RC25685 (WAT1909-033) September 25, 2019
Computer Science

IBM Research Report

Proceedings of the 13th Symposium and Summer School On Service-Oriented Computing (SummerSoc19)

Johanna Barzen[1], Rania Y. Khalaf[2], Frank Leymann[1], Bernhard Mitschang[1]
Editors

[1]University of Stuttgart
Universitätsstraße 38
70569 Stuttgart
Deutschland

[2]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Research Division
Almaden – Austin – Beijing – Cambridge – Dublin - Haifa – India – Melbourne - T.J. Watson – Tokyo - Zurich

The 13th Advanced Summer School on Service-Oriented Computing

June 17 - June 23
2019
Hersonissos, Crete, Greece

The 13th advanced Summer School on Service-Oriented Computing (SummerSOC'19) continued a successful series of summer schools that started in 2007, regularly attracting world-class experts in Service-Oriented Computing to present state-of-the-art research during a week-long program organized in several thematic tracks: IoT, formal methods for SOC, Cloud Computing, Data Science, Advanced Manufacturing, Software Architecture, Digital Humanities, Quantum Computing, and emerging topics. The advanced summer school is regularly attended by top researchers from academia and industry as well as by PhD and graduate students.

During the different sessions at SummerSOC renowned researchers gave invited tutorials on subjects from the themes mentioned above. The afternoon sessions were also dedicated to original research contributions in these areas: these contributions have been submitted in advance as papers that had been peer-reviewed. Accepted papers were presented during SummerSOC and during the poster session. Furthermore, PhD students had been invited based on prior submitted and reviewed extended abstracts to present the progress on their theses and to discuss it during poster sessions. Some of these posters have been invited to be extended as a full paper, which are included in this technical report.

Johanna Barzen, Rania Khalaf, Frank Leymann, Bernhard Mitschang
- Editors -

Content

Poster Session: Extended Abstract

A Survey on Cloud Migration Strategies for High Performance Computing

Stefan Kehrer and Wolfgang Blochinger

Parallel and Distributed Computing Group, Reutlingen University, Alteburgstr. 150, 72762 Reutlingen, Germany
[firstname.lastname]@reutlingen-university.de

Abstract. The cloud evolved into an attractive execution environment for parallel applications from the High Performance Computing (HPC) domain. Existing research recognized that parallel applications require architectural refactoring to benefit from cloud-specific properties (most importantly elasticity). However, architectural refactoring comes with many challenges and cannot be applied to all applications due to fundamental performance issues. Thus, during the last years, different cloud migration strategies have been considered for different classes of parallel applications. In this paper, we provide a survey on HPC cloud migration research. We investigate on the approaches applied and the parallel applications considered. Based on our findings, we identify and describe three cloud migration strategies.

Keywords: Cloud Computing · High Performance Computing · Parallel Application · Cloud Migration · Cloud-aware Refactoring ·

1 Introduction

Traditionally, many parallel applications have been designed and developed for HPC clusters. However, more recently, the cloud evolved into an attractive execution environment for HPC workloads [26, 38]. Former research on this topic mainly investigates how to make cloud environments HPC-aware [25]. In particular, resource pooling and virtualization leading to heterogeneous processing speeds as well as low network throughput and high network latency have been addressed [8, 41]. During the last years great progress has been made with respect to HPC-aware cloud environments. As of today, many cloud providers, including Amazon Web Services (AWS)[1] and Microsoft Azure[2], offer cloud environments optimized for HPC workloads [44, 1].

On the other hand, there is a growing interest to make parallel applications cloud-aware [12, 13, 34, 30]. However, this requires architectural refactoring, which comes with many challenges and cannot be applied to all applications due to performance issues [19, 5]. As a result, different cloud migration strategies have been applied to different parallel applications.

[1] https://aws.amazon.com.
[2] https://azure.microsoft.com.

As more and more research considers the migration of parallel applications to the cloud, we argue that a survey on HPC cloud migration research is required to understand current research issues. In this paper, we investigate on HPC cloud migration and describe three cloud migration strategies identified in existing research. The remainder of this paper is structured as follows. In Section 2, we describe two different types of cloud environments that can be employed to operate parallel applications. In Section 3, we present the research method and search strategy underlying our survey. Based on existing research, we describe the HPC cloud migration strategies identified in Section 4. Moreover, we discuss the key findings of our survey. In Section 5, we conclude our work and describe future research opportunities.

2 Cloud Environments for HPC

Two different types of cloud environments can be employed to operate parallel applications: Standard and HPC-aware cloud environments. Standard cloud environments often suffer from CPU time sharing leading to heterogeneous processing speeds as well as low network throughput and high network latency, which are well-known effects of resource pooling and virtualization [8, 41]. On the other hand, HPC-aware cloud environments limit these negative side-effects by means of the following concepts:

- CPU affinity: HPC-aware cloud environments ensure CPU affinity both at the application and at the hypervisor level. As a result, vCPUs are mapped to physical CPU cores leading to improved cache locality and higher cache hit rates [13]. This concept is also referred to as CPU pinning.
- HPC-aware virtual machine (VM) placement policies: Standard cloud schedulers do not ensure co-location of VMs. Existing work has shown that HPC-aware VM placement effectively resolves this issue and leads to significant performance gains [14].
- Guaranteed network performance: HPC-aware cloud environments are based on novel concepts such as single root input/output virtualization (SR-IOV) to ensure guaranteed network bandwidth and latency with constant quality of service while supporting resource pooling and network virtualization [25].
- Disabled VM migration: Typically, VM migration is disabled to avoid environmental overhead.
- Disabled memory overcommitment: Memory overcommitment leads to preemption and paging and is typically disabled [7].
- Lightweight virtualization: Container-based virtualization (OS-level virtualization) ensures lower overheads compared to hypervisor-based virtualization [43].

3 Research Method and Search Strategy

Our survey is based on a literature review to identify existing research on HPC cloud migration. We followed the steps of a literature review process described

in [3]. For the search process, we employed the ACM Digital Library[3], IEEE Xplore[4], and Google Scholar[5]. The following search query was used for the search: *("cloud" OR "elastic" OR "elasticity") AND ("migrat*" OR "transform" OR "convert" OR "refactor*" OR "adapt*" OR "modification") AND ("hpc" OR "parallel application").*

We selected relevant articles based on a manual selection process: Only peer-reviewed articles (published in English) have been included. Moreover, only articles that discuss in detail how to migrate parallel applications to the cloud were selected. Additional literature has been identified (1) by reviewing the references of selected articles and (2) by analyzing the citations of these articles [40].

4 Cloud Migration Strategies for HPC

In this section, we review existing work on HPC cloud migration. This section is structured according to the three cloud migration strategies that we have identified: (1) *Copy & Paste*, (2) *Cloud-aware Refactoring*, and (3) *Cloud-aware Refactoring & Elasticity Control*. Table 1 summarizes our classification of existing work. For each cloud migration strategy, we describe the key findings in detail.

4.1 Copy & Paste

This migration strategy proposes the migration of existing parallel applications without modifications. Both standard and HPC-aware cloud environments have been evaluated by following this migration strategy. For instance, the authors of [6] evaluate an existing application based on the Message Passing Interface (MPI) [10] deployed to a standard cloud environment. Moreover, serial versions of the NAS Parallel Benchmarks (NPB) [2] have been employed to assess the computational performance of different instance types. The results obtained show that the lower performance measured is mainly related to the high network latencies and low network bandwidths in standard cloud environments. The authors of [33] also evaluate existing MPI-based applications from the NPB in standard cloud environments. On the other hand, the authors of [23] investigate MPI-based applications in HPC-aware cloud environments and measure low variance in network bandwidth. Moreover, the raw computation performance has been shown to be comparable to HPC clusters, even if virtualization overhead exists. The authors of [42] consider HPC-aware cloud environments more cost-effective compared to traditional HPC clusters if a cluster does not achieve high utilization. The authors of [12] evaluate different parallel applications based on MPI and CHARM++ deployed to both standard and HPC-aware cloud environments. Whereas specifically tightly-coupled applications suffer from the low network bandwidth and high network latency in standard cloud environments,

[3] https://dl.acm.org.
[4] https://ieeexplore.ieee.org.
[5] https://scholar.google.com.

Table 1. This table shows our classification of existing work on HPC cloud migration.

Selected Article	**Cloud Environment**	**Elasticity Control**	**Cloud Migration Strategy**
Evangelinos et al. [6]	Standard	None	*Copy & Paste*
Roloff et al. [33]	Standard	None	*Copy & Paste*
Marathe et al. [23]	HPC-aware	None	*Copy & Paste*
Zhai et al. [42]	HPC-aware	None	*Copy & Paste*
Gupta et al. [12]	Standard / HPC-aware	None	*Copy & Paste*
Rajan et al. [28]	HPC-aware	None	*Copy & Paste*
Gupta et al. [15]	Standard	None	*Cloud-aware Refactoring*
Rajan et al. [30]	Standard	None	*Cloud-aware Refactoring*
Vu et al. [39]	Standard	None	*Cloud-aware Refactoring*
Kehrer et al. [20]	Standard	None	*Cloud-aware Refactoring*
Da Rosa Righi et al. [34]	Standard	Reactive	*Cloud-aware Refactoring & Elasticity Control*
Da Rosa Righi et al. [35]	Standard	Reactive	*Cloud-aware Refactoring & Elasticity Control*
Rodrigues et al. [32]	Standard	Hybrid	*Cloud-aware Refactoring & Elasticity Control*
Da Rosa Righi et al. [36]	Standard	Hybrid	*Cloud-aware Refactoring & Elasticity Control*
Raveendran et al. [31]	Standard	Reactive	*Cloud-aware Refactoring & Elasticity Control*
Rajan et al. [28]	Standard	Reactive	*Cloud-aware Refactoring & Elasticity Control*
Haussmann et al. [16]	Standard	Reactive	*Cloud-aware Refactoring & Elasticity Control*

HPC-aware cloud offerings have been shown to effectively overcome these issues. Thus, tightly-coupled applications benefit from on-demand access to compute resources and the freedom to select the number of processing units. Having the freedom to select the number of processing units per application run is new to HPC users as traditionally the number of processing units is limited by resource quotas or one tries to optimize the number of processing units to get a job scheduled faster (e.g., in HPC clusters with job schedulers).

Whereas most approaches require the manual selection of the number of processing units, recent work also shows how to automatically select the number of processing units in an HPC-aware cloud environment when the computational steps and communication patterns of the application can be captured in form of an *application model*. Based on the application model, one is able to calculate how the number of processing units effects execution time, speedup, efficiency, and monetary costs thus allowing versatile optimizations per application run. The authors of [28] specifically address applications based on the split-map-merge paradigm and consider the cost-time product as objective function to statically select the optimal number of processing units per application run. Therefore, the automated selection process considers (1) information on the input problem, (2) an application model built for split-map-merge applications, (3) a user-defined objective function (in this case the cost-time product as a function of the number of processing units), and (4) information on the execution environment (e.g., processing speed, network bandwidth) obtained by measuring sample workloads.

By following this approach, parallel applications benefit from on-demand access to compute resources and pay-per-use, which enables fine-grained cost control per application run. Because the number of processing units does not have to be adapted at runtime, an existing parallel application can be deployed to an HPC-aware cloud environment without modifications.

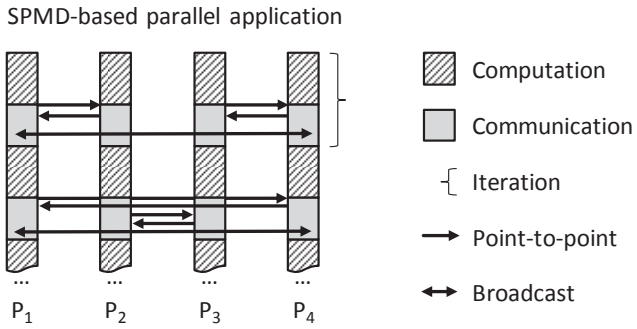

Fig. 1. Many parallel applications are developed based on the Single Program Multiple Data (SPMD) application model and rely on synchronous communication in globally defined communication phases.

Findings: In general, many existing parallel applications are implemented based on the Single Program Multiple Data (SPMD) application model [22, 24, 25] (especially supported by MPI) and rely on frequent synchronous communication. A prototypical SPMD-based MPI application is given in Fig. 1. In each iteration, every MPI process executes local computations. Thereafter, a pair of MPI processes exchanges application-specific data via point to point communication. After all process pairs finished their data transfers, updates required globally are sent to other processes by means of an MPI broadcast primitive (cf. Fig. 1). For tightly-coupled SPMD-based parallel applications, HPC-aware cloud offerings provide an execution environment that can be used analogously to an HPC cluster but allows individual configuration of compute resources (by means of virtualization techniques). In existing work, elasticity is not employed and thus the number of processing units has to be statically selected. However, by following the *Copy & Paste* migration strategy, parallel applications benefit from an on-demand provisioned execution environment that can be payed on a per-use basis and individual configuration of compute resources.

4.2 Cloud-aware Refactoring

This cloud migration strategy proposes architectural refactoring to make existing parallel applications cloud-aware [27, 19]. We discuss several examples for cloud-aware refactoring in the following.

The authors of [15] introduce a dynamic load balancing mechanism to address the challenges of tightly-coupled iterative MPI-based applications in standard cloud environments caused by virtualization and resource pooling. Therefore, a dynamic load balancer continuously monitors the load of each vCPU and reacts to a measured imbalance by adapting the task distribution to virtual machines. Task overdecomposition is used to enable dynamic load balancing, which effectively reduces idle time.

The authors of [30] employ the Work Queue framework [4] to develop parallel applications for standard cloud environments. The Work Queue framework is designed for scientific ensemble applications and provides a master/worker architecture with an elastic pool of workers. The application employed in the presented case study is designed for replica exchange molecular dynamics (REMD) and can be considered as iterative-parallel. Similar applications are discussed by the authors in [29].

The authors of [39] employ standard cloud environments to operate irregular task-parallel applications and present a work stealing algorithm that selects victims (other processing units) based on the measured network link latency. Processing units with a lower latency are preferred for stealing operations. The presented algorithm is self-adaptive and has been shown to outperform other load balancing mechanisms that do not consider network link latency.

The authors of [20] specifically address applications with dynamic task parallelism and discuss how these applications can benefit from elastic scaling. They argue that, in cloud environments, parallel applications have to dynamically adapt the degree of logical parallelism based on a dynamically changing physical

parallelism, given by the number of processing units (e.g., number of vCPUs, VMs). Based on their findings, the authors describe the design and implementation of a cloud-aware runtime system for elastic task-parallel processing in the cloud. The presented runtime system transparently controls the parallelism of an application to ensure elastic scaling. Therefore, developers mark parallelism in the program and the runtime system automatically adapts the logical parallelism by generating tasks whenever required. The runtime system exploits available processing units with maximum efficiency by mapping the logical parallelism (tasks) to the physical parallelism (processing units). An application based on this cloud-aware runtime system is elastically scalable because newly provisioned processing units (VMs) automatically receive tasks by means of load balancing and a task migration mechanism releases processing units that have been selected for decommissioning. To decouple task generation and task processing, the runtime system is based on the distributed task pool execution model and solves parallel coordination problems based on Apache ZooKeeper[6]. The authors state that the runtime system is not limited to any specific cloud management approach or tooling: Cloud management may comprise any kind of external decision making logic that finally adapts the number of processing units (i.e., the physical parallelism).

Findings: Cloud-aware refactoring has been specifically employed in the context of standard cloud environments. Typically, heterogeneous processing speeds and varying network latencies negatively effect parallel applications that employ synchronous communication and / or barrier synchronization. Cloud-aware refactoring can be employed to make these applications less affected by the characteristics of standard cloud environments. By following this migration strategy, one is able to exploit (low cost) standard cloud resources while still maximizing parallel performance. However, it has also been recognized that such an approach cannot be applied to all applications. Specifically, in the context of parallel applications with frequent communication and synchronization, architectural refactoring leads to fundamental performance issues and thus cannot be applied.

4.3 Cloud-aware Refactoring & Elasticity Control

This cloud migration strategy proposes the use of elasticity to process HPC workloads in the cloud. Therefore, architectural refactoring of parallel applications is fundamentally required to deal with a varying number of processing units. In the following, we discuss several examples for this migration strategy and describe the elasticity control mechanisms considered.

The authors of [34] describe a reactive elasticity control mechanism for iterative-parallel applications. The presented concept called AutoElastic supports the automated transformation (source-to-source translation) of existing MPMD[7]-based

[6] https://zookeeper.apache.org.
[7] Multiple Program Multiple Data.

MPI-2 applications with a master/worker architecture into elastic parallel applications. MPI-2 features dynamic process management and thus supports a varying number of MPI processes [11]. The presented concepts are evaluated by using a numerical integration application which simulates different dynamic workload patterns (e.g., ascending, descending, and wave workload). Similar concepts are discussed in [35].

The authors of [32] present a hybrid elasticity controller based on a technique called live thresholding, which has also been used in [36]. Live thresholding dynamically adapts the thresholds of a reactive elasticity controller, which is implemented as a closed feedback-loop architecture. Workload patterns are detected by comparing the last two average load values calculated based on monitored time series data and simple exponential smoothing. Similar concepts are discussed in [36]. Both approaches address iterative-parallel applications.

The authors of [31] propose a concept to transform MPI-based iterative-parallel applications into elastic applications. They describe how to adapt existing MPI-based applications to deal with a dynamically changing number of processing units. The presented approach basically terminates a running application and restarts the application with a different number of processing units. Termination can only be applied at certain points in the program, e.g., at the end of an iteration. The described elasticity controller is designed to optimize the desired execution time, which is estimated based on the number of iterations and the average iteration time. The underlying assumption is that the amount of work per iteration is constant. Scaling decisions are made by comparing the measured average iteration time with the required iteration time to complete within the user-defined execution time: If the average iteration time is below the required iteration time, processing units are added. Otherwise, processing units are removed.

The authors of [28] (we already discussed this work in Section 4.1) also describe a second approach to use their application model: Whenever the characteristics of the cloud environment (e.g., processing speed, network bandwidth) are expected to change at runtime, an elasticity controller monitors the environment and continuously evaluates the objective function based on monitoring data. When the optimal (with respect to the user-defined objective function) number of processing units changes, the elasticity controller dynamically adapts the resource configuration. A cloud-aware application architecture is required to support such adaptations at runtime.

The authors of [16] discuss elasticity-related opportunities and challenges for irregular task-parallel applications. Their computation and communication patterns are input-dependent, unstructured, and evolving during the computation and thus their runtime and scaling behavior cannot be determined upfront [9, 37]. As a result, the total number of essential basic computational steps per time unit is unknown a priori and cannot be predicted. These applications comprise an unpredictable workload pattern. The authors discuss the two conflicting objectives of fast processing and low monetary costs finally leading to a multi-objective optimization problem and Pareto optimal solutions, which prevents automated

decision making with respect to the number of processing units. To deal with this problem, the authors employ the concept of opportunity costs to convert the underlying objective functions into a single aggregated objective function, thus allowing cost-based selection of the number of processing units. Because one cannot reason about the effects on execution time, speedup, efficiency, and monetary costs in absolute terms, the authors present a reactive elasticity controller for heuristic cost optimization: The cost function is approximated based on metrics monitored at runtime. Therefore, the elasticity controller continuously monitors the application and evaluates the defined objective function (minimize the monetary costs based on the presented cost model). The authors empirically evaluate their elasticity controller by comparing the results of their heuristic cost optimization approach with the minimum monetary costs (which can be obtained by measuring the scalability of the application with exemplary input problems).

Findings: This migration strategy proposes the use of elasticity to deal with dynamic and unpredictable workload patterns and / or environmental changes in standard cloud environments. Either proactive, reactive, or hybrid elasticity control mechanisms can be employed (depending on the characteristics of the application). Additionally, a cloud-aware application architecture is fundamentally required to ensure that a parallel application dynamically adapts to a changing number of processing units (selected by an elasticity controller). By following this migration strategy, parallel applications benefit from elasticity in form of more efficiently employed compute resources.

4.4 Summary and Discussion

Specifically for the *Copy & Paste* migration strategy, the ongoing evolution of HPC-aware cloud environments provides attractive benefits when compared to traditional HPC cluster environments. Existing work following this migration strategy does not make use of elasticity. Because mainly tightly-coupled data-parallel applications have been considered, this can be explained by considerable repartitioning efforts (when processing units are added or removed). With the technology available today, it is not possible for tightly-coupled SPMD-based applications to make use of elasticity to optimize costs and efficiency by adding or removing processing units during the computation due to the high overheads that would result from repartitioning.

The *Cloud-aware Refactoring* migration strategy proposes architectural refactoring of existing parallel applications. Architectural refactoring, in general and specifically in the context of parallel applications, is a comparatively new concept. The authors of [45] applied architectural refactoring to develop cloud-native applications. In [19], we present an approach to assess the cloud readiness of parallel applications that can be used to gain insights into the architectural changes required. In this context, we also recognized that many parallel applications provide several degrees of freedom with respect to their architecture.

Finally, we identified a third migration strategy: *Cloud-aware Refactoring & Elasticity Control*. Existing work following this migration strategy focuses on the benefits that can be obtained by means of elasticity. Specifically, in [28] and

[16], monetary costs and time are explicitly considered. The different approaches described basically result from the different characteristics of the applications addressed: Whereas the scaling behavior of applications based on the split-map-merge paradigm can be predicted based on an application model [28], the scaling behavior of irregular task-parallel applications is unknown upfront and unpredictable by nature, which requires reactive elasticity control [16]. On the other hand, both approaches convert the underlying multi-objective optimization problem into a single-objective optimization problem to enable automated decision making with respect to the optimal number of processing units. The authors of [28] use the cost-time product to create a single-objective optimization problem. The authors of [16] employ the concept of opportunity costs, which can be used to express time in terms of costs, thus enabling a purely cost-driven optimization.

5 Conclusion

With the aim of providing contributions to practitioners and researchers alike, we present a classification of HPC cloud migration research and describe the key findings. Most importantly, we recognized that elasticity, which is often considered to be the fundamental property of cloud environments, can only be beneficially employed under certain circumstances. More research is required to understand elasticity-related opportunities and challenges in the context of HPC. Whereas HPC users traditionally had no visibility of the monetary costs of compute resources in cluster environments, the pay-per-use property requires users to consider costs in cloud environments. Related work shows how to exploit on-demand compute resources and elasticity to control the monetary costs of parallel computations in the cloud. These approaches dynamically adapt the number of processing units under consideration of scare resources such as time and money. However, as of today, a clear and generally applicable understanding of elasticity in the context of HPC does not exist.

Our long-term goal is to understand how to design, develop, and manage cloud-aware parallel applications, i.e., applications that leverage cloud-specific properties such as on-demand access to compute resources, pay-per-use, and elasticity. Therefore, we follow a multi-faceted approach by investigating design-level, programming-level, and system-level aspects [19, 20] as well as delivery and deployment automation [18, 17, 21].

Acknowledgements. This research was partially funded by the Ministry of Science of Baden-Württemberg, Germany, for the Doctoral Program *Services Computing*.

References

1. Aljamal, R., El-Mousa, A., Jubair, F.: A comparative review of high-performance computing major cloud service providers. In: 2018 9th International Conference on Information and Communication Systems (ICICS). pp. 181–186 (April 2018)

2. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., Simon, H.D., Venkatakrishnan, V., Weeratunga, S.K.: The nas parallel benchmarks summary and preliminary results. In: Supercomputing '91:Proceedings of the 1991 ACM/IEEE Conference on Supercomputing. pp. 158–165 (Nov 1991)
3. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software **80**(4), 571 – 583 (2007)
4. Bui, P., Rajan, D., Abdul-Wahid, B., Izaguirre, J., Thain, D.: Work queue+python: A framework for scalable scientific ensemble applications. In: Workshop on Python for High-Performance and Scientific Computing (2011)
5. Ekanayake, J., Fox, G.: High performance parallel computing with clouds and cloud technologies. In: Avresky, D.R., Diaz, M., Bode, A., Ciciani, B., Dekel, E. (eds.) Cloud Computing. pp. 20–38. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
6. Evangelinos, C., Hill, C.N.: Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2. In: In The 1st Workshop on Cloud Computing and its Applications (CCA (2008)
7. Ferreira, K.B., Bridges, P., Brightwell, R.: Characterizing application sensitivity to os interference using kernel-level noise injection. In: 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 1–12 (Nov 2008)
8. Galante, G., Erpen De Bona, L.C., Mury, A.R., Schulze, B., da Rosa Righi, R.: An analysis of public clouds elasticity in the execution of scientific applications: a survey. Journal of Grid Computing **14**(2), 193–216 (Jun 2016)
9. Gautier, T., Roch, J.L., Villard, G.: Regular versus irregular problems and algorithms. In: Ferreira, A., Rolim, J. (eds.) Parallel Algorithms for Irregularly Structured Problems. pp. 1–25. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
10. Gropp, W., Lusk, E., Skjellum, A.: Using MPI: portable parallel programming with the message-passing interface. MIT press, third edn. (2014)
11. Gropp, W., Thakur, R., Lusk, E.: Using MPI-2: Advanced features of the message passing interface. MIT press (1999)
12. Gupta, A., Faraboschi, P., Gioachin, F., Kale, L.V., Kaufmann, R., Lee, B., March, V., Milojicic, D., Suen, C.H.: Evaluating and improving the performance and scheduling of hpc applications in cloud. IEEE Transactions on Cloud Computing **4**(3), 307–321 (July 2016)
13. Gupta, A., Kale, L.V., Gioachin, F., March, V., Suen, C.H., Lee, B.S., Faraboschi, P., Kaufmann, R., Milojicic, D.: The who, what, why, and how of high performance computing in the cloud. In: IEEE 5th International Conference on Cloud Computing Technology and Science. vol. 1, pp. 306–314 (Dec 2013)
14. Gupta, A., Kal, L.V., Milojicic, D., Faraboschi, P., Balle, S.M.: Hpc-aware vm placement in infrastructure clouds. In: 2013 IEEE International Conference on Cloud Engineering (IC2E). pp. 11–20 (March 2013)
15. Gupta, A., Sarood, O., Kale, L.V., Milojicic, D.: Improving hpc application performance in cloud through dynamic load balancing. In: 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. pp. 402–409 (May 2013)
16. Haussmann, J., Blochinger, W., Kuechlin, W.: Cost-efficient parallel processing of irregularly structured problems in cloud computing environments. Cluster Computing (Dec 2018)

17. Kehrer, S., Blochinger, W.: Autogenic: Automated generation of self-configuring microservices. In: Proceedings of the 8th International Conference on Cloud Computing and Services Science. pp. 35–46. SciTePress (2018)
18. Kehrer, S., Blochinger, W.: Tosca-based container orchestration on mesos. Computer Science - Research and Development **33**(3), 305–316 (Aug 2018)
19. Kehrer, S., Blochinger, W.: Migrating parallel applications to the cloud: assessing cloud readiness based on parallel design decisions. SICS Software-Intensive Cyber-Physical Systems **34**(2), 73–84 (Jun 2019)
20. Kehrer, S., Blochinger, W.: Taskwork: A cloud-aware runtime system for elastic task-parallel hpc applications. In: Proceedings of the 9th International Conference on Cloud Computing and Services Science. pp. 198–209. SciTePress (2019)
21. Kehrer, S., Riebandt, F., Blochinger, W.: Container-based module isolation for cloud services. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). pp. 177–186 (2019)
22. Keutzer, K., Massingill, B.L., Mattson, T.G., Sanders, B.A.: A design pattern language for engineering (parallel) software: merging the plpp and opl projects. In: Proceedings of the 2010 Workshop on Parallel Programming Patterns. ACM (2010)
23. Marathe, A., Harris, R., Lowenthal, D.K., de Supinski, B.R., Rountree, B., Schulz, M., Yuan, X.: A comparative study of high-performance computing on the cloud. In: Proceedings of the 22Nd International Symposium on High-performance Parallel and Distributed Computing. pp. 239–250. HPDC '13, ACM, New York, NY, USA (2013)
24. Massingill, B.L., Mattson, T.G., Sanders, B.A.: Reengineering for parallelism: an entry point into plpp for legacy applications. Concurrency and Computation: Practice and Experience **19**(4), 503–529 (2007)
25. Mauch, V., Kunze, M., Hillenbrand, M.: High performance cloud computing. Future Generation Computer Systems **29**(6), 1408 – 1416 (2013)
26. 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 Computing Surveys (CSUR) **51**(1), 8:1–8:29 (Jan 2018)
27. Parashar, M., AbdelBaky, M., Rodero, I., Devarakonda, A.: Cloud paradigms and practices for computational and data-enabled science and engineering. Computing in Science Engineering **15**(4), 10–18 (July 2013)
28. Rajan, D., Thain, D.: Designing self-tuning split-map-merge applications for high cost-efficiency in the cloud. IEEE Transactions on Cloud Computing **5**(2), 303–316 (April 2017)
29. Rajan, D., Thrasher, A., Abdul-Wahid, B., Izaguirre, J.A., Emrich, S., Thain, D.: Case studies in designing elastic applications. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. pp. 466–473 (May 2013)
30. Rajan, D., Canino, A., Izaguirre, J.A., Thain, D.: Converting a high performance application to an elastic cloud application. In: IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom). pp. 383–390. IEEE (2011)
31. Raveendran, A., Bicer, T., Agrawal, G.: A framework for elastic execution of existing mpi programs. In: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. pp. 940–947 (May 2011)
32. Rodrigues, V.F., da Rosa Righi, R., da Costa, C.A., Singh, D., Munoz, V.M., Chang, V.: Towards combining reactive and proactive cloud elasticity on running

hpc applications. In: Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS. pp. 261–268. INSTICC, SciTePress (2018)
33. Roloff, E., Diener, M., Carissimi, A., Navaux, P.O.A.: High performance computing in the cloud: Deployment, performance and cost efficiency. In: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings. pp. 371–378 (Dec 2012)
34. da Rosa Righi, R., Rodrigues, V.F., da Costa, C.A., Galante, G., de Bona, L.C.E., Ferreto, T.: Autoelastic: Automatic resource elasticity for high performance applications in the cloud. IEEE Transactions on Cloud Computing **4**(1), 6–19 (Jan 2016)
35. da Rosa Righi, R., Rodrigues, V.F., da Costa, C.A., Kreutz, D., Heiss, H.U.: Towards cloud-based asynchronous elasticity for iterative HPC applications. Journal of Physics: Conference Series **649**, 012006 (oct 2015)
36. da Rosa Righi, R., Rodrigues, V.F., Rostirolla, G., da Costa, C.A., Roloff, E., Navaux, P.O.A.: A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications. Future Generation Computer Systems **78**, 176 – 190 (2018)
37. Sun, Y., Wang, C.L.: Solving irregularly structured problems based on distributed object model. Parallel Computing **29**(11-12), 1539–1562 (Nov 2003)
38. Vecchiola, C., Pandey, S., Buyya, R.: High-performance cloud computing: A view of scientific applications. In: 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN). pp. 4–16. IEEE (2009)
39. Vu, T.T., Derbel, B.: Link-heterogeneous work stealing. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. pp. 354–363 (May 2014)
40. Webster, J., Watson, R.T.: Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly **26**(2), xiii–xxiii (2002)
41. Yang, X., Wallom, D., Waddington, S., Wang, J., Shaon, A., Matthews, B., Wilson, M., Guo, Y., Guo, L., Blower, J.D., Vasilakos, A.V., Liu, K., Kershaw, P.: Cloud computing in e-science: research challenges and opportunities. The Journal of Supercomputing **70**(1), 408–464 (Oct 2014)
42. Zhai, Y., Liu, M., Zhai, J., Ma, X., Chen, W.: Cloud versus in-house cluster: Evaluating amazon cluster compute instances for running mpi applications. In: SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 1–10 (Nov 2011)
43. Zhang, J., Lu, X., Panda, D.K.: Performance characterization of hypervisor-and container-based virtualization for hpc on sr-iov enabled infiniband clusters. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 1777–1784 (May 2016)
44. Zhang, J., Lu, X., Panda, D.K.D.: Designing locality and numa aware mpi runtime for nested virtualization based hpc cloud with sr-iov enabled infiniband. In: Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. pp. 187–200. VEE '17, ACM, New York, NY, USA (2017)
45. Zimmermann, O.: Architectural refactoring for the cloud: a decision-centric view on cloud migration. Computing **99**(2), 129–145 (Feb 2017)