Minimizing Communication Overheads in Container-based Clouds for HPC Applications

Anderson M. Maliszewski*†, Adriano Vogel‡, Dalvan Griebler†‡, Eduardo Roloff*,
Luiz G. Fernandes‡, Philippe O. A. Navaux*
* Informatics Institute, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
†Laboratory of Advanced Research on Cloud Computing (LARCC), Três de Maio Faculty (SETREM), Três de Maio, Brazil
‡ School of Technology, Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
Email: {andersonm.maliszewski,eroloff,navaux}@inf.ufrgs.br,
{adriano.vogel,dalvan.griebler}@acad.pucrs.br, luiz.fernandes@pucrs.br

***Abstract*—Although the industry has embraced the cloud computing model, there are still significant challenges to be addressed concerning the quality of cloud services. Network-intensive applications may not scale in the cloud due to the sharing of the network infrastructure. In the literature, performance evaluation studies are showing that the network tends to limit the scalability and performance of HPC applications. Therefore, we proposed the aggregation of Network Interface Cards (NICs) in a ready-to-use integration with the OpenNebula cloud manager using Linux containers. We perform a set of experiments using a network microbenchmark to get specific network performance metrics and NAS parallel benchmarks to analyze the performance impact on HPC applications. Our results highlight that the implementation of NIC aggregation improves network performance in terms of throughput and latency. Moreover, HPC applications have different patterns of behavior when using our approach, which depends on communication and the amount of data transferring. While network-intensive applications increased the performance up to 38%, other applications with aggregated NICs maintained the same performance or presented slightly worse performance.**

***Index Terms*—Cloud Computing; NIC Aggregation; Network Performance; Bonding; Linux Containers**

I. Introduction

Cloud Computing (CC) is a model which provides ubiquitous, convenient, on-demand network access for a shared pool of computing resources that can be provisioned on-demand and released with minimal efforts, requiring a network connection [1]. This model is divided into three layers, known as IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service) [2]. As cloud computing benefits have attracted attention from both academia and industry fields, there is a great deal of research interest for estimating the performance impact when moving applications, environments, and server infrastructures to the cloud.

The observed benefits of using cloud computing came at the price of performance unpredictability and potential overheads [3], [4]. Performance losses in clouds occur mainly due to the negative impact of virtualization layer as well as the overhead of multi-tenants sharing/competing for resources. A relevant aspect related to performance on cloud environments is the network speed achievable by cloud instances. A fast network interconnection enables rapid resources provision and can potentially improve the quality of services for instances. Nonetheless, the network performance is relevant for cloud applications, mainly for distributed processing, load balance, and high availability [5], [6].

Additionally, as previous research results [7] demonstrated, performance challenges of HPC applications executing on cloud environments are usually imposed by the network interconnection. Also, as CC is built on geographically distributed data centers and clustered servers, its performance is highly network-dependent and therefore can seriously affect the performance of data-intensive and HPC applications [8].

In this work, we tackle this problem by proposing and providing a new implementation/validation using the aggregation of multiple Network Interface Cards (NICs) working with balance round-robin mode. The goal is to improve network performance for HPC applications executing on cloud instances. For the experiments, we deployed a container-based cloud and executed a network microbenchmark to evaluate throughput and latency. We selected the NAS parallel benchmarks suite to observe the advantages of our approach.

To the best of our knowledge, there is no other paper in the literature that performs the same studies. The closest investigation is from Rista et al. [9], which performs NIC aggregation with bonding mode 4 (802.3ad) and its performance was verified to a specific application from the big data field. Our paper extends the contributions of this previous work. We instead are using a different aggregation mode (mode 0 - Balance Round-robin) and focusing on HPC applications. We can, therefore, summarize the paper contributions as follows:

- A container-based cloud deployment approach using NIC aggregation with balance round-robin mode that improves both latency and throughput.
- An analysis of network performance impacts on HPC applications, comparing the LXD deployment and the native one with different number of NICs aggregated.

This paper is organized as follows. The related work is described in Section II. The implementation and validation are presented in Section III. Section IV presents an experimental evaluation of the proposed solution with HPC applications. Finally, Section V discusses and draws the paper's conclusion as well as future works.

978-1-7281-2999-0/19/$31.00 ©2019 IEEE

II. Related Work

Provide network performance optimization for applications executing in cloud environments is a relevant aspect. In this work, we consider as related works those that tackle network performance optimization for cloud environments. The selected related works are described and compared to our work. For instance, Vogel *et al.* [5] conducted a network performance evaluation using CloudStack IaaS manager, deploying KVM and LXC-based clouds. They measured the network throughput and latency, and they indicated alternatives for network performance improvements. They used *vhost-net* module for optimizing the network performance. Results showed that KVM achieves fair throughput rates but performance degradation regarding latency. On the other hand, LXC presented a better performance on latency but lacked support and compatibility. In contrast, in this work, the focus is to implement and validate NIC aggregation with different cloud deployments for intensive HPC applications.

Shafer [10] evaluated network I/O performance in a private cloud environment, which uses different Eucalyptus deployments. The experimental evaluation covered different native OS and hypervisors (Ubuntu with KVM and CentOS with Xen). Moreover, Shea *et al.* [4] analyzed the Xen hypervisor architecture and the network performance of EC2 instances on the geographically distributed area (WAN). Importantly, Shea *et al.* [4] provided performance optimization for intensive applications. Differently than Shafer [10] and Shea *et al.* [4], in this work, we focus on providing performance optimizations with NIC aggregation on different cloud deployments for intensive HPC applications.

The study of Wang *et al.* [11] proposed the applicability of the MultiPath TCP (MPTCP) protocol for improving the performance of a distributed Hadoop/MapReduce architecture. Their scenario exploited GPUs resources and shown the impact of network bottlenecks on the application's performance. Noteworthy, the network link aggregation scheme reduced data transfer time and, consequently improved the overall performance of executing applications. In contrast, our work focuses on applying/implementing NIC aggregation with network bonding mode 0 known as balance round-robin in favor of improving the network performance. Differently from other works, our scenario covers HPC applications execting on real-world cloud environments and assessing their performance with different deployment scenarios.

Rista *et al.* [9] create an evaluation methodology of performance measurements like bandwidth, throughput, latency and completion times of Hadoop applications. In its evaluation, they employed the Network Bonding mode 4 (IEEE 802.3ad) and executed Hadoop up to 3 instances concurrently in LXC containers. As a result, they obtained performance improvements in reducing the execution time of the Hadoop applications in about 33.73%. Differently, we extended the implementations for using more NICs with different environments and technologies. Besides, while Rista *et al.* [9] focus on big data applications, we are targeting HPC applications.

III. Implementation

Cloud computing was built upon other consolidated technologies. The key among these technologies is the virtualization, which dynamically abstract hardware resources such as memory, CPU, storage, and even network for cloud instances [12]. Although virtualization adds more complexities regarding, *e.g.,* management of resources, meaningful optimizations and kinds of virtualization like lightweight virtualization were created to avoid performance losses.

On the other hand, network interconnection is still a significant bottleneck for distributed workloads executing on cloud environments and demands continuous performance characterization and optimizations [5], [13], [7]. In this work, we evaluate the feasibility of using NIC aggregation for improving the performance of HPC applications.

A. NIC Aggregation

Considering the need for optimizing the network performance for HPC applications, NIC aggregation is a potential solution. First, we started our NIC aggregation configuration in the native scenario[1] by setting up the network bonding. This configuration was performed in the Linux network service by selecting up to four interfaces to be slaves from an aggregated virtual interface. Then, the bond mode was selected that uses the balance round-robin (balance-rr or mode 0).

Round-robin (RR) is a well-known and widely adopted algorithm in network schedulers which stripes the packets in sequential order, from the first available interface to the last one [14]. This model was selected because it is the only one (among Ethernet Bonding modes) that can turn a single TCP/IP connection stream to use more than one interface, potentially improving the throughput.

As we intended to use the bond implementation in virtualized environments, which usually exploit Linux bridges to provide network access to executing instances, the aggregated interface was bridged. Finally, the routing table has also been modified, making the bridged interface the default one for incoming/outgoing traffic. The same configuration was followed in the second host, and additional hosts could be easily added by replicating this setup.

B. Cloud Deployment

OpenNebula 5.6.1 was configured in the first host acting as the cloud manager, which orchestrated the containers, and computing node. Then, we added the second host as a computing node for supporting the cloud instances. We configured OpenNebula to use the linux bridge interface (previously created in the native scenario during NICs aggregation configuration Section III-A) that provides network connectivity. Thus, the LXD instances were able to exploit potential optimizations from the aggregated NICs at the host OS. It is important to note that any cloud instances that use bridges interfaces for interconnection, such as, *e.g.,* virtual machines (VMs), could also benefit from the NICs aggregated.

[1]We defined as scenarios the native (without virtualization) and LXD-based cloud and the deployed NIC aggregation environments with 1, 2, 3 or 4 NICs.

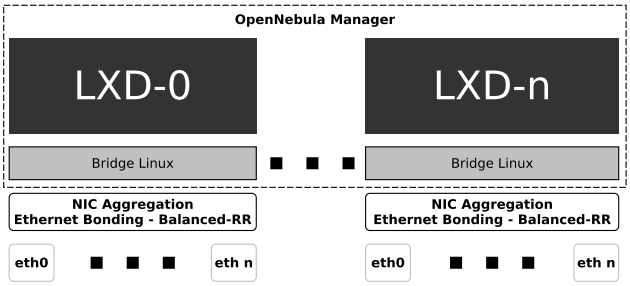

Fig. 1. Scalable conceptual high-level model of NIC aggregation. Adapted from [9].

In our approach, we used OpenNebula because it was already deployed in our environment and also is a representative private cloud manager. Linux containers were used because of their smaller overhead compared to VMs using full and paravirtualization. Containers offer lightweight OS-level virtualization for instances to avoid processing penalties caused by the abstraction of additional layers and hardware emulation. Thus, the cloud environment can achieve near-native performance [15].

The most recent Linux container open source project is known as LXD [16]. It is built on top of LXC, providing both improvements and new features with other functionalities to create and manage containers with fine-grained control and operational security. The conceptual model for creating containers is the same employed by LXC, using namespaces and cgroups and also Linux bridges for the network. However, LXD provides a REST API over Unix socket that can also be enabled over the network as well, to the management of local/remote hosts.

As LXD has currently no native implementations upon OpenNebula, we used the LXDone[2] project to create and manage LXD templates under version 3.0.3. One instance with this template was deployed in each node. Besides, the deployed LXD-based cloud had dedicated access to the hardware resources. A representation of scalable conceptual model using NICs aggregated for interconnecting a cloud deployment is depicted in Figure 1, which highlights the possibility of using several NICs in the form of computer infrastructure supporting potentially large scale cloud environments.

C. Deployment's Network Evaluation

To evaluate the implementations, we used a cloud environment composed by two HP ProLiant servers with identical hardware configurations; each node has two six-core AMD Opteron processor 2425 HE and 32GB of RAM, 4 Intel Gigabit network interface cards (NICs) interconnected by a Gigabit Switch. The operating system used was Ubuntu Server 18.04 64-bit (kernel 4.15.0-45). It is important to note that the performance experiments use the network interfaces with aggregated traffic in the containers. The native scenario that uses an OS without cloud and containers layer also exploits the aggregated interfaces.

The network speed was evaluated by executing NetPipe benchmark between two hosts, which provide the results of throughput and latency. NetPipe (Network Protocol-Independent Performance Evaluator) [17] is an intensive network communication benchmark, which exposes the network performance under a variety of conditions. It chooses the message sizes at regular intervals and with variations to fully evaluate the network. Also, latencies are calculated considering the round trip time (RTT).

In this evaluation, we used NetPipe 5.1 compiled with MPI module to measure aggregated network throughput and latency between the nodes. Also, the options `async` which use asynchronous "MPI_Irecv()" to pre-post and `bdir` which send data in both directions (download and upload) at the same time were used. `Bidir` option was used because multiple overlapping unidirectional communication may become out of sync. However, this option also outputs the combined throughput. As the number of cores by each server is 12, NetPipe was configured to use 24 processes simultaneously, pinned to 12 processes in each server.

In Figure 2, the TCP latency is plotted and presented in microseconds concerning the message size in kilobytes (kB). To show more representative results, we choose the interval of message size increasing of 1 to 100 kB and plotted the latency with 2 and 4 NICs aggregated in native and LXD-based cloud respectively. In general, latency performance has been significantly improved on both LXD and native scenario when more NICs are aggregated.

As can be seen, in the environment with 1 NIC, a similar performance on both native and LXD-based cloud scenarios was achieved. Besides, with two NICs, both LXD-based cloud and native overcomes the baseline. LXD-based cloud overcomes native scenario with message sizes about 10 to 30 kb. Between 40 and 60 kB, both have similar results. With message size greater than 70 kB, the native scenario shows improved performance. With 4 NICs aggregated, the results show performance improvements compared to 2 NICs. This aspect is relevant because it demonstrates that latency has improved as more NICs were aggregated. When using 4 NICs, LXD-based cloud overcomes native scenario from about 10 kB message size. The difference between them increases according to the message sizes.

The results of throughput are presented in Gigabits per second (Gbps) concerning message size in kilobytes (kB) in Figure 3 and shown that performance between 1 NIC from LXD and native scenario has no significant differences. However, both 1 NIC - native and LXD reaches almost 2 Gbps. This performance is reached because NetPipe used the `bidir` command line option, which outputs the combined throughput. This parameter was used to represent real-world applications that are network intensive, sending, and receiving data at the same time.

In the evaluation with 2 NICs aggregated using message sizes between 10 to 50 kB, the LXD scenario significantly

[2]https://github.com/OpenNebula/addon-lxdone

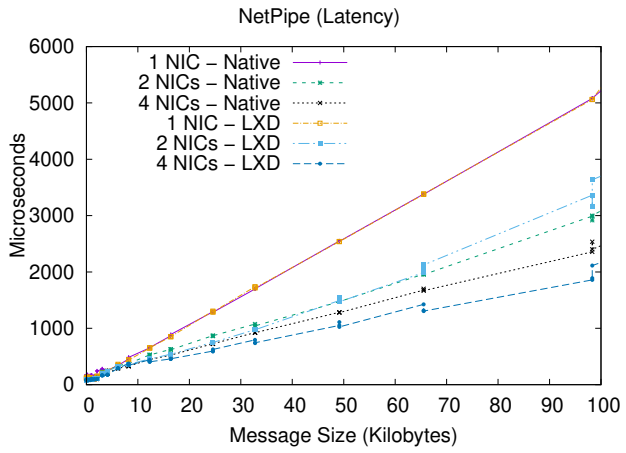

Fig. 2. TCP Latency with NIC Agreggation (Less is better).

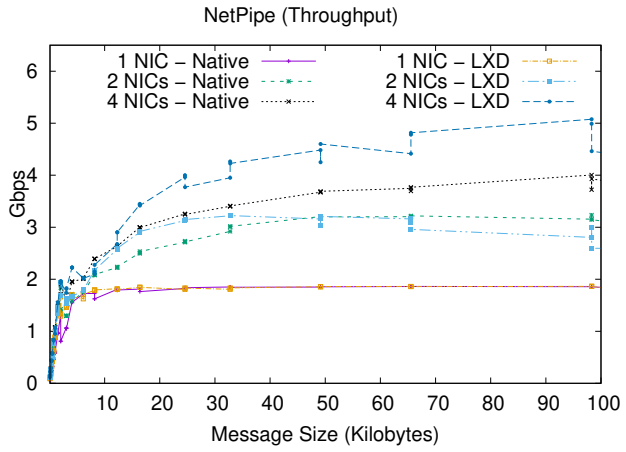

Fig. 3. TCP Throughput with NIC Aggregation (More is better).

overcomes the native one, reaching maximum performance with 30 kB message size and 3.2 Gbps. After that, LXD-based cloud starts to decrease performance until it is outperformed by the native scenario with 65 kB. Regarding the 4 NICs aggregation, when message size reaches about 10 kB, the LXD-based cloud overcomes the native with significant difference achieving the maximum throughput of 5 Gbps.

IV. Experimental Evaluation

In Section III-C, we obtained a first perspective of how NIC aggregation impacts in the network interconnection. We designed our approach to evaluate representative applications from real-world and verify if the applications can obtain performance improvements provided by NIC aggregation.

A. Benchmarks and Methodology

We chose to conduct our evaluation using applications from Numerical Aerodynamic Simulation Parallel Benchmarks (NPB) suite [18] version 3.3.1 compiled with MPI. NAS was designed to aid performance benchmarking of parallel supercomputers. Derived from computational fluid dynamics (CFD) applications, these benchmarks can represent a wide variety of HPC applications.

TABLE I
Overview of the NAS benchmarks used in the evaluation.

Name	Description	Focus	Language
BT	Block Tridiagonal	Floating point performance	Fortran
FT	Fast Fourier Transform	All to All communication	Fortran
IS	Integer Sort	Integer performance	C
SP	Scalar Pentadiagonal	Floating point performance	Fortran

NAS was compiled with class C problem size and all five kernels (IS, EP, CG, MG, and FT) and three pseudo-applications (BT, SP, and LU) were used. Also, the applications used 16 processes because each one of them requires a specific division of work. The number of processors must be a square root, or even it must be a power of two. As the nodes have 12 cores, 8 processes were set to execute on each node to balance the load distribution.

However, for the sake of space, we only present representative tests from FT, IS, BT, and SP. These experiments demonstrate whether network NIC aggregation can significantly improve the performance of the applications. Table I summarizes these benchmarks. Applications like EP, CG, MG, SP, and LU shows no significant differences in performance regarding execution time when they were executed in our 4 environments (1, 2, 3, and 4 NICs). Moreover, in this paper, we are not focusing on the number of MPI processes that scale the performance. Hence, we execute all applications with 16 processes to use a suitable value for the tested machines, which generates enough communication and consequently, network traffic for evaluating the network optimization.

The experiments were executed 10 times on dedicated identical machines (described in Sectionn III-C) and presented an average performance with the related standard deviation. Moreover, to measure network usage, network NICs statistics were collected using the `ifstat` utility.

B. Performance of HPC Applications

In Figure 4 and 5 are plotted the execution time of the BT and SP applications under our two scenarios (native and LXD-based cloud) with the four environments (1 NIC as a baseline and 3 NICs using NIC aggregation). As can be seen, BT and SP in native scenario obtained worse performance results when aggregation is applied. BT and SP are applications with executions under blocking operations (*e.g.,* barriers). Consequently, as a result of the BT and SP characteristics, the network aggregation was not exploited by the low utilization, which is the reason for not improving the performance even when using several aggregated NICs.

FT kernel results are shown in Figure 6. This application performs reduction operations and all to all communication, which creates an extremely network utilization pattern [19]. FT can significantly improve its execution time up to 36% with 3 NICs aggregated. This performance gain follows throughput improvements with 3 NICs (about 97 MB/s). Moreover, performance differences with 3 and 4 NICs between native and

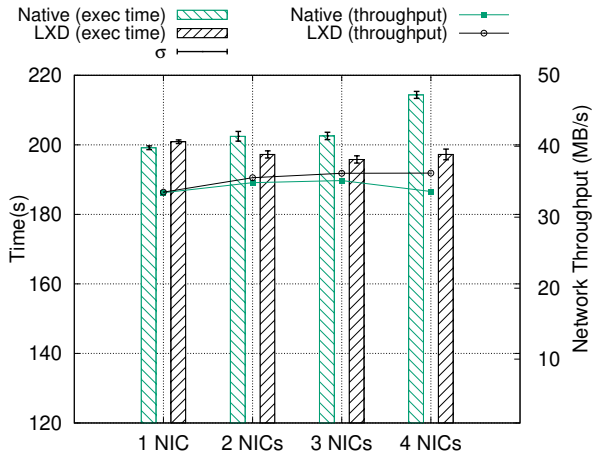

Fig. 4. BT experiments with 16 processes.

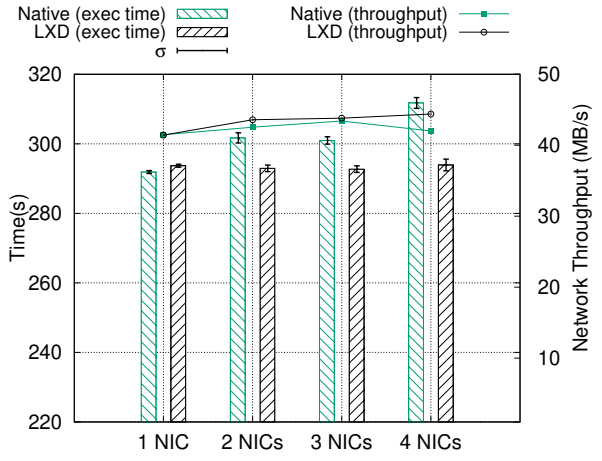

Fig. 5. SP experiments with 16 processes.

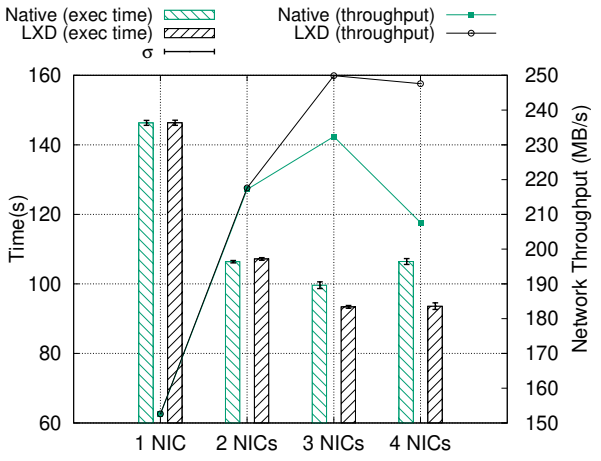

Fig. 6. FT experiments with 16 processes.

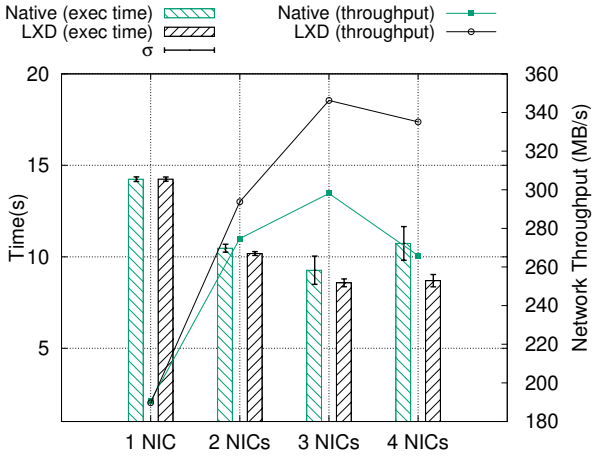

Fig. 7. IS experiments with 16 processes.

LXD-based cloud scenarios are most significant. With the help of the right-hand side of Figure 6 is explicit that the throughput plays a crucial role in the performance of these applications.

Figure 7 shows the experiments of IS, which is characterized by intensive computations during the majority of its execution, using the network intensively during reduce/accumulative operations that gather results from all executing processes. IS is also characterized by only negligible synchronization. In resume, the IS executions using network aggregation improved the performance by faster communicating and collecting results. The intensive use of the network by this application can also be viewed in Figure 7. The performance gains can be seen mostly in the LXD-based cloud scenario where the execution time was reduced 38.95% with 4 aggregated NICS.

A comparison between results from NIC aggregation on native and LXD-based cloud scenarios emphasize that LXD achieved higher performance. The LXD performance optimization (*e.g.,* namespace isolation) enabled such scenario to outperform the native one. Performance optimization of containers executing with multi-threaded/distributed applications is known in the related literature [5], [3]. The unexpected result came from the native scenario, which has shown performance loses when using 3 and 4 aggregated NICs. The reason for degraded performance in the native scenario is related to the combinations of the applications and the low-level aspects from the network implementation.

The overhead related to NIC aggregation is due to the re-transmission of packets. The balance-rr aggregation mode stripes the network packets between the physical interfaces, which resulted in additional unordered packets. Consequently, the high number of unordered packages exceeded the congestion buffer limit, then the TCP/IP's congestion control flushed unordered packets. Hence, the flushed packets had to be re-transmitted, causing processing slowdown due to the communication delay and so reducing the application performance. The re-transmission also caused additional network traffic. This overhead of the native scenario mainly with BT and FT using 3 and 4 NICs can be viewed in Table II, which shows the total network traffic during the application's execution.

TABLE II
NETWORK TRAFFIC (DOWNLOAD + UPLOAD) IN MB GENERATED DURING THE APPLICATION'S EXECUTION.

	Native				LXD			
	1 NIC	2 NICs	3 NICs	4 NICs	1 NICs	2 NICs	3 NICs	4 NICs
BT	6662	7060	7123	7208	6736	7016	7087	7145
SP	12106	12836	13065	13097	12170	12769	12823	13049
FT	22314	23105	23153	22101	22333	23320	23337	23174
IS	2714	2876	2761	2853	2703	2991	2971	2916

V. CONCLUSION

Considering that communication can cause overheads for HPC applications executing on cloud environments, we aimed at mitigating this problem by proposing an approach using NICs aggregation. The results showed that our NIC aggregation approach integrated into the cloud significantly improved the network performance by providing higher throughput while reducing the latency. NIC aggregation showed the potential to be easily integrated with other virtualization technologies that use network bridging.

NICs aggregation for HPC application evinced performance improvements and slight losses in other cases. IS and FT applications make intensive use of the network, which enabled to improve the performance up to 36% (FT with 3 NICs) and 38.95% (IS with 4 NICs) due to the faster communication as more NICs were aggregated. On the other hand, BT and SP applications present worse results when executed in the native scenario. However, the same worse performance is not shown when executing in LXD-based cloud.

The results achieved show that on scenarios where the network infrastructure is scalable in terms of NICs, cables and switch ports, straightforward implementations with a ready-to-use integration to a cloud environment can provide significant performance improvements. Although the experiments were executed on two nodes, we expect that the performance trends would be similar in large scale environments, especially when executing applications that use the network intensively for exploiting the optimization provided by NICs aggregation. We argue that network optimizations are relevant for improving the performance of computational intensive cloud environments.

In future works we plan to: (I) optimize the performance of Round-robin algorithm to reduce the TCP congestion; (II) evaluate network interference between multi-tenant instances; (III) investigate the possibility to obtain higher network performance with the MultiPath TCP.

ACKNOWLEDGMENT

This work has been partially supported by the projects; 1) "GREEN-CLOUD: Computação em Cloud com Computação Sustentavel" (#16/2551-0000 488-9), from FAPERGS and CNPq Brazil, program PRONEX 12/2014. 2) Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001. 3) FAPERGS 01/2017-ARD project PARAELASTIC (No. 17/2551-0000871-5). 4) School of Technology from PUCRS. Finally, we thank the Laboratory of Advanced Research on Cloud Computing (LARCC) for providing computing resources.

REFERENCES

[1] P. Mell, T. Grance *et al.*, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology (NIST)*, 2011.

[2] A. Vogel, D. Griebler, C. A. F. Maron, C. Schepke, and L. G. Fernandes, "Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack," in *24th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Heraklion, Greece, 2016, pp. 672–679.

[3] D. Griebler, A. Vogel, C. A. F. Maron, A. M. Maliszewski, C. Schepke, and L. G. Fernandes, "Performance of Data Mining, Media, and Financial Applications under Private Cloud Conditions," in *23rd IEEE Symposium on Computers and Communications (ISCC)*, Natal, Brazil, 2018, pp. 450–456.

[4] R. Shea, F. Wang, H. Wang, and J. Liu, "A Deep Investigation Into Network Performance in Virtual Machine Based Cloud Environments," in *33rd IEEE Conference on Computer Communications (INFOCOM)*, Toronto, Canada, 2014, pp. 1285–1293.

[5] A. Vogel, D. Griebler, C. Schepke, and L. G. Fernandes, "An Intra-Cloud Networking Performance Evaluation on CloudStack Environment," in *25th Euromicro Int. Conference on Parallel, Distributed and Network-based Processing (PDP)*, St. Petersburg, Russia, 2017, pp. 468–472.

[6] L. Chao, *Cloud Computing Networking: Theory, Practice, and Development*. CRC Press, 2015.

[7] E. Roloff, M. Diener, L. P. Gaspary, and P. O. A. Navaux, "HPC Application Performance and Cost Efficiency in the Cloud," in *25th Euromicro Int. Conference on Parallel, Distributed and Network-based Processing (PDP)*, St. Petersburg, Russia, 2017, pp. 473–477.

[8] J. Moura and D. Hutchison, "Review and Analysis of Networking Challenges in Cloud Computing," *J. Netw. Comput. Appl.*, vol. 60, no. C, pp. 113–129, 2016.

[9] C. Rista, D. Griebler, C. A. F. Maron, and L. G. Fernandes, "Improving the Network Performance of a Container-Based Cloud Environment for Hadoop Systems," in *15th Int. Conference on High Performance Computing & Simulation (HPCS)*, Genoa, Italy, 2017, pp. 619–626.

[10] J. Shafer, "I/O Virtualization Bottlenecks in Cloud Computing Today," in *Proceedings of the 2Nd Conference on I/O Virtualization*, Berkeley, USA, 2010, pp. 5–5.

[11] C. Wang, C. Yang, W. Liao, R. Chang, and T. Wei, "Coupling GPU and MPTCP to improve Hadoop/MapReduce performance," in *2016 2nd Int. Conference on Intelligent Green Building and Smart Grid (IGBSG)*, Prague, Czech Republic, 2016, pp. 1–6.

[12] R. Buyya, C. Vecchiola, and S. T. Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*. Newnes, 2013.

[13] E. Roloff, M. Diener, A. Carissimi, and P. O. A. Navaux, "High Performance Computing in the cloud: Deployment, performance and cost efficiency," in *4th IEEE Int. Conference on Cloud Computing Technology and Science Proceedings (CloudCom)*, Taipei, Taiwan, 2012, pp. 371–378.

[14] T. Davis, W. Tarreau, C. Gavrilov, C. N. Tindel, J. Girouard, J. Vosburgh, J. Vosburgh, and M. Williams, "Linux Ethernet Bonding <https://www.kernel.org/doc/Documentation/networking/bonding.txt>," 2011, last access fev, 2019.

[15] I. M. A. Jawarneh, P. Bellavista, L. Foschini, G. Martuscelli, R. Montanari, A. Palopoli, and F. Bosi, "QoS and Performance Metrics for Container-based Virtualization in Cloud Environments," in *20th Int. Conference on Distributed Computing and Networking (ICDCN)*, Bangalore, India, 2019, pp. 178–182.

[16] LXD, "LXD < https://linuxcontainers.org/lxd/introduction/>," 2019, last access fev, 2019.

[17] Q. O. Snell, A. R. Mikler, and J. L. Gustafson, "Netpipe: A Network Protocol Independent Performance Evaluator," in *Int. Conference on Intelligent Inf. Management and Systems*. Washington, USA, 1996.

[18] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS parallel benchmarks summary and preliminary results," in *ACM/IEEE Conference on Supercomputing (SC)*, Albuquerque, USA, 1991, pp. 158–165.

[19] A. Faraj and X. Yuan, "Communication Characteristics in the NAS Parallel Benchmarks," in *Int. Conference on Parallel and Distributed Computing and Systems (PDCS)*, Cambridge, USA, 2002, pp. 724–729.