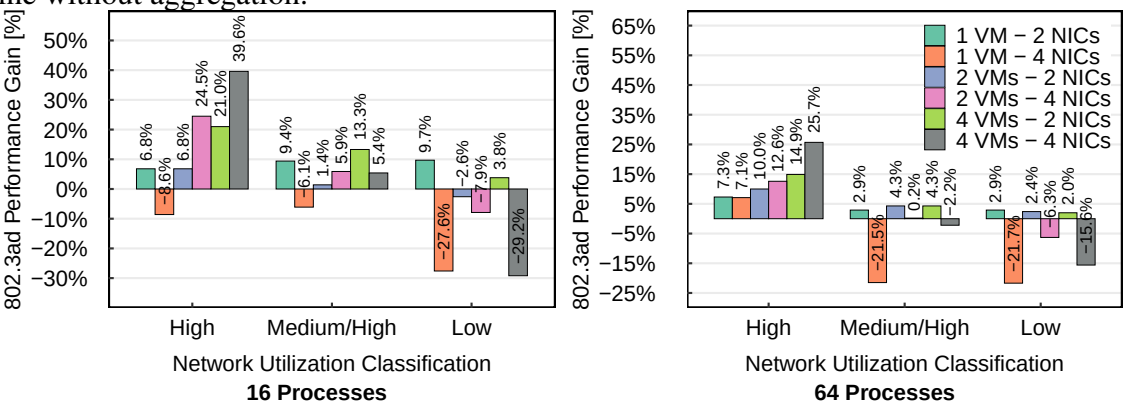
In Figure 5.18, we depict the comparison of 802.3ad aggregation mode against the baseline environment with the applications combinations from the higher to the low-

est network utilization executed with 16 and 64 processes. At a first look, we can see that different from the RR mode, 802.3ad aggregation mode can significantly gain and lose performance in specific results compared to the baseline. In the high network utilization group with 16 processes, five from six aggregation configurations performed better than baseline. The only aggregation that loss performance is with one VM and 4 NICs aggregated (≈-8.6% compared to baseline). With medium/high network utilization, five from six configurations performed better than baseline, with the only loss of performance in the executions with one VM and 4 NICs aggregated (≈-6.1% compared to the baseline). Only 2 from 6 aggregation configurations have outperformed the baseline in the last group with low network utilization. The outstanding results in this graph are with one VM and 4 NICs aggregated (≈-27.6%, results also seen in BT x BT (Figure 5.13), and BT x SP (Figure 5.14)), and with 4 VMs and 4 NICs aggregated (≈-29.2%, results also seen in BT x SP (Figure 5.14), and SP x SP (Figure 5.15)) which significantly loss performance.

With 64 processes in the high network utilization group, all configurations performed better than baseline. It is also noticeable that the performance gain is higher as the number of VMs increases and the number of NICs aggregated. In the Medium/High network utilization group, two from six configurations are outperformed by baseline, mostly with 1 VM and 4 NICs aggregated (≈-21.5%, results seen in BT x IS (Figure 5.9), and BT x FT (Figure 5.10)). Finally, in the last group, with low network utilization, three from six configurations performed worse than baseline, mostly seen with 1 VM and 4 NICs (≈-21.7%, results seen in the executions of BT x BT (Figure 5.13), and BT x SP (Figure 5.14), and with 4 VMs and 4 NICs (≈-15.6%, results seen in BT x SP (Figure 5.14), and SP x SP (Figure 5.15)). What calls attention is that these three results happen in the aggregation with four NICs. As shown by the results of Figure 5.18, 802.3ad aggregation mode can provide better performance with applications that demand more network utilization. On the other hand, for other applications, the results may not be as expected.

Finally, in Figure 5.19, we depict the comparison between the aggregation modes Round Robin and 802.3ad. In Y-axis, we have the RR performance gains, which means that in all positive percentages, RR outperformed 802.3ad. In the executions with 16 processes (left side graph), all configurations from the high network utilization group, RR obtained better results than 802.3ad, with outstanding results with 2 VMs and 2 NICs aggregated (≈106.9%, results seen in IS x FT (Figure 5.6), and FT x IS (Figure 5.8)). In the Medium/High network utilization group, RR outperformed 802.3ad in three of six configurations. As can be seen, performance losses are generally up to ≈4%, but the gains

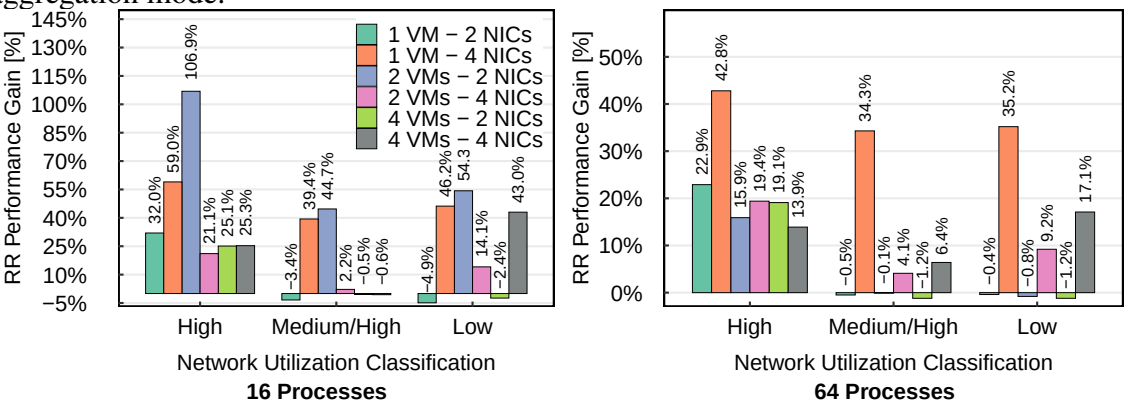
Figure 5.18: Results from the comparison between 802.3ad aggregation mode and baseline without aggregation.



can reach more than ≈44%. Finally, in the low network utilization group, 802.3ad only outperformed RR in two of the six configurations.

With 64 processes executions, the outstanding results are the configuration with one VM and 4 NICs aggregated, which outperformed 802.3ad mode in all network utilization groups in more than ≈30% (Results seen in FT x FT (Figure 5.7), and FT x IS (Figure 5.8) for the high group, BT x IS (Figure 5.9), and BT x FT (Figure 5.10) for the medium/high group, and BT x BT (Figure 5.13), and BT x SP (Figure 5.14) for the low group). In the high network utilization group, again, all configurations have better results than 802.3ad mode. In the Medium/High network utilization group, three from six configurations outperformed RR mode in up to ≈1.2%. On the other hand, RR outperformed 802.3ad by up to ≈34%. In the low network utilization group, three from six configurations outperformed 802.3ad mode in up to ≈35%. As can be seen in both graphs, in the majority of the executions, the performance has obtained better results using RR aggregation mode than 802.3ad aggregation mode.

Figure 5.19: Results from the comparison between RR aggregation mode and 802.3ad aggregation mode.



6 CONCLUSION AND FUTURE WORK

This thesis presented an analysis of the performance and cost of using high-performance interconnections in a public cloud and NIC aggregation configurations in a private cloud. With both evaluations, the general conclusion and main contribution are that network interconnection is crucial and can severely impact the performance of HPC applications executed in the cloud.

For applications with a high level of data dependency, the network reaches or even surpasses the same level of importance as computing power. Several efforts have already been made in the development of new network technologies or specific approaches for HPC. Usually, such techniques are not widely accessible by the entire community because of their cost and complexity. To overcome this limitation, the cloud offers a simplified environment with auto-configured instances. It is recommended to perform a profile or characterization of the HPC applications to improve the performance and cost efficiency in clouds. Thus, it is possible to determine the requirements concerning the network and computational power and to be able to allocate the cloud environment correctly.

In Azure public cloud provider, we demonstrated that the interconnection plays a crucial role in speeding up MPI applications. For example, the FT application performs ≈373% better with 50 GbE InfiniBand than the 10 Gigabit Ethernet interconnection, which leads to a cheaper execution cost. On the other hand, a faster interconnection may not impact their performance for CPU-bound applications, like EP. The Accelerated Networking approach of Microsoft Azure proved to be very effective. It improves the performance of MPI applications and reduces the cost as Azure does not include an additional charge when the approach is used. Recapturing the previous research question "*Considering that a faster interconnection theoretically results in improved application performance, can it also lead to a higher cost efficient?*" With the obtained results, we can infer that in the case of Microsoft Azure, using the Accelerated Networking approach, the instances with a high-performance interconnect can be used to speed up the performance of HPC applications and higher better cost efficiency.

In our private cloud evaluation, our first question was "*Can the NIC aggregation approach improve HPC applications' performance on the cloud?*" Answering that with the obtained results, we argue that the NIC aggregation approach integrated into the cloud improved the applications' performance concerning the high network usage scenario in most executions. For instance, RR and 802.3ad modes performed better than baseline in

≈98% of the executions. In the medium/high network usage scenario, RR and 802.3ad modes outperformed the baseline in ≈86% of the executions. Finally, RR and 802.3ad modes outperformed the baseline in ≈64% of the executions in the low network usage scenario. Secondly, "*Which is the number of NICs aggregated and aggregation mode that provides better performance?*" RR performed better than 802.3ad in the majority of the executions. As expected, the NIC aggregation technique tends to have better results when we execute network-intensive applications.

For the future, based on the results of this work, we plan to carry out the same assessment with public clouds on other leading public cloud providers such as Amazon AWS and Google Cloud, comparing not just one provider against itself but between providers. Also, in private clouds, we plan to expand the analysis using other virtualization technologies like KVM and assess this environment with a wide range of real applications, considering more complicated scenarios (i.e., real-time environments).

6.1 Limitations

This work has some limitations in techniques used as well as in methodologies applied in performance evaluation. For instance, although NIC aggregating can improve performance, all the configuration is done manually. The maximum number of aggregated physical links is limited to eight, and all network interfaces must operate at the same speed to be aggregated. Besides, the IEEE 802.3ad mode also imposes its request, which requires a switch with support to use this aggregation mode. Concerning the methodologies used, this work was limited by utilizing a single public cloud provider (Azure) and a group of instances sizes. Finally, we also have limitations in the hardware used to create our private cloud evaluation, which is not up to date.

6.2 Publications

The following papers (listed in reverse chronological order) were published since entering the master program and contain material that is relevant to this dissertation:

- Maliszewski, Anderson M; Vogel, Adriano; Griebler, Dalvan; Schepke, Claudio; Navaux, Philippe O A. **Ambiente de Nuvem Computacional Privada para Teste e Desenvolvimento de Programas Paralelos**. In: Minicursos da XXI Escola

Regional de Alto Desempenho da Região Sul (Minicursos ERAD). Santa Maria, Brazil, 2021.

- Maliszewski, Anderson M; Roloff, Eduardo; Carreño, Emmanuell D; Griebler, Dalvan; Gaspary, Luciano P; Navaux, Philippe O A. **Performance and Cost-Aware in Clouds: A Network Interconnection Assessment**. In: IEEE Symposium on Computers and Communications (ISCC). Rennes, France, 2020. (Qualis A2).
- Maliszewski, Anderson M; Roloff, Eduardo; Griebler, Dalvan; Gaspary, Luciano P; Navaux, Philippe O A. **Performance Impact of IEEE 802.3ad in Container-based Clouds for HPC Applications**. In: International Conference on Computational Science and its Applications (ICCSA). Cagliari, Italy, 2020. (Qualis A3).
- Maliszewski, Anderson M; Roloff, Eduardo; Griebler, Dalvan; Navaux, Philippe O A. **Avaliando o Impacto da Rede no Desempenho e Custo de Execução de Aplicações HPC** In: 20th Escola Regional de Alto Desempenho da Região Sul (ERAD-RS). Santa Maria, RS, Brazil, 2020.
- Maliszewski, Anderson; Roloff, Eduardo; Griebler, Dalvan; Navaux, Philippe. **O Impacto da Interconexão de Rede no Desempenho de Programas Paralelos**. In: XX Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD). Campo Grande, Brazil, 2019. (Qualis B3).
- Maliszewski, Anderson M; Vogel, Adriano; Griebler, Dalvan; Roloff, Eduardo; Fernandes, Luz G; Navaux, Philippe O A. **Minimizing Communication Overheads in Container-based Clouds for HPC Applications** In: IEEE Symposium on Computers and Communications (ISCC). Barcelona, Spain, 2019. (Qualis A2).

REFERENCES

BADGER, M. L. et al. **Cloud Computing Synopsis and Recommendations**. [S.l.]: National Institute of Standards & Technology, 2012.

BAILEY, D. et al. The NAS Parallel Benchmarks; Summary and Preliminary Results. In: **ACM/IEEE Conference on Supercomputing (SC)**. [S.l.: s.n.], 1991.

BHOWMIK, S. **Cloud Computing**. [S.l.]: Cambridge University Press, 2017. ISBN 9781316638101.

BUYYA, R. et al. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. **Future Generation Computer Systems**, 2009.

CHAKTHRANONT, N. et al. Exploring the Performance Impact of Virtualization on an HPC Cloud. In: **International Conference on Cloud Computing Technology and Science (CloudCom)**. [S.l.: s.n.], 2014.

CHAUFOURNIER, L. et al. Performance evaluation of multi-path tcp for data center and cloud workloads. In: **ACM/SPEC International Conference on Performance Engineering**. [S.l.: s.n.], 2019.

DAVIS, T. et al. **Linux Ethernet Bonding**. 2011. Disponível em: <https://www.kernel.org/doc/Documentation/networking/bonding.txt>.

EMERAS, J. et al. Amazon Elastic Compute Cloud (EC2) versus In-House HPC Platform: A Cost Analysis. **IEEE Transactions on Cloud Computing (TCC)**, 2019.

ESCUDERO-SAHUQUILLO, J. et al. Efficient and Cost-Effective Hybrid Congestion Control for HPC Interconnection Networks. **Transactions on Parallel and Distributed Systems (TPDS)**, 2015.

EXPÓSITO, R. R. et al. Performance Analysis of HPC Applications in the Cloud. **Future Generation Computer Systems**, 2013.

GUPTA, A. et al. Evaluating and Improving the Performance and Scheduling of HPC Applications in Cloud. **IEEE Transactions on Cloud Computing (TCC)**, 2016.

IEEE. IEEE Standard for Information Technology - Local and Metropolitan Area Networks - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications-Aggregation of Multiple Link Segments. **IEEE Std 802.3ad-2000**, 2000.

INTEL. **Intel®MPI Benchmarks: User Guide and Methodology Description**. [S.l.], 2014.

JAIN, R. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. **Digital Equipment Corporation-Litleton, Massachusetts. John Wiley & Sons, Inc**, 1991.

KAMBURUGAMUVE, S. et al. Low Latency Stream Processing: Apache Heron with Infiniband/Intel Omni-Path. In: **International Conference on Utility and Cloud Computing (UCC)**. [S.l.: s.n.], 2017.

KNÜPFER, A. et al. Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir. In: **Tools for High Performance Computing 2011**. [S.l.]: Springer, 2012.

LIU, J. et al. Performance Comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics. In: **ACM/IEEE Conference on Supercomputing (SC)**. [S.l.: s.n.], 2003.

LIU, J.; WU, J.; PANDA, D. K. High Performance RDMA-based MPI Implementation over InfiniBand. **International Journal of Parallel Programming (IJPP)**, 2004.

MALISZEWSKI, A. M. et al. Performance and cost-aware hpc in clouds: A network interconnection assessment. In: **IEEE Symposium on Computers and Communications (ISCC)**. [S.l.: s.n.], 2020.

MALISZEWSKI, A. M. et al. Performance Impact of IEEE 802.3ad in Container-based Clouds for HPC Applications. In: **International Conference on Computational Science and its Applications (ICCSA)**. [S.l.: s.n.], 2020.

MALISZEWSKI, A. M. et al. Minimizing Communication Overheads in Container-based Clouds for HPC Applications. In: **IEEE Symposium on Computers and Communications (ISCC)**. [S.l.: s.n.], 2019.

MALISZEWSKI, A. M. et al. Ambiente de Nuvem Computacional Privada para Teste e Desenvolvimento de Programas Paralelos. In: **Minicursos da XXI Escola Regional de Alto Desempenho da Região Sul**. [S.l.: s.n.], 2021.

MAUCH, V.; KUNZE, M.; HILLENBRAND, M. High Performance Cloud Computing. **Future Generation Computer Systems**, 2013.

MELL, P.; GRANCE, T. et al. The NIST Definition of Cloud Computing. **National Institute of Standards and Technology (NIST)**, Gaithersburg, United States, 2011.

MOURA, J.; HUTCHISON, D. Review and Analysis of Networking Challenges in Cloud Computing. **Journal of Network and Computer Applications (JNCA)**, 2016.

RISTA, C. et al. Improving the Network Performance of a Container-Based Cloud Environment for Hadoop Systems. In: **International Conference on High Performance Computing & Simulation (HPCS)**. [S.l.: s.n.], 2017.

ROLOFF, E. et al. High Performance Computing in the Cloud: Deployment, Performance and Cost Efficiency. In: **International Conference on Cloud Computing Technology and Science Proceedings (CloudCom)**. [S.l.: s.n.], 2012.

ROLOFF, E. et al. HPC Application Performance and Cost Efficiency in the Cloud. In: **Euromicro Int. Conference on Parallel, Distributed and Network-based Processing (PDP)**. [S.l.: s.n.], 2017.

RUIVO, T. P. P. D. L. et al. Exploring Infiniband Hardware Virtualization in OpenNebula towards Efficient High-Performance Computing. In: **IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)**. [S.l.: s.n.], 2014.

SADOOGHI, I. et al. Understanding the Performance and Potential of Cloud Computing for Scientific Applications. **IEEE Transactions on Cloud Computing**, 2017.

SILVA, G. et al. **Create a Linux Virtual Machine with Accelerated Networking using Azure CLI**. 2019. Disponível em: <https://docs.microsoft.com/en-us/azure/virtual-network/create-vm-accelerated-networking-cli?WT.mc_id=docs-azuredevtips-micrum>.

STANISIC, L.; LEGRAND, A.; DANJEAN, V. An Effective Git and Org-mode Based Workflow for Reproducible Research. **ACM SIGOPS Operating Systems Review**, ACM, 2015.

VÁZQUEZ, M. et al. Alya: Multiphysics Engineering Simulation Toward Exascale. **Journal of Computational Science**, 2016.

VIENNE, J. et al. Performance Analysis and Evaluation of InfiniBand FDR and 40GigE RoCE on HPC and Cloud Computing Systems. In: **Symposium on High-Performance Interconnects (HOTI)**. [S.l.: s.n.], 2012.

VOGEL, A. et al. Private iaas clouds: A comparative analysis of opennebula, cloudstack and openstack. In: **Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)**. [S.l.: s.n.], 2016.

VOGEL, A. et al. An Intra-Cloud Networking Performance Evaluation on CloudStack Environment. In: **Euromicro Int. Conference on Parallel, Distributed and Network-based Processing (PDP)**. St. Petersburg, Russia: [s.n.], 2017.

WANG, C. et al. Coupling GPU and MPTCP to Improve Hadoop/MapReduce Performance. In: **International Conference on Intelligent Green Building and Smart Grid (IGBSG)**. [S.l.: s.n.], 2016.

WATANABE, T. et al. Impact of Topology and Link Aggregation on a PC cluster with Ethernet. In: **IEEE International Conference on Cluster Computing**. [S.l.: s.n.], 2008.

ZAHID, F. **Network Optimization for High Performance Cloud Computing**. Tese (Doutorado) — Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway, 2017.

ZHANG, J. et al. MVAPICH2 over OpenStack with SR-IOV: An Efficient Approach to Build HPC Clouds. In: **2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing**. [S.l.: s.n.], 2015.

APPENDIX A — RESUMO EM PORTUGUÊS

This chapter presents a summary of this master thesis in the Portuguese language, as required by the PPGC Graduate Program in Computing.

Este capítulo apresenta um resumo desta dissertação de mestrado em língua portuguesa, conforme exigido pelo Programa de Pós-Graduação em Computação.

A.1 Introdução

Com o aumento da complexidade e número de problemas computacionais, bem como do valor de aquisição de infraestruturas privadas, há uma migração significativa de ambientes tradicionais para aqueles que fornecem recursos de uma forma rápida, escalável e pagável pelo uso, tais como a computação em nuvem (BHOWMIK, 2017).

Os ambientes de nuvens foram desenvolvidos através de várias tecnologias (por exemplo, virtualização), e características de computação distribuída, em grade, e paralela, disponíveis praticamente desde 2010. Atualmente, é um modelo consolidado capaz de fornecer recursos computacionais sob demanda (por exemplo, CPU, GPU, memória, armazenamento, rede) sem investimentos iniciais através de três camadas de serviço, conhecidas como IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*), e SaaS (*Software as a Service*) (MELL; GRANCE et al., 2011).

Conforme as previsões da Gartner[1], a migração de ambientes de computação habituais para nuvens públicas, que já era considerada significativa antes da pandemia, tende a aumentar em cerca de 18% em 2021, gastando 304 mil milhões de dólares. Esta estatística considera principalmente a completa “validação” do ambiente de nuvens, visto que durante a crise da COVID-19 vários trabalhos se tornaram remotos ou precisaram mesmo de maior flexibilidade de recursos por parte das empresas.

A computação de alto desempenho (HPC), que é fornecida por clusters, grades e modelos de computação em nuvem "como um serviço", tem sido historicamente utilizada para acelerar o processamento de dados. O potencial da Cloud Computing (CC) aumentou devido às melhorias na pilha de tecnologia, e pode fornecer uma alternativa aos métodos de computação habituais, tanto na escalabilidade de recursos como na redução de custos.

Tendo em vista estes benefícios, temos testemunhado esforços para executar apli-

[1] <https://www.gartner.com/en/newsroom/press-releases/2020-11-17-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-18-percent-in-2021>

cações de HPC em nuvens. Estes benefícios foram tipicamente obtidos ao preço de perdas de desempenho devido ao impacto negativo da camada de virtualização (em comparação com o ambiente nativo) e à sobrecarga da partilha/competição por recursos (por exemplo, interconexão de redes) (EMERAS et al., 2019; GUPTA et al., 2016).

Além disso, como as aplicações HPC executadas em clusters ou mesmo em ambientes de computação em nuvem são geralmente desenvolvidas utilizando Interface de Passagem de Mensagens (MPI), e as características de comunicação das aplicações variam devido à sua finalidade específica, a interconexão de redes pode ter impacto no desempenho global. Por conseguinte, o ambiente de computação deve assegurar uma comunicação de alto desempenho: elevado rendimento e baixa latência para responder aos requisitos da aplicação e não se tornar o gargalo de todo o sistema. No entanto, tal como estudos de investigação anteriores mostraram (ROLOFF et al., 2017; MOURA; HUTCHISON, 2016; SADOOGHI et al., 2017), a interconexão de redes é ainda um desafio considerável para aplicações paralelas executadas em nuvens.

A.1.1 Contribuições

A computação em nuvem oferece vários benefícios através dos seus modelos "como um Serviço", como o pagamento pelo uso, elasticidade, e acesso instantâneo a um conjunto de recursos computacionais. Uma migração significativa de ambientes inteiros é comumente vista nos últimos anos. Esta migração inclui também a execução de aplicações HPC na nuvem, procurando os benefícios acima referidos.

Em aplicações que exigem uma quantidade considerável de recursos computacionais, o desempenho tem de ser assegurado. No entanto, devido às características da nuvem, dois problemas principais bem conhecidos podem limitar o desempenho global.

O primeiro problema refere-se à virtualização, utilizada como base para a CC, que pode induzir perdas de desempenho em comparação com o ambiente nativo. Com esforços conjuntos da academia e indústria para reduzir tais perdas, foram criados contêineres, que utilizam a virtualização leve, conhecida como virtualização ao nível do SO.

O segundo problema diz respeito a uma premissa de HPC, que procura extrair o máximo de recursos possível. Esta condição só é alcançada com a garantia de exclusividade de recursos. Embora esta atribuição prioritária possa ser garantida pela nuvem em memória, processador e armazenamento, não é garantida em relação à rede, visto que o equipamento de comutação é inevitavelmente partilhado entre vários servidores.

Além disso, observa-se que em ambientes de nuvem a infraestrutura de rede subjacente é transparente para o usuário que aloca recursos. Assim, o processamento de vários fluxos de rede provenientes de diferentes instâncias pode causar flutuações na latência e no rendimento, tornando a interligação da rede um dos principais pontos de estrangulamento da computação em nuvem (ROLOFF et al., 2017; SADOOGHI et al., 2017).

Devido à falta de estudos abrangentes que estimem o impacto da interligação da rede nas nuvens, esta situação foi identificada como uma oportunidade de investigação. Para atingir o nosso objetivo, realizamos três avaliações: Selecionamos aplicações HPC representativas e criamos seu respectivo perfil. Comparamos a utilização de alto desempenho com as interconexões de rede tradicionais em relação ao desempenho e à relação custo-eficácia numa nuvem pública. Utilizamos a configuração de agregação de NIC integrada com uma nuvem privada e comparamos vários cenários com usuários simultâneos (*multi-tenants*) executando aplicações de HPC. Com ambos os resultados da avaliação do desempenho, podemos observar o impacto da rede nas nuvens privadas/públicas.

A.2 Avaliações

Nas seções a seguir, definiremos o porquê e como nossas avaliações foram feitas. Além disso, apresentaremos em breve a metodologia de avaliação.

A.2.1 Criação de perfis de aplicações HPC

Computação de alto desempenho é um termo usado para descrever a capacidade de processar dados e realizar cálculos complexos em altas velocidades. Suas soluções são vistas principalmente em infraestruturas computacionais gigantes conhecidas como supercomputadores. Essas infraestruturas são compostas por milhares de nós de computação criando um cluster, interconectado por tecnologias de rede, trabalhando juntos para completar uma ou mais tarefas.

A idealização de supercomputadores foi criada a partir da evolução da programação de aplicações e do projeto de processadores de computadores. Em primeiro lugar, todos os processadores foram criados com um único núcleo, portanto, as aplicações foram executados em série. Esse conceito mudou com a introdução de processadores multi e many-core, que permitiram que aplicações fossem executadas usando o paradigma de

computação paralela. Com o surgimento do paralelismo computacional, problemas complexos que exigem grandes quantidades de cálculo podem ser resolvidos decompondo o problema em tarefas menores e executando as instruções de cada tarefa em paralelo usando vários processos e/ou núcleos computacionais.

Com o advento do sistema multiprocessador, diferentes modelos de programação paralela tornaram possível melhorar o desempenho do sistema. Existem várias classes de hardware paralelo e, muitos modelos de programação paralela diferentes. Entre eles, podemos destacar os modelos de memória compartilhada e memória distribuída.

O modelo de memória compartilhada é usado em ambientes com vários processadores que compartilham o espaço de endereço de uma única memória; ou seja, os processadores podem operar independentemente, mas compartilhar os recursos de memória. A técnica mais comum para criar programas paralelos usando este modelo é o OpenMP.

Por outro lado, na memória distribuída, o ambiente possui diversos processadores, cada um com sua memória e interligados por uma rede de comunicação. As tarefas compartilham dados através da comunicação de envio e recebimento de mensagens, e assim, múltiplas tarefas são iniciadas e distribuídas pelos processadores do ambiente, utilizando seu endereço de memória. A técnica mais comum para usar este modelo é o MPI, usada em ambientes distribuídos com geralmente mais de um nó (clusters, grades, nuvens).

Esta dissertação usou aplicações paralelas representativas usando a implementação *Message Passing Interface* (MPI) do conjunto NAS Parallel Benchmarks. Esses benchmarks foram escolhidos devido a sua alta utilização na academia e para cobrir uma gama maior de padrões paralelos. Assim, para explorar os padrões paralelos e corroborar nossa avaliação de desempenho nas nuvens, iniciamos o perfil das aplicações. Com os resultados da criação de perfil, esperamos obter as características das aplicações e determinar se uma aplicação específica é orientado por comunicação ou computação.

A.2.2 Nuvens Públicas

As nuvens públicas são geralmente criadas em grandes infraestruturas computacionais (também conhecidas como clusters ou supercomputadores), que um provedor de nuvem possui. Ele fornece recursos de computação sob demanda para seus clientes, que pagam por seu uso (premissa de pagamento por uso) conforme a utilização. O uso de nuvens públicas remove a idealização de manter e dar suporte à infraestrutura de hardware, conforme o provedor faz isso. Assim, o usuário pode se concentrar em sua utilização.

No entanto, esta disposição transparente também tem suas desvantagens. Por exemplo, o usuário não conhece a topologia de rede interna da infraestrutura e pode alocar hardware de diferentes pods para criar um cluster. Além disso, o usuário está limitado ao uso das opções fornecidas pelo provedor de nuvem e não pode gerenciar a configuração de baixo nível dos servidores. Ainda, a infraestrutura de rede não é alocada por um único usuário, portanto, os fluxos de várias instâncias simultâneas tendem a compartilhá-los.

Avaliamos como a interconexão de rede afeta o desempenho e a eficiência de custos em uma nuvem pública, tendo em vista essas desvantagens. Implantamos três clusters individuais no provedor de nuvem pública do Microsoft Azure, cada um com oito instâncias e diferentes tamanhos e interconexões de rede. Por exemplo, um cluster com tamanho de instância A10 e 10GbE, outro cluster com tamanho de instância DS4_v2 e IB de 40GbE, e um cluster com tamanho de instância F8 e interconexão IB de 50GbE.

Considerando que uma interconexão mais rápida teoricamente resulta em melhor desempenho, também verificamos se ela poderia ser mais econômica. Nesses clusters, executamos aplicações paralelas representativas do conjunto NAS Parallel Benchmarks e a aplicação Alya HPC. Com os resultados obtidos, espera-se responder a seguinte questão: *Considerando que uma interconexão mais rápida teoricamente resulta em melhor desempenho da aplicação, também pode levar a um maior custo-benefício?*

A.2.3 Nuvens Privadas

Como o próprio nome sugere, nuvens privadas são infraestruturas computacionais privadas gerenciadas e pertencentes a uma identidade privada, por exemplo, uma empresa ou um laboratório de pesquisa. Ao contrário das nuvens públicas, os ambientes de nuvem privada não são totalmente aderentes às características essenciais da computação em nuvem definidas pelo NIST. Isso acontece porque o número de recursos, a elasticidade e o modelo de pagamento por uso são inconsistentes. Além disso, este é o modelo de nuvem com um preço mais alto, pois a organização precisa manter e comprar a infraestrutura de computação. Por outro lado, também fornece mais segurança para a organização e gerenciamento de baixo nível superior para realizar atualizações ou alterações específicas.

Neste trabalho, implantamos uma nuvem privada usando o gerenciador de nuvem OpenNebula. Em seguida, exploramos as configurações de baixo nível da interconexão de rede. Usamos uma nuvem privada porque este era o único modelo de nuvem acessível, permitindo-nos realizar configurações de baixo nível. Criamos clusters com quatro hosts

idênticos usando contêineres LXD e os integramos com a técnica de agregação de NICs. Desenvolvemos uma metodologia de avaliação considerando diferentes números de NICS agregados (até quatro), diferentes modos de agregação (802.3ad e Balanced-RR) e até três instâncias LXD simultâneas executando aplicações específicos para criar ruído na rede em relação à quarta instância LXD principal. Dessa forma, experimentamos um ambiente real e avaliamos a interferência que ocorre nele.

Com essa avaliação, pretendemos responder às seguintes perguntas: A abordagem de agregação NIC pode melhorar o desempenho das aplicações HPC na nuvem? Qual é o número de NICs agregados e o modo de agregação que fornece melhor desempenho?

A.3 Objetivo e Contribuições

O principal objetivo da nossa pesquisa é avaliar o impacto da interconexão de rede em ambientes de computação em nuvem para aplicações de computação de alto desempenho. Para este objetivo, nossas principais contribuições são as seguintes:

- Avaliação de desempenho e custo para ambiente de nuvem pública em instâncias com diferentes interconexões de rede.
- Avaliação de desempenho para ambiente de nuvem privada em diferentes configurações de rede com base em metodologias de agregação NIC.

A.4 Trabalhos Relacionados

Com a análise dos trabalhos relacionados, podemos concluir que existem várias técnicas para avaliar e criar soluções relacionadas com a melhoria do desempenho quanto às interligações de rede. Porém, a maioria normalmente envolve a atualização dos equipamentos de comunicação, exigindo uma quantidade significativa de recursos financeiros.

Devido à falta de estudos abrangentes que estimam o impacto da interconexão de redes em nuvens, essa situação foi identificada como uma oportunidade de pesquisa. Nosso trabalho vai além e busca uma comparação entre as interconexões HPC em desempenho e custo, e também avalia uma técnica de agregação NIC para melhorar o desempenho de forma econômica.

A.5 Computação em Nuvem

A computação em nuvem é hoje um modelo consolidado para fornecer recursos de computação sob demanda. Ele foi desenvolvido combinando várias características de computação distribuída, em grade e paralela e também consolidou tecnologias como virtualização, que abstrai dinamicamente os recursos de hardware (BUYYA et al., 2009).

Uma definição de computação em nuvem bem adotada pela comunidade e usada como base neste documento é fornecida pelo Instituto Nacional de Padrões e Tecnologia (NIST) como "um modelo que permite o acesso a um pool compartilhado de recursos de computação (por exemplo, redes, servidores, armazenamento e serviços) sob demanda através da rede, que podem ser rapidamente provisionados e liberados com esforços mínimos ou interações pelo provedor de serviços" (MELL; GRANCE et al., 2011). Além disso, inclui cinco características essenciais, três modelos de serviço e quatro modelos de implantação, que serão explicados a seguir.

A.5.1 Características essenciais

Conforme o NIST (MELL; GRANCE et al., 2011), a computação em nuvem tem cinco características essenciais.

- **Utilização sob demanda:** Esta primeira característica diz respeito à alocação e desalocação de recursos computacionais sem interação com a equipe do provedor.
- **Amplo acesso à rede:** Todos os serviços sob demanda oferecidos pelos provedores devem ser acessíveis em uma rede através de mecanismos padrão. Por exemplo, em uma rede local (LAN) ou na própria Internet.
- **Pooling de recursos:** Os provedores são responsáveis por possuir e gerenciar os recursos físicos e virtuais de computação e fornecê-los a vários usuários (*multi-tenant*) de acordo com sua demanda. Esses usuários não têm controle ou conhecimento da localização dos recursos, mas podem ser capazes de especificar a localização em um nível superior (por exemplo, país, estado ou datacenter).
- **Elasticidade rápida:** Os provedores de nuvem oferecem os recursos do usuário em qualquer quantidade e a qualquer momento. Do ponto de vista do cliente, os recursos parecem ilimitados.
- **Serviço medido:** Os sistemas em nuvem precisam fornecer transparência tanto para

o provedor quanto para os usuários. É permitido ao provedor usar técnicas para medir o uso e disponibilidade de recursos, como monitoramento, que também será usado para garantir o acordo de nível de serviço (SLA) e para fins de faturamento.

A.6 Agregação de NICs

Também conhecido como Agregação de Links (LA) ou NICs e Bonding, é uma técnica que combina vários NICs em um link lógico. É comumente usado para interconectar pares de dispositivos de rede (ou seja, switches, roteadores.) Para melhorar a largura de banda e a resiliência de maneira econômica, adicionando novos links e os existentes em vez de substituir o equipamento (IEEE, 2000; DAVIS et al., 2011). O comportamento específico das interfaces conectadas é baseado na escolha de um modo de uso entre sete modos existentes. Esses modos são configurados diretamente no arquivo de configuração da interface de rede do Linux sendo expressos por nome e número. Outro uso igualmente importante da agregação NIC é fazer o failover de forma transparente. Isso é preferido para implantações onde a alta disponibilidade é crítica. A mesma ideia pode ser estendida para fornecer uma combinação de largura de banda aumentada e failover transparente com desempenho degradado em um evento de falha de NIC.

A.6.1 Modos de agregação

Os modos de agregação são responsáveis por especificar quais políticas serão usadas durante a agregação NIC. Por padrão, o modo 0 ou balance-rr é usado. Os sete modos existentes são listados e descritos abaixo.

- **Balance-rr ou 0:** implementa uma política Round-robin, transmitindo todos os pacotes em sequência do primeiro ao último nó, proporcionando balanceamento de carga e tolerância a falhas.
- **Backup ativo ou 1:** implementa uma política de backup ativo, onde apenas um escravo (placa de rede) permanece ativo. A única possibilidade de outro escravo se tornar ativo é em caso de falha. Neste modo, o endereço MAC é visível em apenas uma porta do adaptador de rede para evitar confusão entre os escravos.
- **Balance-xor ou 2:** implementa uma política XOR, na qual seleciona uma interface para transmissão de pacotes baseada no resultado de uma operação XOR na

contagem das interfaces de rede escravas do MAC de origem e destino módulo de endereço. Este cálculo garante que a mesma interface seja selecionada para cada endereço MAC de destino usado.

- **Broadcast ou 3:** implementa uma política de Broadcast, permitindo que o tráfego de dados ocorra em todas as interfaces escravas.
- **802.3ad ou 4:** implementa o protocolo de agregação de link dinâmico IEEE 802.3ad, criando grupos de agregação com as mesmas velocidades e configurações duplex. Ele usa todos os escravos no agregador ativo, conforme a especificação 802.3ad. Este modo possui dois pré-requisitos, sendo: Suporte Ethtool nos drivers, além de um switch que suporte agregação de link IEEE 802.3ad.
- **Balance-tlb ou 5:** implementa uma transmissão adaptativa de balanceamento de carga, onde faz a conexão entre canais que não requerem nenhum suporte especial no switch. O tráfego de saída é distribuído conforme a carga atual da rede em cada escravo, considerando a velocidade do tráfego. O escravo atual recebe o tráfego de entrada. Se o escravo receptor falhar, outro escravo assumirá o endereço MAC do escravo com falha. Este modo requer que o sistema tenha Ethtool.
- **Balance-alb ou 6:** implementa balanceamento de carga adaptável, incluindo o balance-tlb, e recebe modos de balanceamento de carga para tráfego IPv4, não requerendo suporte especial no switch. O recebimento do balanceamento de carga é obtido por meio da negociação ARP.

A.7 Metodologia de Avaliação

Este capítulo está estruturado da seguinte forma. Na seção A.8 detalhamos a metodologia usada em nossa primeira avaliação, que cria o perfil das aplicações HPC do NAS Parallel Benchmark em relação às frações de tempo que cada processo MPI gasta em Computação e Comunicação MPI. Na seção A.9 apresentamos a metodologia de avaliação em nuvem pública Azure quanto ao desempenho e eficiência de custo da aplicação. Finalmente, a metodologia de avaliação da nuvem privada relativa às configurações de agregação NIC com vários cenários diferentes é apresentada na Seção A.10.

A.8 Perfil das aplicações

Para determinar o comportamento de execução das aplicações, realizamos um procedimento de rastreamento, que nos permite criar um perfil de execução de cada aplicação. Abaixo, descrevemos como esse processo foi feito.

Esta avaliação foi feita usando a implementação paralela *Message Passing Interface* (MPI) da suite *Numerical Aerodynamic Simulation Parallel Benchmarks* (NPB) (BAILEY et al., 1991). Realizamos um perfil computando as frações de tempo que cada processo MPI gasta na Computação e Comunicação MPI das aplicações. Com esses dados, classificamos as aplicações em quatro grupos, começando com o altamente dependente da rede (utilização intensiva da rede) até o grupo sem dependência da rede.

A.8.1 Infraestrutura Computacional/Configuração de Experimentos

Nossos experimentos foram realizados com quatro nós idênticos, cada um composto com dois Intel ®Xeon ®E5-2650 v3 (Q3'14) Haswell 2,3 GHz, 20 núcleos (10 por CPU) com Hyper-Threading habilitado resultando em 40 threads e 128 GB de RAM DDR4. Cada núcleo tem caches L1 (instrução de 32 KB e dados de 32 KB) e L2 (256 KB). O cache L3 (256 MB) é compartilhado entre todos os núcleos. Os nós são interconectados através de um switch genérico de 1 Gbps. O software possui Ubuntu Server 18.04 64 bits (kernel 4.15.0-48) como sistema operacional (SO), biblioteca MPI Open MPI 2.1.1 e compilador GCC/GNU Fortran 7.4.0.

Para rastrear as aplicações e expor o comportamento MPI e Computacional, foi utilizado o Score-P (KNÜPFER et al., 2012) versão 6.0, responsável por introduzir a instrumentação do código nas aplicações durante sua compilação. Depois disso, a execução do rastreamento da aplicação foi feita normalmente. Ao final desta etapa, fizemos uma análise post-mortem do traço, primeiro convertendo os traços originais criados no Open Trace Format Versão 2 (OTF2).

Para converter os traços de formato OTF2 atuais, usamos as ferramentas Akypuera [2], mais especificamente a ferramenta `otf22paje` [3]. Depois de concluir esta conversão, obtemos um arquivo com formato de rastreamento, que no nosso caso precisa ser con-

[2] <https://github.com/schnorr/akypuera>
[3] <https://github.com/schnorr/akypuera/wiki/OTF2WithAkypuera>

vertido para um CSV com `pj_dump` tool [4]. Finalmente, o arquivo CSV contendo todas as informações de rastreamento foi analisado na linguagem estatística R. As aplicações foram executadas com o maior número de processos possíveis/suportados pelo cluster, desta forma, BT e SP utilizaram 144 processos (o número de processos deve ser raiz quadrada), e o restante das aplicações (CG, EP, FT, IS, LU e MG) usaram 128 processos (o número de processos deve ser uma potência de 2).

Empregamos uma metodologia de pesquisa reproduzível (STANISIC; LEGRAND; DANJEAN, 2015), usando R, Git, Zenodo e um caderno de laboratório. Todos os dados coletados neste trabalho estão disponíveis publicamente [5]. Os códigos-fonte e a metodologia do projeto também estão disponíveis em um repositório Git [6].

A.9 Interconexões de alto desempenho em nuvens públicas

Como um novo passo importante para a caracterização do impacto da interconexão de rede no desempenho de aplicações HPC, nesta avaliação (MALISZEWSKI et al., 2020a), estamos avaliando instâncias do provedor de nuvem pública Microsoft Azure, que possuem diferentes opções de interconexão de rede. Mais especificamente, realizamos a análise com os tamanhos de instância *A10*, *DS4_v2* e *F8* usando 10GbE, 40GbE InfiniBand e 50GbE InfiniBand, respectivamente. Nossa avaliação usa três clusters individuais, cada um com oito tamanhos de instância - A10, DS4_v2 ou F8. Executamos benchmarks representativos sintéticos do NAS Parallel Benchmarks (BAILEY et al., 1991) suite e do aplicativo Alya HPC real (VÁZQUEZ et al., 2016) com 64 processos (8 processos por instância). Seguindo uma metodologia de pesquisa reproduzível, analisamos essas aplicações em relação ao desempenho e à relação custo-benefício.

A.9.1 Infraestrutura Computacional/Configuração de Experimentos

Esta seção descreve a configuração experimental em relação às especificações de hardware/software com os detalhes de eficiência de custo usados em nossa avaliação. Comparamos os tamanhos *A10*, *DS4_v2* e *F8* porque eles têm 8 vCPUs e suportam difer-

[4] <https://github.com/schnorr/pajeng/wiki/pj_dump>
[5] <https://doi.org/10.5281/zenodo.3581280>
[6] <https://github.com/andermm/CMP223>

entes interconexões [7]; *A10* com 10 GbE, *DS4_v2* com 40GbE IB (Família Mellanox Technologies MT27500 / MT27520 [Função Virtual ConnectX-3 / ConnectX-3 Pro]) e *F8* com 50 GbE IB (Família Mellanox Technologies MT27710 [Função Virtual ConnectX-4 Lx] (rev 80)), e diferentes custos de alocação por hora; *A10* US$ 0,78 instância, *DS4_v2* US$ 1,008 instância e *F8* US$ 0,792 instância).

Nossos experimentos foram realizados com oito instâncias com os tamanhos *A10*, *DS4_v2* e *F8* do Microsoft Azure Public Cloud. A especificação do software para as instâncias têm Ubuntu Server 18.04 de 64 bits (Kernel 5.0.0-1032-azure) como SO, biblioteca MPI Open MPI 2.1.1, compilador GCC/GNU Fortran versão 7.4.0. Os tamanhos de instância *DS4_v2* e textit F8 usaram a abordagem de rede acelerada do Microsoft Azure. Ele permite a virtualização de E/S de raiz única (SR-IOV) para a instância, melhorando seu desempenho de rede. Esse caminho de alto desempenho ignora o host do caminho de dados, reduzindo a latência, o jitter e a utilização da CPU. Além disso, habilitar a Rede Acelerada não tem custo extra.

Sem rede acelerada, todo o tráfego de rede dentro e fora da VM deve atravessar o host e o switch virtual. O switch virtual fornece toda a aplicação de políticas, como grupos de segurança de rede, listas de controle de acesso, isolamento e outros serviços virtualizados de rede para o tráfego de rede. Com a rede acelerada, o tráfego de rede chega à interface de rede da máquina virtual (NIC) e é encaminhado para a VM. Todas as políticas de rede aplicadas pelo switch virtual agora são descarregadas e implementadas no hardware. Usar a política no hardware permite que o NIC encaminhe o tráfego de rede diretamente para a VM, ignorando o host e o switch virtual, enquanto mantém toda a política aplicada no host (SILVA et al., 2019).

Executamos dez aplicações que abrangem vários padrões paralelos. Eles incluem o conjunto NAS Parallel Benchmarks (IS, EP, CG, FT, MG, BT, SP e LU) como aplicações sintéticos, Alya como um aplicativo HPC real e Intel MPI Benchmarks (Ping-Pong) para obter indicadores de desempenho de rede. A aplicação **Ping-Pong** da Intel (INTEL, 2014) é executada para uma primeira visão geral da latência e throughput da rede, usando dois processos entre dois nós (um processo por nó), aumentando o tamanho das mensagens variando de 1 Byte até 4 MBytes. Tanto o conjunto de aplicações NPB quanto Alya são executados usando 8 instâncias com até 64 processos.

Empregamos uma metodologia de pesquisa reproduzível (STANISIC; LEGRAND;

[7]Embora os tamanhos da memória principal variem entre os diferentes tamanhos de instância, todos os valores foram suficientes para nossos experimentos e, portanto, não são mencionados.

DANJEAN, 2015), usando R, GIT [8], e um caderno de laboratório, disponibilizando publicamente todos os dados deste trabalhar. Seguimos um projeto de experimento fatorial completo aleatório (JAIN, 1991) para orientar a execução dos experimentos. O projeto possui 30 replicações com dois fatores (dez aplicações e três instâncias). Os tempos de execução reportados e as medidas de Ping-Pong são médias das repetições, e as barras de erro foram calculadas considerando um nível de confiança de 99,7 %, assumindo uma distribuição Gaussiana.

A métrica de custo-eficiência introduzida em Roloff et al., (ROLOFF et al., 2012) foi usada nesta avaliação para verificar qual nuvem oferece o melhor desempenho pelo preço pago. É representado pelo número de execuções de uma determinada aplicação que poderiam ser feitas em uma hora, dividido pelo custo horário da instância selecionada, neste caso, a soma das oito *A10*, *DS4_v2* ou *F8* tamanhos de instâncias.

A.10 Agregação de NICs em nuvens privadas

Nesta avaliação, usamos quatro aplicações sintéticas (BT, SP, FT, IS) do NAS Parallel Benchmarks, executados em clusters criados com containers LXD no gerenciador de nuvem OpenNebula (VOGEL et al., 2016; MALISZEWSKI et al., 2021). Nós progredimos no estado da arte avaliando várias combinações de diferentes modos de agregação NIC (802.3ad modo 4 e modo Balanced Round-Robin 0), número de NICs agregados (quatro, dois e um), também, introduzindo o impacto de instâncias paralelas (quatro, dois e linha de base).

Os recursos computacionais são divididos igualmente entre o número de instâncias. Por exemplo, com 4 instâncias LXD simultâneas implantadas, cada uma tinha 25% do total de recursos de computação disponíveis. Criamos uma metodologia (ver Tabela4.2) que divide as execuções em três ambientes quanto à utilização da rede pelas aplicações utilizados: Alta, Média/Alta e Baixa. Esta metodologia foi criada com base em pesquisas anteriores (MALISZEWSKI et al., 2019; MALISZEWSKI et al., 2020b) e em seu perfil (Seção A.8), em que consideramos as aplicações BT e SP como de médio/baixo utilização da rede e a aplicação FT e IS como tendo alta utilização da rede.

[8] https://github.com/andermm/ISCC-2020.git

A.10.1 Infraestrutura Computacional/Configuração de Experimentos

O ambiente computacional era composto por quatro servidores HP ProLiant com recursos de hardware idênticos. Cada um tem dois processadores AMD Opteron de seis núcleos 2425 HE, 32 GB de RAM, 4 placas de interface de rede Intel Gigabit (NICs) interconectadas por um switch Gigabit. A especificação do software tem Ubuntu Server 18.04 64 bits (kernel 4.15.0-99) como SO, biblioteca MPI Open MPI 2.1.1, compilador GCC/GNU Fortran versão 7.5.0. Além disso, o gerenciador de nuvem OpenNebula foi usado com a versão 5.10.1 e o driver Ethernet Channel Bonding com a versão 3.7.1. Todos os softwares envolvidos no processo de avaliação foram usados com sua última versão estável disponível. As instâncias LXD foram criadas usando o LXC versão 3.0.3 e usaram o mesmo SO, MPI wrapper e versão GCC dos servidores físicos.

Empregamos uma metodologia de pesquisa reproduzível usando R, Git e um caderno de laboratório. Para orientar a execução dos experimentos, criamos um programa para iniciá-lo automaticamente em todos os nós (principal e paralelo). Por exemplo, quando pretendemos avaliar a interferência causada por três instâncias executando o BT, contra a execução do BT na instância principal, nosso programa primeiro baixa e compila os benchmarks em todas as instâncias, então, na instância principal, ele lê o experimento projeto, que contém a ordem de execução das aplicações e cria um arquivo de saída em uma pasta NFS. Em seguida, nos nodos paralelos, ele chama um script que mata todas as aplicações em execução, chama um script para ler a saída criada pela instância principal na pasta NFS, cria um arquivo de confirmação na mesma pasta dizendo que está pronto para causar o ruído e, finalmente, selecione e execute uma aplicação em um loop infinito. Após confirmar que as instâncias paralelas estão realizando suas execuções, a aplicação começará a ser executado na instância principal. Cada vez que a execução da aplicação na instância principal termina e muda para outra, uma mensagem é enviada por meio do NFS e o loop de execução infinito nas instâncias paralelas é eliminado. Finalmente, o programa relê o projeto experimental e reinicia as etapas anteriores.

Cada vez que encerramos a execução dos experimentos para a linha de base, cada número de NICs agregados (0, 2 e 4) e diferentes modos de agregação (802.3ad e Balanced Round-Robin), precisamos reiniciar os servidores subjacentes. Com o processo de reinicialização, também nos certificamos de que não havia interferência nos experimentos relacionados aos vários níveis de cache (por exemplo, memória, instruções do processador). Os projetos possuem 10 repetições, e as medidas dos tempos de execução

relatados são médias das replicações com as barras de erro calculadas considerando um nível de confiança de 95%, assumindo uma distribuição gaussiana. Consideramos os resultados das aplicações executadas na instância principal. Os resultados das instâncias paralelas foram descartados visto que foram utilizados para causar ruído na rede e consequentemente afetar ou não o desempenho da instância principal.

Conduzimos nossa avaliação usando quatro benchmarks de HPC (IS, FT, BT e SP) do pacote Numerical Aerodynamic Simulation Parallel Benchmarks (NPB) (BAILEY et al., 1991). O conjunto NPB, usado com a versão 3.4.1, foi projetado para avaliar o desempenho de diferentes hardwares e softwares em sistemas HPC. Todos os benchmarks de NAS foram compilados com tamanho C com -O3 flag, mpifort e mpicc para códigos Fortran e C. Executamos as aplicações com duas variações no número de processos. Com 16 processos MPI (4 por instância) porque queremos avaliar apenas a simultaneidade da rede, visto que as instâncias têm seu hardware igualmente dividido (paralelo), e com 64 processos MPI (16 por instância) porque queremos adicionar mais um fator de simultaneidade, resultando em mais processos por instância do que o servidor físico possui.

A.11 Conclusão e Trabalhos Futuros

Esta tese apresentou uma análise do desempenho e custo do uso de interconexões de alto desempenho em uma nuvem pública e uma análise das configurações de agregação NIC em uma nuvem privada. Com ambas as avaliações, a conclusão geral e a principal contribuição é que a interconexão de rede é um aspecto crucial e pode impactar severamente o desempenho das aplicações HPC executados na nuvem.

Para aplicações com alto nível de dependência de dados, a rede atinge ou até ultrapassa o mesmo nível de importância do poder de computação. Vários esforços já foram feitos no desenvolvimento de novas tecnologias de rede ou abordagens específicas para HPC. Normalmente, essas técnicas não são amplamente acessíveis a toda a comunidade devido ao seu custo e complexidade. Para superar essa limitação, a nuvem oferece um ambiente simplificado com instâncias configuradas automaticamente. Recomenda-se realizar um perfil ou caracterização das aplicações HPC para melhorar o desempenho e a relação custo-benefício nas nuvens. Assim, é possível determinar os requisitos relativos à rede e capacidade computacional e conseguir alocar o ambiente de nuvem corretamente.

No provedor de nuvem pública do Azure, demonstramos que a interconexão desempenha um papel crucial na velocidade de aplicações MPI. Por exemplo, o aplicativo

FT tem desempenho aproximadamente 373% melhor com 50 GbE InfiniBand em comparação com a interconexão 10 Gigabit Ethernet, o que leva a um custo de execução mais barato. Por outro lado, para aplicações vinculados à CPU, como EP, uma interconexão mais rápida pode não afetar seu desempenho. A abordagem de Rede Acelerada do Microsoft Azure provou ser muito eficaz. Ela melhora o desempenho das aplicações MPI e reduz o custo, pois o Azure não inclui um custo adicional quando a abordagem é usada. Recapturando a questão de pesquisa anterior *Considerando que uma interconexão mais rápida teoricamente resulta em melhor desempenho do aplicativo, ela também pode levar a uma melhor relação custo-benefício?*. Com os resultados obtidos, podemos inferir que no caso do Microsoft Azure, utilizando a abordagem Accelerated Networking, as instâncias com interconexão de alto desempenho podem ser utilizadas para agilizar o desempenho das aplicações HPC e maior melhor custo-benefício.

Em nossa avaliação de nuvem privada, a primeira pergunta de pesquisa foi *A abordagem de agregação NIC pode melhorar o desempenho das aplicações HPC na nuvem?* Em resposta, com os resultados obtidos, argumentamos que a abordagem de agregação NIC integrada à nuvem melhorou o desempenho das aplicações relativo ao cenário de alto uso da rede na maioria das execuções. Por exemplo, os modos RR e 802.3ad tiveram um desempenho melhor do que a linha de base em aproximadamente 98% das execuções. No cenário de uso de rede médio/alto, os modos RR e 802.3ad superaram a linha de base em aproximadamente 86% das execuções. Finalmente, no cenário de baixo uso de rede, os modos RR e 802.3ad superaram a linha de base em aproximadamente 64% das execuções. Em segundo lugar, *Qual é o número de NICs agregados e modo de agregação que fornece melhor desempenho?* RR teve desempenho melhor do que 802.3ad na maioria das execuções. Como esperado, a técnica de agregação de NIC tende a ter melhores resultados quando executam as aplicações com uso intenso de rede.

Para o futuro, com base nos resultados deste trabalho, planejamos realizar a mesma avaliação com nuvens públicas em outros provedores de nuvem pública líderes, como Amazon AWS e Google Cloud, comparando não apenas um provedor com ele mesmo, mas entre provedores. Além disso, em nuvens privadas, planejamos expandir a análise usando outras tecnologias de virtualização como KVM e avaliar este ambiente com uma ampla gama de aplicações reais, considerando cenários mais complicados (ou seja, ambientes em tempo real).