

Performance Impact of IEEE 802.3ad in Container-Based Clouds for HPC Applications

Anderson M. Maliszewski[1,3(✉)], Eduardo Roloff[1], Dalvan Griebler[2,3], Luciano P. Gaspary[1], and Philippe O. A. Navaux[1]

[1] Informatics Institute, Federal University of Rio Grande Do Sul (UFRGS), Porto Alegre, Brazil
{ammaliszewski,eroloff,paschoal,navaux}@inf.ufrgs.br
[2] Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
dalvan.griebler@acad.pucrs.br
[3] Laboratory of Advanced Research on Cloud Computing (LARCC), Três de Maio Faculty (SETREM), Três de Maio, Brazil

Abstract. Historically, large computational clusters have supported hardware requirements for executing High-Performance Computing (HPC) applications. This model has become out of date due to the high costs of maintaining and updating these infrastructures. Currently, computing resources are delivered as a service because of the cloud computing paradigm. In this way, we witnessed consistent efforts to migrate HPC applications to the cloud. However, if on the one hand cloud computing offers an attractive environment for HPC, benefiting from the pay-per-use model and on-demand resource allocation, on the other, there are still significant performance challenges to be addressed, such as the known network bottleneck. In this article, we evaluate the use of a Network Interface Cards (NIC) aggregation approach, using the IEEE 802.3ad standard to improve the performance of representative HPC applications executed in LXD container based-cloud. We assessed the aggregation impact using two and four NICs with three distinct transmission hash policies. Our results demonstrated that if the correct hash policy is selected, the NIC aggregation can significantly improve the performance of network-intensive HPC applications by up to ≈40%.

Keywords: Cloud computing · NIC aggregation · Bonding · LXD

1 Introduction

In recent years, consistent efforts have been made to migrate HPC applications to cloud computing (CC). This approach is based mainly on the use of the characteristics of these environments, such as; pay-per-use model and access to a shared pool of computing resources. Thus, when a request is made, resources are provisioned on-demand in a scalable/elastic manner and released almost instantly with minimal effort, requiring only a network connection to access [10]. However, with the increasing adoption of CC, some challenges were also posed, including

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 158–167, 2020.
https://doi.org/10.1007/978-3-030-58817-5_13

the virtualization impact, which presents an overhead due to its additional layer when compared to the native environment as well as the multi-tenants who share/compete for resources (*i.e.*, network interconnection) [6,9].

HPC applications used in clouds are often developed with the Message Passing Interface (MPI) standard to take advantage of the underlying distributed system. Since this type of workload typically handles large amounts of data, requiring high computing power, a key factor explored is how to speed up its performance, as this consequently reduces allocation costs. Such performance improvements can be made with both software and hardware optimization. On the hardware side, the network interconnection is pointed out by several studies as one of the main bottlenecks [11,13,14], because it is the central point of interconnection between servers, and thus also be shared between the processing of several flows originated from different cloud instances. In addition, another alternative is to allocate a faster cloud instance, which can increase overall performance, but theoretically also increases the cost.

In previous studies, we introduced our first Network Interface Cards (NIC) aggregation approach, which used bonding mode 0, also known as Balance Round Robin to improve network performance using a private cloud deployment [8] with LXD containers. The results highlighted that parallel applications with network-intensive patterns increased their performance by up to 38%. Thus, we argue that this approach could be employed by public cloud providers, taking advantage of existing hardware, increasing performance, and reducing costs. In this work, we make progress on the state-of-the-art by evaluating a different NIC aggregation approach which used the bonding mode 4, also known and specified as IEEE 802.3ad. We used 4 synthetic HPC applications, and executed them with two and four NICs aggregated, using three distinct hash policies (layer 2, layer 2 + 3, and layer3 + 4), and the environment without NICs aggregated (Regular TCP) as a baseline to the results. As far as we know, there is no other work in the literature that performs the same studies. With this assessment we provide the following contributions:

- A reproducible performance assessment of the IEEE 802.3ad NIC aggregation standard implemented with three distinct hash policies and two different numbers of aggregated NICs, relative to a baseline without aggregation, in container-based cloud deployments, evaluating their impact in the performance of representative synthetic high-performance applications.
- We demonstrate that the NIC aggregation approach deployed with the cloud instances can significantly improve the applications performance by up to ≈40% when the correct transmission hash policy is selected.

The remainder of the paper is organized as follows. In Sect. 2, we present our methodology, concerning hardware/software specification, NIC aggregation, and transmission hash policies description, alongside as our private cloud deployment, benchmarks used, and the experimental setup. Next, in Sect. 3 we present the evaluation and discuss the obtained results. In Sect. 4 we cover some of the most prominent related works. Finally, in Sect. 5, we conclude the paper with some final remarks and prospective directions for future research.

2 Scope and Methodology

This Section describes the hardware/software specifications alongside with the NIC aggregation approach, in which the bonding mode 4 or 802.3ad and its different transmit hash policies are covered. Also, the private cloud management platform, benchmarks, and the experimental setup used to conduct the experiments are reported.

2.1 Hardware/Software Specifications

The computational environment which has supported our experiments was composed of four HP ProLiant server with identical hardware resources. Each one has two six-core AMD Opteron processor 2425 HE, 32 GB 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 softwares involved in the evaluation process were used with their last stable available version. The LXD instances were created using the LXC versions 3.0.3 and used the same OS, MPI wrapper, and GCC version as the physical servers.

2.2 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, etc.) to improve bandwidth and resilience in a cost-effective way, by merely adding new links together with existing ones instead of replacing equipment [1,5]. The specific behavior of connected interfaces is based on the choice of a mode of use, among seven existing modes. Another equally important use of NIC aggregation is to fail over 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 fail over with degraded performance in a NIC failure event.

In our approach, we used up to 4 NICs with the IEEE 802.3ad Dynamic link aggregation. This mode creates aggregation groups that share the same speed and duplex settings. The selection of the slave for outgoing traffic is made according to the transmission hash policy, which can be changed from the standard simple XOR policy using the xmit_hash_policy option. To compare the performance between hash policies, we evaluated three of them, which are the most used. 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 is required to reach most destinations.
- **Layer 3 + 4:** This policy uses upper layer protocol information, when available, to generate the hash. This allows for 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. For fragmented TCP or UDP packets and all other IPv4 and IPv6 protocol traffic, the source and destination port information are omitted. This algorithm is not fully 802.3ad compliant. A single TCP or UDP conversation containing both fragmented and unfragmented packets will see packets striped across two interfaces. This 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.

Limitations. Although NIC aggregating has the potential to improve performance, it also has implications for its usage. For example, 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.

2.3 Private Cloud Deployment

In the deployment of the private cloud environment, we used the OpenNebula manager. It was chased because of being one of the most popular private cloud managers and by following our previous work [8]. Also, we deployed instances using the LXD containers because they use a lightweight virtualization. In the Fig. 1 is depicted the representation of our system. We used four servers, each one with four NICs connected to the same switch. NICs are them grouped into a logical link called bond 0 and bridged to the containers. OpenNebula manages the containers and create a cluster establishing the communication over the underlying bonded NICs.

2.4 Benchmarks

We conduct our evaluation using four HPC benchmarks (IS, FT, BT, and SP) from the Numerical Aerodynamic Simulation Parallel Benchmarks (NPB)

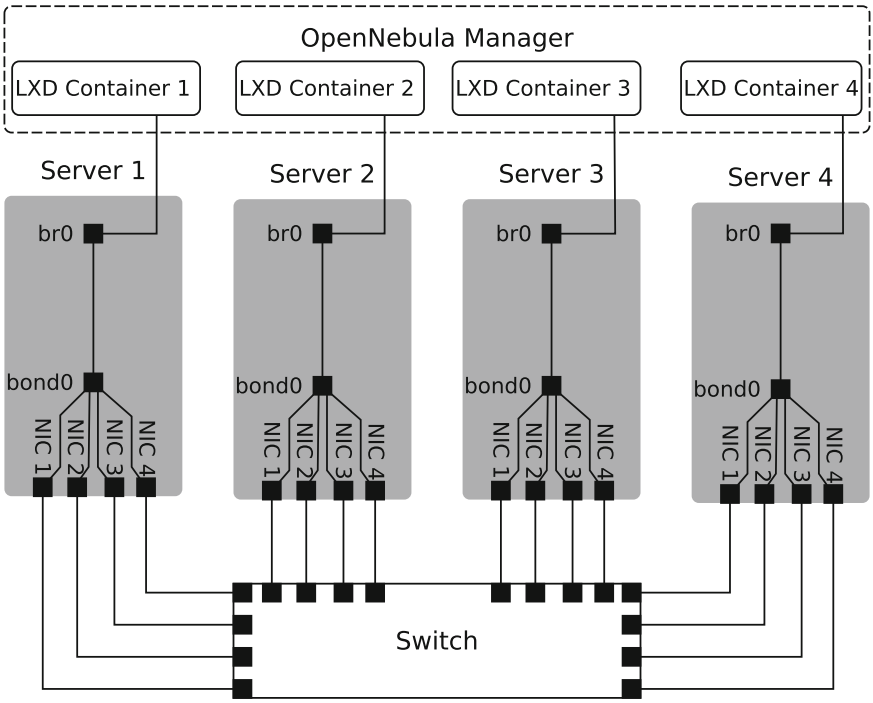


Fig. 1. High level representation of our system.

suite [2]. The NPB set, used with version 3.4, was designed to evaluate the performance of different hardware and software in HPC systems. In this paper, the NPB benchmarks were chosen based on our previous work, which demonstrated that these applications tend to benefit or get worse results when using the NIC aggregation approach. All NAS benchmarks were compiled with size C with `-O3` flag, `mpifort` and `mpicc` for Fortran and C codes, and executed with 32 MPI processes (8 per node). Above is a short description of the NPB benchmarks. **IS** performs a sort of integer keys using a linear time Integer Sorting algorithm on computation of the key histogram. **FT** contains the computational kernel of a 3-D Fast Fourier Transform (FFT). **BT** and **SP** both apply variations of the Alternating Direction Implicit (ADI) approximate factorization technique to decouple solution of the x,y, and z-coordinate directions which results are 5×5 block-tridiagonal and scalar pentadiagonal, respectively [2,12].

2.5 Experimental Setup

We employ a reproducible research methodology [15], using R, Git, and a laboratory notebook. All data collected in this work is publicly available[1]. To guide the experiments execution, for the baseline (Regular TCP), each number of NICs aggregated (802.3ad-2NICs or 802.3ad-4NICs), and different transmit hash policy (layer2, layer2 + 3, and layer3 + 4) used, we generated an experiment design [7], totaling seven individual designs. Different designs were created due to the need to restart the underlying server when a different aggregation or transmission policy was applied. With the reboot process, we also made sure that there was no interference in the experiments related to various levels of cache (e.g., memory, processor instructions). The designs have 30 replications, where the five applications executed were randomized. The reported execution times measurements are averages of the replications, and the error bars were calculated considering a confidence level of 99.7%, assuming a Gaussian distribution.

3 Evaluation

Our performance evaluation results are shown in Fig. 2 for BT, SP, IS, and FT applications, each represented in an individual sub-figure. The execution times using the unit of time measurement in seconds is represented on the Y-axis using different scales. The name of the application is shown on the X-axis.

BT has improved its performance in ≈4.66% when executed using two NICs aggregated and in ≈12.52% when executed with four NICs aggregated with the layer 2 policy. On the other hand, when higher layers policies are used, the execution time was increased. For instance, using the layer 2 + 3 policy and 2 NICs aggregated, we can not stat a difference compared with the execution time of Regular TCP, because the error bars overlap. Using the same policy, but with four NICs, the times were improved in ≈6.30% compared to our baseline, but worse if compared to the layer 2 policy with 4 NICs. Using the higher layer policy (layer3 + 4), BT worsened its performance with 2 NICs in ≈5.87%, and improved in ≈5.18% with four NICs.

SP improved its performance in ≈2.78% with two NICs and ≈16.92% with four NICs aggregated, with the layer 2 policy. Similarly as BT, SP executed with 2 NICs with layer2 + 3 policy has no difference compared to the baseline and with 4 NICs improved ≈8.54%. Finally, with layer3 + 4 policy, SP with 2 NICs aggregated has worse results than the baseline in ≈10.90% and improved its performance in ≈4.83% with four NICs.

IS has a short execution time (less than 15 s for all number of NICs aggregated and policies). Layer 2 shows the best performance, with two NICs aggregated having improved its performance in ≈32.74% and with four NICs in ≈40.37%. With layer 2 + 3 policy the error bars of two interfaces aggregated have a range which stays in the same line as our baseline, so it can not be point out as improved or not. On the other hand, with four NICs in layer2 + 3 IS has improved its

[1] https://github.com/andermm/ICCSA-2020.

performance in ≈15.27%. Finally both two and four NICs aggregated with the layer 3 + 4 policy lost performance compared to the baseline, in ≈5.32% and ≈8.13%, respectively.

FT follows a similar pattern as other applications on layer 2 and layer 2 + 3. For instance, layer 2 with two NICs has improved its performance in ≈23.39% and with four NICs in ≈26.56%. Also, in layer 2 + 3 FT with two NICs has error bars overlapped with baseline and an improvement in ≈7.86% with four interfaces. Contrary to the other applications behaviour, FT improved its performance in both number of NICs aggregated using the policy 3 + 4, in ≈10.94% with two network interfaces and in ≈4.83% with four NICs. FT application is known to be Communication-Bound spending almost its entire execution doing MPI operations, with a slight portion of Computing. This application sends a considerable amount of small messages, which turns it latency-sensitive.

4 Related Work

Aggregation of NICs has been explored with different purposes and approaches using techniques such as bonding or MultiPath TCP (MPTCP) to improve performance. In our previous work [8], we focused on implementing NIC aggregation with network bonding mode 0 - balance round-robin. Our results highlighted the aggregation potential to improving the performance of HPC applications.

Watanabe *et al.* [16] investigated the impact of topology and link aggregation on a large-scale PC cluster with Ethernet. They performed several experiments 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 hardware side.

Chaufournier *et al.* [4] 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.* [17] 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 application performance. As a result, it demonstrated that the use of aggregation of network links reduced the data transfer time and improved the overall performance.

Rista *et al.* [3] created a methodology for evaluating performance measures such as bandwidth, throughput, latency and execution times for Hadoop applications. In the assessment, they 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 of, they achieved performance improvements by reducing application times in ≈33%. Although the results obtained are promising, the use of simultaneous instances, also known as

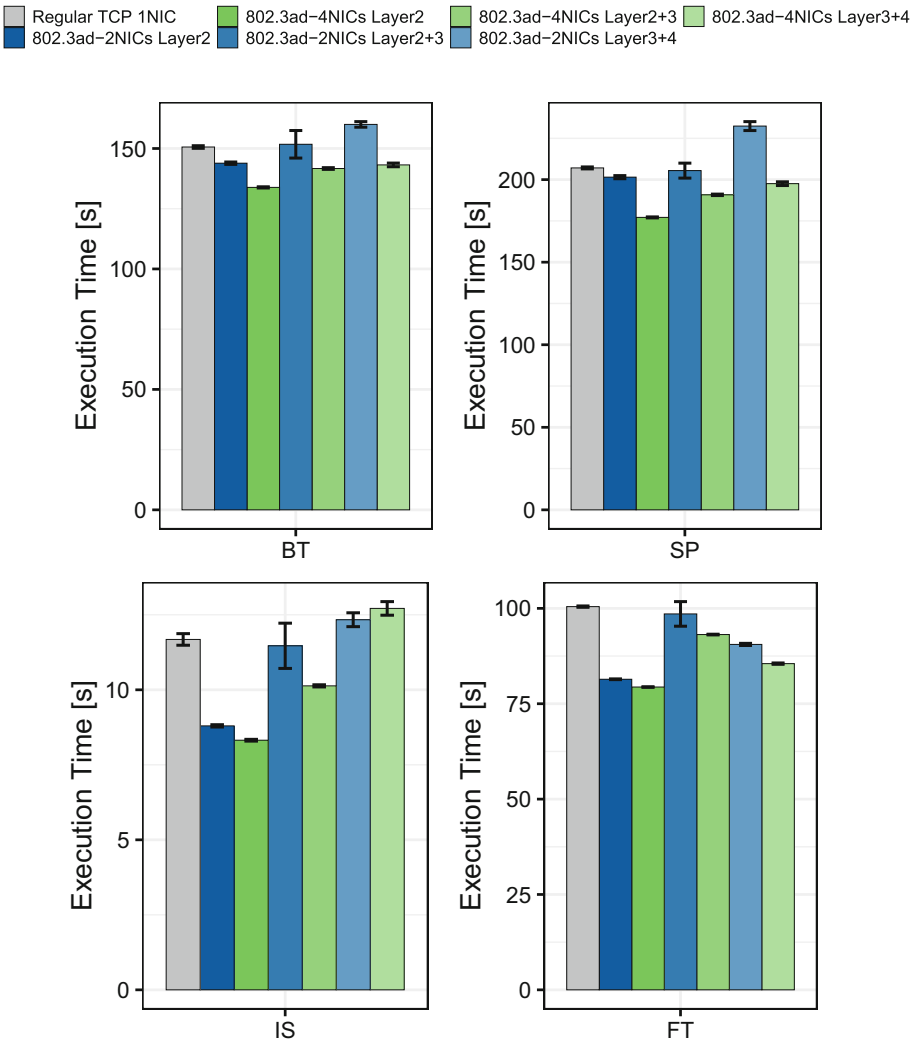


Fig. 2. Performance results of BT, SP, IS and FT applications executed using Regular TCP, two, and four NICs aggregated with bonding mode 4 - 802.3ad for Regular TCP as baseline, layer2, layer2 + 3, and layer3 + 4 transmission hash policies.

multi-tenant, does not apply to HPC applications, as these require that there is no competition for computational resources.

In contrast to the previous works, our evaluation focuses on applying/implementing NIC aggregation with network bonding mode 4 or IEEE 802.3ad to reduce the execution time of HPC applications. Differently for other works, our scenario covers HPC applications running on real-world cloud environments and assessing their performance with two and four NICs aggregated. Finally, we

also evaluated different transmission hash policies algorithm to discover which of them offer the best performance.

5 Final Considerations and Future Work

This work sought to evaluate the performance of link aggregation in LXD instances with three different transmission hash policies in comparison with the baseline of a single network interface (Regular TCP). Considering that network interconnection can cause overhead for HPC applications executing in cloud environments, our goal was to evaluate an approach using NIC aggregation.

The results showed that our NIC aggregation approach integrated into the cloud improved the applications performance in the majority of the hash policies implementations. For instance, IS applications, which have a short execution time, improved its performance with four NICs aggregated in ≈40.37%. Other applications like BT and SP which have blocking MPI operations have not fully exploited NIC aggregation by the low network utilization. We can highlight a pattern that all applications followed, which is a significant gain of performance using the layer 2 policy, and a scalar loss of performance as hash policies that use higher layers are applied (i.e., layer 2 + 3 and layer 3 + 4). The only application that does not followed exactly this pattern is FT, which could improve its performance using layer 3 + 4 for both two and four NICs aggregated. This happens because of application characteristics, in which FT makes intensive use of the network, allowing it to improve its performance.

In addition, when comparing the use of two against four NICs aggregated, we can see a gain of performance on all applications using four NICs between any hash policy used. This shows the potential of NIC aggregation to improve the performance. In our experiences, we also can highlight that this implementation can be easily integrated into a ready-to-use cloud environment through the use of linux bridges. In future work we plan to: (I) compare the results obtained with the use of IEEE 802.3ad mode against the balanced round-robin mode, evaluated in our previous article [8]. (II) evaluate this approach with other virtualization technologies. (III) assess this environment with a wide range of real applications, considering more complicated scenarios (i.e real-time environments).

Acknowledgment. This work has been supported by the projects; 1) "GREEN-CLOUD: Computação em Cloud com Computação Sustentável" (#16/2551-0000 488-9) from FAPERGS and CNPq Brazil, program PRONEX 12/2014. 2) Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. 3) FAPERGS 01/2017-ARD project ParaElastic (N° 17/2551-0000871-5) and the Universal MCTIC/CNPq N° 28/2018 project SParCloud (No. 437693/2018-0). 4) BRICS Pilot Call 2016 project CloudHPC. 5) CNPq/MCTIC/BRICS-STI No 18/2016 Project Number 441892/2016-7. Finally, we thank the Três de Maio Faculty (SETREM) and the Laboratory of Advanced Research on Cloud Computing (LARCC), for providing access to computational infrastructure.

References

1. 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, pp. 1–184 (2000)
2. Bailey, D., et al.: The NAS parallel benchmarks; summary and preliminary results. In: ACM/IEEE Conference on Supercomputing (SC), pp. 158–165 (1991)
3. Rista, C., Griebler, D., Maron, C.A.F., Fernandes, L.G: Improving the network performance of a container-based cloud environment for hadoop systems. In: 15th International Conference on High Performance Computing & Simulation (HPCS) (2017)
4. Chaufournier, L., Ali-Eldin, A., Sharma, P., Shenoy, P., Towsley, D.: Performance evaluation of multi-path TCP for data center and cloud workloads. In: ACM/SPEC International Conference on Performance Engineering, pp. 13–24 (2019)
5. Davis, T., et al.: Linux Ethernet Bonding (2011). https://www.kernel.org/doc/Documentation/networking/bonding.txt
6. Gupta, A., et al.: Evaluating and improving the performance and scheduling of HPC applications in cloud. IEEE Trans. Cloud Comput. (TCC) (2016)
7. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons Inc, Digital Equipment Corporation-Litleton, Massachusetts (1991)
8. Maliszewski, A.M., Vogel, A., Griebler, D., Roloff, E., Fernandes, L., Navaux, P.O.: Minimizing communication overheads in container-based clouds for HPC applications. In: IEEE Symposium on Computers and Communications (ISCC), pp. 1–6 (2019)
9. Mauch, V., Kunze, M., Hillenbrand, M.: High performance cloud computing. Future Gen. Comput. Syst. **29**(6), 1408–1416 (2013)
10. Mell, P., Grance, T., et al.: The NIST definition of cloud computing. Nat. Inst. Stand. Technol. (NIST) (2011)
11. Pretto, G.R., et al.: Boosting HPC applications in the cloud through JIT traffic-aware path provisioning. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11622, pp. 702–716. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24305-0_52
12. Ramachandran, A., Vienne, J., Van Der Wijngaart, R., Koesterke, L., Sharapov, I.: Performance evaluation of NAS parallel benchmarks on Intel Xeon Phi. In: International Conference on Parallel Processing (ICPP), pp. 736–743 (2013)
13. Roloff, E., Diener, M., Gaspary, L.P., Navaux, P.O.A.: HPC application performance and cost efficiency in the cloud. In: Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 473–477 (2017)
14. Sadooghi, I., et al.: Understanding the performance and potential of cloud computing for scientific applications. IEEE Trans. Cloud Comput. (TCC) **5**(2), 358–371 (2017)
15. Stanisic, L., Legrand, A., Danjean, V.: An effective Git and org-mode based workflow for reproducible research. ACM SIGOPS Oper. Syst. Rev. **49**(1), 61–70 (2015)
16. Watanabe, T., Nakao, M., Hiroyasu, T., Otsuka, T., Koibuchi, M.: Impact of topology and link aggregation on a PC cluster with ethernet. In: IEEE International Conference on Cluster Computing, pp. 280–285 (2008)
17. Wang, C., Yang, C., Liao, W., Chang, R., Wei, T.: Coupling GPU and MPTCP to improve hadoop/mapreduce performance. In: International Conference on Intelligent Green Building and Smart Grid (IGBSG), pp.1–6 (2016)