

Harnessing Cloud Computing to Power Up HPC Applications: The BRICS CloudHPC Project

Jonatas A. Marques[1], Zhongke Wu[2], Xingce Wang[2], Ruslan Kuchumov[3], Vladimir Korkhov[3,4], Weverton L. C. Cordeiro[1], Philippe O. A. Navaux[1], and Luciano P. Gaspary[1(✉)]

[1] Federal University of Rio Grande do Sul, Porto Alegre, Brazil
{jamarques,weverton,navaux,paschoal}@inf.ufrgs.br
[2] Beijing Normal University, Beijing, China
{wangxingce,zwu}@bnu.edu.cn
[3] Saint Petersburg State University, St. Petersburg, Russia
v.korkhov@spbu.ru
[4] Plekhanov Russian University of Economics, Moscow, Russia

Abstract. In this paper, we report the main accomplishments obtained in the context of the CloudHPC project. Accepted in response to the BRICS Pilot Call 2016, the project gathered researchers from the Federal University of Rio Grande do Sul (UFRGS), the St. Petersburg State University (SPbSU), and Beijing Normal University (BNU). Its main objective was to identify cloud computing aspects that can be explored to provide a far-reaching, flexible, and efficient environment for running HPC applications. In addition to the main results, we also discuss the lessons learned and perspectives for future research in the area.

Keywords: HPC applications · Cloud · Virtualization · Containers · Network · Medical applications

1 Introduction

Handling massive amounts of data is commonplace for most modern scientific, engineering, and business applications. As these applications need to target several big data-related challenges while delivering expected results in a timely manner, they frequently pose large computing power requirements. In this context, High-Performance Computing (HPC) becomes a key factor for speeding up data processing while also enabling faster time to market, lower capital expenditures, and higher valued innovation. To this end, HPC solutions have traditionally taken advantage of cluster and datacenter infrastructures for running applications having those computing power requirements [3]. In addition, practitioners have also been leveraging cloud computing resources for meeting HPC demands when available resources do not suffice. In fact, the pay-per-use cost model and resource elasticity make cloud computing an exciting environment for HPC, which can be provided with instant availability and flexible scaling of resources, among other benefits.

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12956, pp. 336–349, 2021.
https://doi.org/10.1007/978-3-030-87010-2_24

Despite the benefits of using cloud computing for HPC, a current approach has been the allocation of physical infrastructures in a dedicated mode for fast HPC provisioning. Although convenient, it frequently leads to underutilized resources, e.g., an application may not fully utilize provided CPU and/or network resources. It also prevents dealing adequately with those applications whose resource demands grow beyond available capacity. Traditional virtualization technologies can help to solve the problem. Still, the overhead of both bootstrapping a virtual infrastructure for each application and sharing physical resources among several virtual instances might be significant. Boosting available physical resources by using cloud computing, in turn, has been hampered because of limited support for shifting HPC applications to the cloud. These issues hinder the wide adoption of cloud computing by the HPC community, thus becoming paramount to understand how one can perform smooth and effective migration of (parts of) HPC applications to the cloud.

In the BRICS CloudHPC project, our primary goal was to identify cloud computing aspects that can be explored to provide a far-reaching, flexible (e.g., in terms of adaptation to fluctuating demands) and efficient environment for running HPC applications. In this paper, we report the main accomplishments during this three-year-long effort. We focus on (*i*) models, algorithms, and mechanisms to help users access a suitable cloud execution platform for running their HPC applications and (*ii*) science-related applications that can capitalize on these software artifacts to accelerate their execution.

The remainder of the paper is organized as follows. In Sect. 2, we characterize how existing public clouds perform when used to execute HPC applications. In Sect. 3, we summarize novel computing and networking techniques for HPC efficiency in cloud environments proposed in the context of the project. In Sect. 4, we describe a medical image processing use case used in CloudHPC. Research challenges and opportunities are then briefly discussed in Sect. 5. Finally, in Sect. 6 we close the paper with final remarks.

2 Tuning Public Cloud Instance Allocation for the Execution of HPC Applications

As mentioned earlier, executing HPC applications in cloud environments has become an important research topic in recent years. In contrast to traditional clusters, which consist of primarily homogeneous machines, running HPC applications in the cloud allows them to use heterogeneous resources offered by the cloud provider. Cloud instances can be adapted to match the imbalanced behavior of an HPC application, reducing the price of executing it in the cloud.

In the work [14], the authors performed a comprehensive set of experiments evaluating the potential of using heterogeneous instance types on three public cloud providers, Microsoft Azure, Amazon AWS, and Google Cloud. The difference in demands was explored by selecting multiple instance types adapted to the application's needs.

The experiments configured a cluster of 32 cores, consisting of four instances with eight cores each. All instances ran Ubuntu 18.04 LTS. To provide a fair comparison between the providers and their instances, the authors chose datacenter locations as close as possible. They used ImbBench [15] as the workload to perform the tests of the cloud configurations. ImbBench is a proto-benchmark system that implements several load patterns for MPI processes [13]. To measure the cost of the configuration, the authors used the Cost Efficiency methodology [13]. Briefly, this methodology helps the user determine which provider and configuration offer the higher performance for the price paid.

Microsoft Azure has the largest number of instances, with eight cores, among the three providers of this study. The authors selected ten instance types for the evaluation, covering the whole spectrum of available configurations. To evaluate the CPU performance of Microsoft Azure, they performed tests with different instance types. To identify if the combination of varying instance types could reduce the execution cost, they built several combinations of instances.

To reduce the overall idle time, the authors proposed the execution of the application using a combination of different instances. They could achieve this by using instances with less performance and price. Figure 1 shows the execution of the application in a heterogeneous cluster. In this case, they configured a cluster with one A8v2 instance, two F8 instances, and one H8 instance. The execution time remains the same as the homogeneous cluster. However, the idle time was reduced by using the cheaper instances.

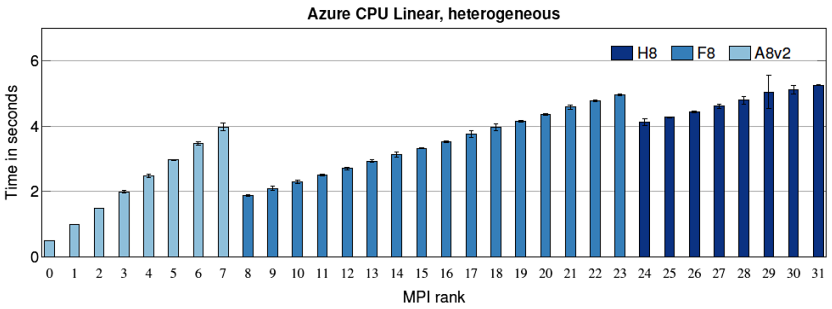

Fig. 1. CPU performance of the Azure heterogeneous cluster to run the Linear pattern.

The results showed that heterogeneous execution is most beneficial on Azure, improving cost efficiency by up to 50% compared to execution on homogeneous instances while maintaining the same performance. The other two providers are less suitable since the cheapest instance type either is the fastest as well (AWS) or the provider offers only instances that vary in memory size but not in performance (Google).

3 Novel Computing and Networking Techniques for HPC Efficiency in Cloud Environments

Our experience with cloud computing infrastructures showed that, although they allow for the execution of HPC applications, it is very hard (if possible) to achieve results comparable to those observed in dedicated infrastructures. Essential metrics such as execution time struggle to achieve acceptable levels. This is due to different issues such as the shared nature of cloud resources, the difficulty in enforcing VM allocation strategies that could potentially favor HPC, and well-known bottlenecks (e.g., virtualization and communication links).

In the project context, we focused on devising novel mechanisms to tailor the cloud towards a more convenient platform to execute HPC applications. Our work encompassed the proposal of solutions within two central axes, namely computing and networking. Next, we briefly review four primary outcomes of the CloudHPC project to make these components more efficient and, at the same time, resource usage judicious.

3.1 Collecting HPC Applications Processing Characteristics to Facilitate Co-scheduling

One of the goals of this project activity [6] was to reduce queue waiting times in cluster schedulers by applying co-scheduling using various strategies. This is especially important in virtualized and cloud environments that might share resources between multiple applications. Co-scheduling enables assigning various applications with different resource requirements to a single computing node with enough resources to run them simultaneously and without interference. For example, some applications that only require disk I/O may be running simultaneously with applications that only use the network intensively since they use entirely different resources. This strategy improves the use of cluster resources and reduces the waiting time for tasks in the queue compared to the commonly used scheduling strategies, where a node can run one application at a time. The scientific community is currently dedicating much attention to studies that assess the feasibility of this strategy and its practical aspects.

Since co-scheduling requires more information about the job resource requirements that the shared scheduler will ultimately use (e.g., the number of cores and the computation time), it is crucial to figure out these parameters and metrics of the job. They will be fed into a mathematical model that can then be used to schedule the solution.

Scheduling theory originated long before the field of high-performance computing. Its primary focus is on formalizing scheduling problems and applying various mathematical methods to create schedules that are subject to certain constraints and optimize a particular objective function. Nevertheless, when it comes to the classical theory, there are essential assumptions that may limit its applicability to the problem of joint planning. First, each job can be processed by no more than one machine, and each machine can process no more than

one job (operation). Second, the task execution time does not change in time. Third, the task completion time is known in advance. In the latest development of scheduling theory, at least one of these assumptions is updated.

In the co-scheduling problem, all the provisions of the classical schedule theory are dynamic, which makes the problem of determining the model quite challenging. Moreover, from a practical point of view, there are also a number of open questions. For example, sharing a resource with multiple applications without provoking any disagreement is not always possible. Additional information about the resource consumption of the task should be collected and taken into account when planning. What is even worse is that the required information is not easily accessible. The user cannot provide it at the stage of submitting the task as well as it is not easy to get it by tracking the behavior of the task during its execution. The scientific community has already done a great job collecting such information by interpreting available metrics, such as hardware counters or operating system events.

In [6], the authors considered ways to define cluster environments using scheduling theory models and described how job processing characteristics from the scheduling theory can be applied to describe HPC applications. They detailed how to track and monitor various resource requirements, such as the number of computing nodes, CPU cores, memory bandwidth, and network bandwidth while the application is running. Using this approach, it is possible to estimate the application execution time as a function of resource constraints. This also enables decisions about the limit of constraining individual applications when they are scheduled together with a co-scheduling strategy.

3.2 Resource Allocation for Running HPC Workloads Simultaneously

An effective resource management system is the key to high-quality solutions for HPC and distributed computing. One of the major problems is to map applications to available computing resources so that optimal application performance is delivered and resources are used efficiently. The traditional solution is to adapt applications to the resources, which is not always easy or effective. There is an alternative way that implies configuring the infrastructure to the application needs. This leads to the necessity of abstracting resources, which can be done with virtualization technologies.

In [4], the authors consider the execution of parallel and distributed applications in virtual clusters that are explicitly configured to match application resource requirements. The authors analyze experimental usage of configurable virtual clusters on different workloads in different resource sharing scenarios: parallel applications (NAS Parallel Benchmarks) and distributed data processing frameworks (Apache Hadoop benchmarks: MRBench, TestDFSIO, and TeraSort) and demonstrate that efficiency of using distributed resources can be increased – even in case of utilizing cloud resources – by the simultaneous execution of lightweight virtual clusters on a single set of resources.

In high-performance computing, schedulers are widely used to maintain queues of jobs created by various users and assign compute nodes to run these jobs. Usually, at any moment, one computational node is assigned to perform one task. Sometimes nodes are divided into slots (for example, one slot per processor core), and a slot is assigned to one job. The latter approach leads to better utilization of compute nodes and increased cluster throughput, as multiple jobs can be running simultaneously. Nevertheless, with this approach, some computational nodes or slots may not be used completely. For example, a slot might not be fully utilized when its assigned tasks require I/O or network. Unused slots cannot be scheduled. The scheduler cannot reclaim them, and the resources allocated to them become idle for some portion of the job's execution time. There may exist an overused slot right next to an underutilized one that can benefit from these unoccupied resources.

Wasted node or slot resources, in some cases, can be significant and expensive. For example, when running HPC in the public cloud, this will increase the cost of job computation as users will also pay for these idle resources. Thus, using a single-slot assignment strategy is not ideal. By assigning one compute node to perform several tasks, it would be possible to reduce the number of unoccupied resources compared to the above strategy with slots. In addition, more jobs could run concurrently as the node resources would be used more efficiently. All this would lead to a reduction in the waiting time for jobs in the queue and an increase in the throughput of the infrastructure.

To achieve the aforementioned goal, it would not be enough to carry out several tasks simultaneously since it is necessary to ensure an equitable distribution of resources between tasks. For example, one job can spawn more processes than another, so on average, it will get more CPU time because the operating system scheduler distributes it evenly. In [5], the authors propose resource allocation strategies that can be applied to HPC workloads to guarantee equal shares of each resource (e.g., CPU time, memory, and network bandwidth) for the jobs and also allow jobs to exceed their shares at the expense of underutilized shares of other jobs. By grouping resource consumers into separate control groups, it is possible to provide fair resource shares between groups regardless of the number of resource consumers in each group.

3.3 Minimizing Communication Overheads in Container-Based Clouds for HPC Applications

Although the cloud computing model has been well adopted in the industry, there are still relevant 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. Therefore, in another work carried out within the CloudHPC context [10], the authors proposed the aggregation of Network Interface Cards (NICs) in a ready-to-use integration with the OpenNebula cloud manager using Linux containers.

In the literature, we find that Vogel et al. [19] conducted a network performance evaluation using CloudStack IaaS manager, deploying KVM and

LXC-based clouds. They measured throughput and latency between the nodes with different configurations. The study of Wang et al. [20] proposed the applicability of the Multi-Path TCP (MPTCP) protocol for improving the performance of a distributed Hadoop/MapReduce architecture.

The motivation for using NIC aggregation is the need to achieve higher network performance without additional financial costs, ultimately leading to application execution speedups. The authors started the NIC aggregation configuration in the native environment by setting up the network bonding (Fig. 2). This configuration was done manually in Netplan (Network Configuration Abstraction Renderer) by selecting up to four interfaces to be slaves from `bond0` interface.

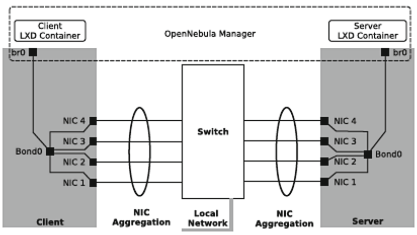

Fig. 2. High-level representation of the NIC aggregation environment.

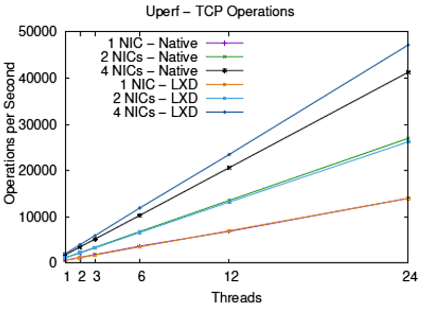

Fig. 3. TCP operations with NIC aggregation.

Figure 3 shows the Operations per second (Op/s) as provided by Uperf with the duration set to 90 s. Uperf uses threads in its evaluation, and the authors configured it to use up to 24 threads, twelve running on each machine. They used one NIC in each scenario to present a baseline where there is no aggregation. The linearity and difference between the scenarios are perceptible and highlight the gains of using link aggregation.

3.4 Boosting HPC Applications in the Cloud Through Dynamic Traffic-Aware Path Provisioning

As previously described, the pay-per-use cost model and resource allocation elasticity that cloud computing brings with it has the potential to improve HPC application performance while decreasing costs, as compared to clusters, grids, and data centers. Several studies [3,11,13] have sought to assess the feasibility of using cloud infrastructures to run HPC applications. The overall consensus is that clouds were not designed to run such applications. More specifically, clouds present poor network performance due to a combination of commodity interconnection technologies, processor sharing, and I/O virtualization overheads.

Additional studies [2,8] have identified the main reason for poor network performance as being the use of simplistic routing schemes (e.g., shortest path), which can lead to degradation in communication performance between compute nodes. In that context, they proposed routing approaches that avoid congested paths. Nevertheless, those approaches compute paths before the execution of an application, limiting their capability to handle fluctuations in network traffic and changes in the communication patterns of applications.

In our work [12], we proposed a novel mechanism for dynamic, just-in-time path provisioning in cloud infrastructures. Our mechanism takes advantage of new advances in programmable networks to dynamically compute and program the best paths between HPC application processes according to their current communication needs. Furthermore, we designed an algorithmic Link Usage-Aware Routing (LUAR) strategy to minimize end-to-end communication delays. Figure 4 illustrates an overview of our proposed mechanism, presenting its components and their interactions. Within the mechanism, LUAR represents a possible instantiation of the Forwarding Strategist component.

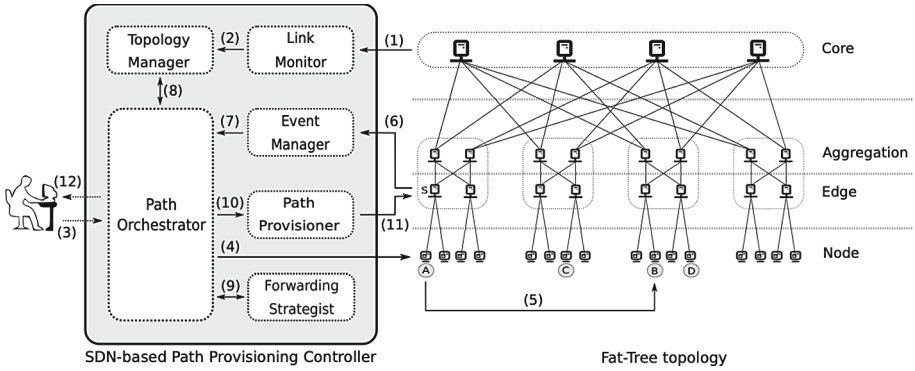

Fig. 4. Overview of the proposed path provisioning approach.

The experiments carried out considered a 32-host fat-tree topology emulated cloud infrastructure based on Mininet [7], Open vSwitch 2.0.2, and the Ryu SDN controller [16]. To generate the workload, the authors employed the NAS Parallel Benchmarks (NPB) version 3.3 [1]. Six representative benchmarks were selected for the evaluation: BT, EP, CG, LU, MG, and SP. Additional traffic was generated to simulate competing background traffic (at about 75% concurrency) in the infrastructure. Three different approaches to traffic routing were considered (*i*) Link Usage-Aware Routing (LUAR), (*ii*) Shortest path, and (*iii*) Random. The comparison metric was application runtime. Figure 5 shows the main results of this experiment.

The results show that LUAR outperforms Shortest Path by 31.35%, on average, among applications. EP, the shortest application, has obtained an improvement to its runtime of about 8% with LUAR as compared to Shortest Path. In turn, SP, the largest application, experimented a gain of 44.24% with LUAR.

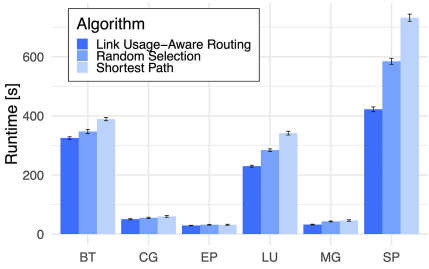

Fig. 5. Performance observed for the different NPB benchmark applications.

4 HPC Application Use Case: Medical Image Processing

In the context of the CloudHPC project, we used as use cases two medical imaging processing applications, namely registration with 3D medical models and efficient algorithms for cerebrovascular segmentation [9]. Next, we summarize the work on cerebrovascular segmentation.

A complete and detailed cerebrovascular image segmented from time-of-flight magnetic resonance angiography (TOF-MRA) data is essential for the diagnosis and therapy of cerebrovascular diseases. In recent years, three-dimensional cerebrovascular segmentation algorithms based on statistical models have been widely used. However, the existing methods always perform poorly on stenotic vessels and are not robust enough. Hence, a parallel cerebrovascular segmentation algorithm was proposed [9] based on the Focused Multi-Gaussians (FMG) model and heterogeneous Markov Random Field (MRF). Specifically, the authors proposed an FMG model with a local fitting region to model the vascular tissue more accurately and introduce the Chaotic Oscillation Particle Swarm Optimization (CO-PSO) algorithm to improve the global optimization capability in parameter estimation. Moreover, they designed a heterogeneous MRF in the three-dimensional neighborhood system to incorporate the precise local character of an image. Finally, the algorithm has been modeled to be parallelizable, as illustrated in Figs. 6, 7, and 8.

Experiments have been carried out using a test dataset to show the advantages of the proposed statistical cerebrovascular segmentation algorithm. The dataset of 3D TOF-MRA consisted of both the public CASILab dataset [17] and the dataset provided by the Peking Union Medical College Hospital. The MRA images consisted of $448 \cdot 448 \cdot 128$ axial slices with a resolution of $0.5 \cdot 0.5 \cdot 0.8\,\mathrm{mm}^3$. All tests were executed on the same Tianhe2A-2A supercomputer. Their execution on GPUs resulted in an approximate 60-time speed up (compared to a serial setup). The experiments showed that the proposed algorithm produces more detailed segmentation results (and quickly) and performs robustly on stenotic vessel segmentation (Table 1).

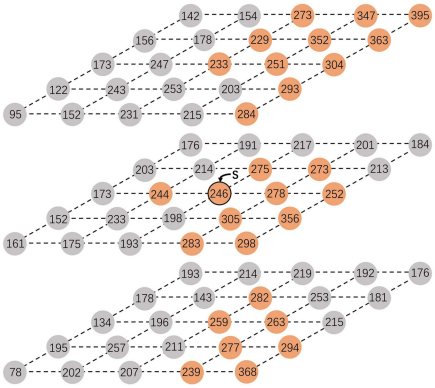

Fig. 6. Three-dimensional neighborhood system with a cuboid of size $5 \times 5 \times 3$ in the proposed heterogeneous MRF model.

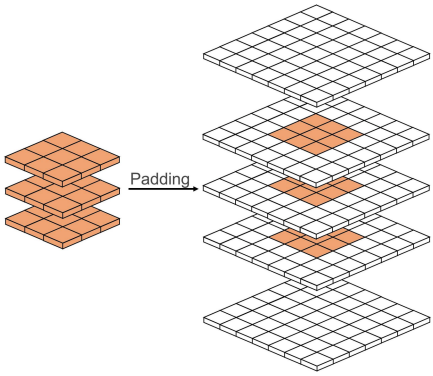

Fig. 7. Padding preprocesses the TOF-MRA data. It extends the dimension of the raw data from $m \cdot n \cdot k$ to $(m+4) \cdot (n+4) \cdot (k+2)$ and sets the intensity value of the new extended voxels to zero.

Table 1. Results of the cerebrovascular parallel segmentation method on Tianhe.

CO-PSO + ICM	CPU	GPU (V100)	Speed-up ratio
200 + 10	493.782 s	8.271 s	59.711
200 + 20	967.123 s	15.358 s	62.975

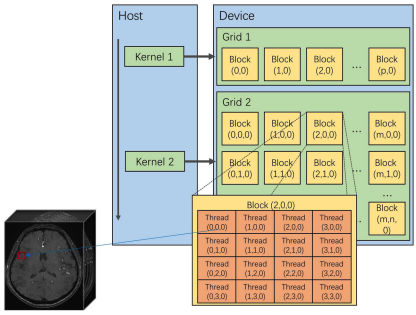

Fig. 8. Parallel programming model for the CO-PSO and ICM algorithms on CUDA.

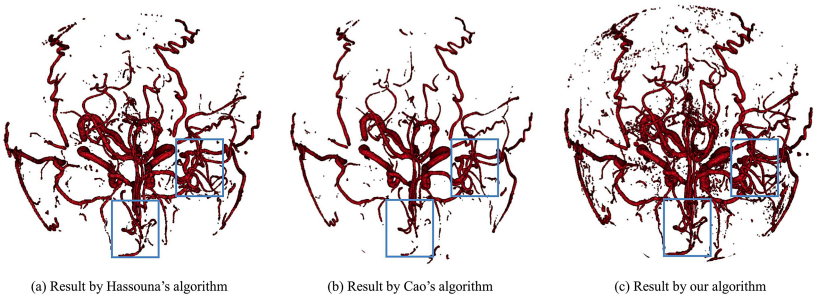

Fig. 9. Segmentation results by different statistical cerebrovascular segmentation algorithms. The segmentation result obtained by our algorithm shows more detailed vessels with stronger continuity than other models, especially in the colored areas.

5 Research Challenges and Opportunities

The work carried out demonstrated promising results and, more importantly, allowed us to reason about how to promote consistent advances in the area. Next, we elaborate on a few research challenges and opportunities.

Dealing with Varying Bottlenecks. HPC applications can be very different from one another regarding required resources. On the other extreme, they are time-stringent, requiring low execution times. The existing "knobs" for calibrating a cloud infrastructure are insufficient for HPC. While we proposed mechanisms for making the cloud less susceptible to well-known bottlenecks, we advocate more granularity is needed, e.g., from host/virtual machine to disaggregated resource allocation, on a per-job demand. We refer the reader to [18] for an insightful presentation corroborating this vision.

Getting Rid of Virtualization Layers. Existing cloud infrastructures resort to several software layers, e.g., especially to virtualize its resources. While traditional virtualization (i.e., hypervisor-based) provides flexibility in building a variety of environments, e.g., allowing to simulate and run completely different architectures and operating systems on top of each other, its overheads are high. Our work on this topic (e.g., [5], [6], and [10]) points towards the necessity to make the cloud more lightweight, for example, exploring techniques such as container-based or operating-system-level virtualization.

Modeling HPC Applications for the Cloud. Modeling applications for the cloud is paramount for fully harnessing the power of cloud computing to boost HPC applications. Design and algorithmic choices and abstractions that used to work well when executed on dedicated setups might underperform in shared infrastructures (e.g., due to the reasons outlined in Sects. 3.1, 3.2, 3.3, and 3.4). We advocate the need for new guidelines based on characterization studies and real experiences that can help application developers make more informed and correct design choices.

Operating Simultaneously with Private Cluster and Public Cloud. HPC solutions have traditionally taken advantage of powerful cluster and datacenter infrastructures. Given the increasingly complex and resource-demanding applications, we observe practitioners leveraging cloud computing resources for meeting HPC demands when available resources do not suffice. A significant challenge that arises in these scenarios is how to deploy applications properly in such hybrid setups. More research is needed to harmonize such distinct infrastructures "to get the best of both worlds" concerning *transparent* computing performance, scalability, and resilience (to mention a few non-functional requirements).

Hiding Complexity from the End-User. All efforts to enable the migration of HPC applications to the cloud result in more complexity to the end-user. The complexity reveals itself in multiple stages of the process of executing an application, ranging from the need to properly model the application for it to benefit, performance-wise, from the cloud all the way to the infrastructure provisioning and deployment phases. Hence, last but not least important, the advocated advances need to be accompanied by strategies that hide the complexity from the end-user. It is essential to work on adequate abstractions and interfaces that the user can interact with; otherwise, this big push towards the cloud might not be sustainable.

6 Conclusion

In this paper, we reported the main accomplishments obtained in the context of the CloudHPC project. To accomplish the project's objectives, researchers at UFRGS, SPbSU, and BNU worked synergistically towards an integrated, coherent result, taking advantage of each group's expertise. Solid contributions were made to understand the possibility of using existing public clouds as platforms for the execution of HPC applications (Sect. 2). Moreover, we proposed novel computing and networking techniques for HPC efficiency in cloud environments (Sect. 3). Finally, we worked on application modeling, focusing on medical image processing use cases (Sect. 4).

Despite the progress made, several challenges and opportunities remain, as briefly summarized in Sect. 5. It is our belief that solid advances in the area do require multi-party collaborations such as those promoted by the BRICS Calls. They allow researchers who work under quite different realities and have complementary expertise to build robust "bridges" and solutions to overcome big societal problems of our time, where HPC and cloud will together play a vital role.

Acknowledgements. This work was carried out in the context of the project CloudHPC – Harnessing Cloud Computing to Power Up HPC Applications, BRICS Pilot Call 2016. It was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq), Project Number 441892/2016-7, Call CNPq/MCTIC/BRICS-STI No 18/2016, the Coordination for the Improvement of Higher Education Personnel (CAPES), as well as the National Key Cooperation

between the BRICS Program of China (No. 2017YE0100500) and the Beijing Natural Science Foundation of China (No. 4172033).

References

1. Bailey, D.H.: NAS parallel benchmarks. In: Encyclopedia of Parallel Computing, pp. 1254–1259 (2011). https://doi.org/10.1007/978-0-387-09766-4_133
2. Faizian, P., Mollah, M.A., Tong, Z., Yuan, X., Lang, M.: A comparative study of SDN and adaptive routing on dragonfly networks. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, p. 51. ACM (2017)
3. Gupta, A., et al.: Evaluating and improving the performance and scheduling of HPC applications in cloud. IEEE Trans. Cloud Comput. **4**(3), 307–321 (2016). https://doi.org/10.1109/TCC.2014.2339858
4. Korkhov, V., et al.: Distributed virtual cluster management system. In: CEUR Workshop Proceedings, Selected Papers of the 8th International Conference on Distributed Computing and Grid-Technologies in Science and Education - GRID 2018, Cham, vol. 2267, pp. 383–387 (2018)
5. Kuchumov, R., Korkhov, V.: Fair resource allocation for running HPC workloads simultaneously. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11622, pp. 740–751. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24305-0_55
6. Kuchumov, R., Korkhov, V.: Collecting HPC applications processing characteristics to facilitate co-scheduling. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12254, pp. 168–182. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58817-5_14
7. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, p. 19. ACM (2010)
8. Lee, J., Tong, Z., Achalkar, K., Yuan, X., Lang, M.: Enhancing InfiniBand with OpenFlow-style SDN capability. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2016, pp. 36:1–36:12. IEEE Press, Piscataway (2016). http://dl.acm.org/citation.cfm?id=3014904.3014953
9. Lv, Z., et al.: Cerebrovascular segmentation algorithm based on focused multi-Gaussians model and weighted 3D Markov random field. In: 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 846–851 (2019). https://doi.org/10.1109/BIBM47256.2019.8983106
10. Maliszewski, A.M., Vogel, A., Griebler, D., Roloff, E., Fernandes, L.G., Philippe O. A.N.: Minimizing communication overheads in container-based clouds for HPC applications. In: 2019 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6 (2019). https://doi.org/10.1109/ISCC47284.2019.8969716
11. Netto, M.A.S., Calheiros, R.N., Rodrigues, E.R., Cunha, R.L.F., Buyya, R.: HPC cloud for scientific and business applications: taxonomy, vision, and research challenges. ACM Comput. Surv. **51**(1), 81–829 (2018). https://doi.org/10.1145/3150224
12. 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

13. Roloff, E., Diener, M., Diaz Carreño, E., Gaspary, L.P., Navaux, P.O.A.: Leveraging cloud heterogeneity for cost-efficient execution of parallel applications. In: Rivera, F.F., Pena, T.F., Cabaleiro, J.C. (eds.) Euro-Par 2017. LNCS, vol. 10417, pp. 399–411. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64203-1_29
14. Roloff., E., Diener., M., Gaspary., L., Navaux., P.: Exploring instance heterogeneity in public cloud providers for HPC applications. In: Proceedings of the 9th International Conference on Cloud Computing and Services Science - CLOSER, pp. 210–222. INSTICC, SciTePress (2019). https://doi.org/10.5220/0007799302100222
15. Roloff, E., Diener, M., Gaspary, L.P., Navaux, P.O.: Exploiting load imbalance patterns for heterogeneous cloud computing platforms. In: CLOSER, pp. 248–259 (2018)
16. Ryu, A.: Component-based software defined networking framework (2021). https://ryu-sdn.org
17. TubeTK: TubeTK/Data - KitwarePublic (2021). http://public.kitware.com/Wiki/TubeTK/Data
18. Vahdat, A.: Coming of age in the fifth epoch of distributed computing: the power of sustained exponential growth (2020). https://tinyurl.com/8frtbpfs
19. Vogel, A., Griebler, D., Schepke, C., Fernandes, L.G.: An intra-cloud networking performance evaluation on cloudstack environment. In: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 468–472 (2017). https://doi.org/10.1109/PDP.2017.40
20. Wang, C.H., Yang, C.K., Liao, W.C., Chang, R.I., Wei, T.T.: Coupling GPU and MPTCP to improve Hadoop/MapReduce performance. In: 2016 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG), pp. 1–6 (2016). https://doi.org/10.1109/IGBSG.2016.7539430