Leveraging Cloud Heterogeneity for Cost-Efficient Execution of Parallel Applications

Eduardo Roloff(✉), Matthias Diener, Emmanuell Diaz Carreño, Luciano Paschoal Gaspary, and Philippe O.A. Navaux

Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
{eroloff,mdiener,edcarreno,paschoal,navaux}@inf.ufrgs.br

Abstract. Public cloud providers offer a wide range of instance types, with different processing and interconnection speeds, as well as varying prices. Furthermore, the tasks of many parallel applications show different computational demands due to load imbalance. These differences can be exploited for improving the cost efficiency of parallel applications in many cloud environments by matching application requirements to instance types. In this paper, we introduce the concept of heterogeneous cloud systems consisting of different instance types to leverage the different computational demands of large parallel applications for improved cost efficiency. We present a mechanism that automatically suggests a suitable combination of instances based on a characterization of the application and the instance types. With such a heterogeneous cloud, we are able to improve cost efficiency significantly for a variety of MPI-based applications, while maintaining a similar performance.

Keywords: Cloud computing · Cost efficiency · Heterogeneity · Performance

1 Introduction

Executing large parallel applications in the cloud has reached the mainstream and has become a major research topic in recent years. Compared to clusters, the cloud provides a higher flexibility and lower up-front costs for the hardware [14]. Public cloud providers such as Amazon's EC2 and Microsoft's Azure provide a large number of cloud instance types with different numbers of cores, processing speeds, and network interconnections [18]. Research in this area focuses mostly on porting applications to the cloud [12], evaluating their performance and cost efficiency [13,18], and improving communication performance [2–4].

An aspect that has received less attention is building a multi-instance cloud system out of different instance types. We refer to such a system as a *heterogeneous cloud* in this paper. Most multi-instance clouds currently use the same instance types, or use different instance types only in the context of accelerators (such as GPUs) [5]. A heterogeneous cloud is an interesting solution for the execution of large parallel applications, as these applications typically have

© Springer International Publishing AG 2017
F.F. Rivera et al. (Eds.): Euro-Par 2017, LNCS 10417, pp. 399–411, 2017.
DOI: 10.1007/978-3-319-64203-1_29

heterogeneous computational demands, with some tasks performing more work than others. In such a scenario, tasks that perform more work can be executed on faster but more expensive instances, while tasks that perform less work can be executed on slower and more cost-efficient instances.

In this paper, we investigate heterogeneous clouds, focusing on their potential for cost-efficient execution of parallel applications. Our main contributions are the following:

- We perform an in-depth evaluation of heterogeneous clouds with a variety of instance combinations and parallel application behaviors.
- We present a mechanism for determining instance combinations that is based on application behavior and instance characteristics.

Our proposal is compatible with a wide range of applications and requires no changes to the applications or runtime environments. In an evaluation with ten MPI-based benchmarks and scientific applications on several types of Microsoft Azure instances, we show that our proposal results in drastic cost reductions of executing parallel applications in heterogeneous clouds, while maintaining a similar performance, which leads to substantial improvements in cost efficiency of up to 18% (6.6% on average).

2 Performance and Cost Differences in the Cloud

Most public cloud providers offer a wide variety of instance types with different characteristics and prices. This section provides an analysis of homogeneous cloud clusters, that is, clusters that are composed of cloud instances of the same type, in terms of their computational performance and cost. We also measure the load imbalance of a set of parallel applications. Combining these two aspects leads us to motivate the introduction of heterogeneous clouds.

2.1 Methodology of the Analysis

Experiments in this section were performed with the following methodology. First, we selected a group of homogeneous clouds to run performance and cost tests, focusing on instance types that are similar to provide a better comparison. Based on these results, we selected instance types that will be used for the rest of this work. Second, we verified the computational load profile of several parallel applications by measuring the number of instructions per task.

All experiments were performed on the Microsoft Azure public cloud, which was selected since it has the largest number of instance types among the main cloud providers. We selected the A10, D4 v2, F8, G3, and H8 instances of Azure to verify their efficiency in terms of performance and cost. All chosen instance types consist of eight cores, which is the most common instance size in Azure. The multi-instance experiments use eight nodes, for a total number of 64 cores in all cases. The software environment consists of Ubuntu server 16.04, with Linux kernel 4.4. We use Open MPI [7] 1.10.2 as the parallel runtime environment. All

Table 1. Characteristics of the Azure instance types. Instance types that are evaluated in depth in this paper are marked in **bold**.

Instance name	Price/hour	Linpack perf. (GFlops)	Price/TFlop
A10	US$ 0.780	155.35	US$ 5.02
D4 v2	US$ 0.559	265.00	US$ 2.11
F8	US$ 0.513	246.10	US$ 2.08
G3	US$ 2.440	280.17	US$ 8.71
H8	US$ 0.971	324.34	US$ 2.99

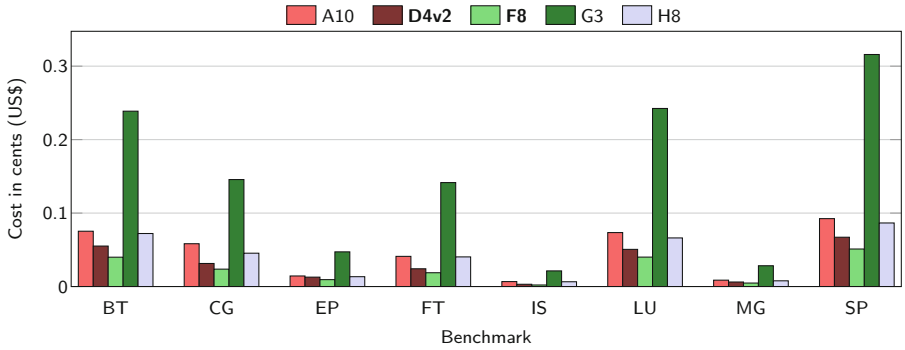

Fig. 1. Cost per execution (in US$ cents) of each NAS benchmark on homogeneous eight-instance cloud systems.

applications were compiled with gcc/gfortran 5.4.0, using the `-O2` optimization level. We measured the raw computational performance of a single instance of each type with the High Performance Linpack (HPL) benchmark [11]. To evaluate the cost of the machines we calculate the price of a TFlop, based on performance results and the price for each instance, with the Linpack performance. Table 1 presents an overview of the characteristics of these instance types as well as the performance and cost results.

Experiments were performed with a variety of MPI-based parallel applications. We use the MPI implementation of the NAS Parallel Benchmarks (NPB) [1], version 3.3.1. Experiments were performed with input class *C*, which represents a medium-large input size. The *DT* application was not used because it needs at least 85 MPI processes to execute using input size *C*. *BRAMS* (Brazilian developments on the Regional Atmospheric Modeling System) [6] is the extended version of the RAMS (Regional Atmospheric Modeling System) weather prediction model. *Alya* [9] is a simulation code for multi-physics problems, based on a variational multi-scale finite element method for unstructured meshes.

The location used to allocate the machines on Microsoft Azure was "West USA". All experiments were executed 10 times. Applications were configured to run with 64 ranks (1 rank per core). In our experiments, we did not notice

significant differences between executions at different times of day and between different allocations of instances.

2.2 Cost of Homogeneous Clouds

This section presents the cost results of executing the NAS benchmarks on homogeneous cloud instances. The instance types selected for the rest of this work are also discussed.

The cost per execution of the NAS benchmarks are shown in Fig. 1. The cost was calculated by multiplying the price per second of each instance with the execution time of the benchmark. The G3 instance presented the highest cost of all benchmarks. Despite its high performance, G3 has a cost that is several times bigger than the other instances. The other instance types show a similar behavior among them. The D4 v2 and F8 instances presented the lowest cost among all the instances tested, resembling thus the cost analysis of Linpack.

Based on these preliminary results, we selected two instance sizes, D4 v2 and F8, for our analysis in this paper. They were chosen because they have the best relation of cost and performance among all the types we evaluated. Furthermore, despite having differences in the price per hour and performance, their relation between cost and performance is very similar and therefore provides an interesting tradeoff between price and speed. For simplicity, the D4 v2 instance type will be referred to as D4 in the rest of the paper.

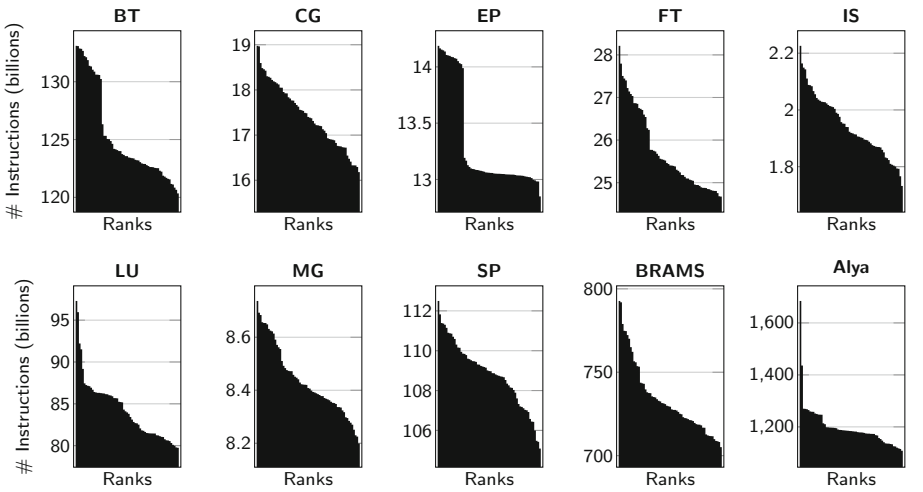

Fig. 2. Load distribution of the benchmarks, running with 64 ranks. Each bar corresponds to the number of instructions executed by a rank. Ranks are sorted according the numbers of instructions executed.

2.3 Load Imbalance

Another important aspect of our proposal is that parallel applications have different computational demands. Such a load imbalance can be caused for various reasons, such as an imperfect distribution of work, interference from other applications or users, or if an algorithm has different complexities for different regions of the input data. To evaluate the load imbalance of the parallel applications, we used the `perf` tool [10], measuring the number of instructions executed by each MPI rank. All applications were executed with 64 ranks.

Results of this experiment are shown in Fig. 2. In the figure, we show the numbers of instructions executed by each rank for each benchmark, sorted in descending order. The results show varying degrees of imbalance between the applications. In general, imbalance is considerable, with differences between the minimum and maximum numbers of instructions for each benchmark reaching up to 35% in the case of *Alya*. Some applications, such as *FT*, *SP*, and *Alya*, have considerable sequential parts that increase their imbalance. Others, such as *BT*, *EP*, and *FT*, show two distinct levels of numbers of instructions executed by the ranks. Executing the applications multiple times results in a very similar load profile, with similar loads for each rank.

3 A Mechanism to Improve Cost Efficiency in the Cloud

This section describes our proposed mechanism to automatically leverage the heterogeneity for an improved cost efficiency. The mechanism calculates for a given profile of an application and cloud instance types how many instances of each type should be used, and which MPI ranks should be executed on each instance. An overview of our proposal is shown in Algorithm 1.

Our mechanism receives as input the load profile of the application, the instance profile of the possible instance types. Currently, two different instance types are taken into account, which we refer to as HI (higher performance and cost) and LO (lower performance and cost), which correspond to the D4 and F8 instances in our experiments, respectively. We focus on two instance types since many of the applications show a two-level load distribution. The mechanism outputs the instance combination, that is, how many instances of each type should be used, and which ranks should be placed on each instance.

3.1 Mechanism Inputs

Our mechanism requires the following inputs. The first two, application profile and instance profile, are generated automatically.

Application Profile. The load profile of the application is generated via the `perf` tool [10]. We focus on the number of instructions per rank, as in Sect. 2. This could be extended to a more fine-grained differentiation for several types of instructions (for example, floating point or integer operations) and other types

Algorithm 1. Algorithm of our proposed mechanism

```
Data: Application profile, Instances profiles
Result: MPI rankfile (mapping of MPI ranks to instances)
calculate instance performance ratio;
while ranksize/VMs count < rankgroup do
    calculate ratio (rankgroup[n], rankgroup[n+1]);
    if rankgroup ratio < instance performance ratio then
        assign rankgroup to a HI VM;
        fetch next rankgroup;
    else
        assign rest of rankgroups to LO VMs;
    end
end
output rankfile;
```

of resources (such as communication, memory, I/O). This application profile usually needs to be generated only once for a given set of input data and number of ranks. The result of this stage is the load vector of the application.

Instance Profile. The mechanism generates the instance profile of the instance types by measuring the execution time of the desired application on homogeneous instances. The result of this stage is the relative performance between the two instance types.

3.2 Mechanism Outputs

Our mechanism outputs the instance combination, as well as which ranks should run on which instance. For the discussion in this section, we first sort the load vector in descending order, as shown in the load distributions in Fig. 2.

Instance Combination. The first output step of our mechanism is to determine the instance combination. We group the sorted list of ranks into groups of the same size as the number of cores per instance and calculate the cumulative load of each group. The mechanism then iterates over the list of groups and calculates the load ratio between subsequent groups. Until the ratio reaches a threshold, groups will be executed on HI instances. When the ratio is above the threshold, all subsequent groups will be executed on LO instances. The threshold is determined by the performance ratio of the instance types.

Rank Placement. After determining the number of HI and LO instance types, our mechanism assigns ranks to the instances in the following way. It iterates over the sorted list of ranks, and assigns ranks to instances sequentially, starting with the ranks with the highest load, which are assigned to the HI instances.

As soon as one instance has the maximum number of ranks assigned to it, the mechanism continues the rank placement with the next instance, until all ranks are assigned. In the final step, our mechanism creates a rank file that specifies the task to instance assignment for Open MPI [15].

4 Results

This section presents the evaluation methodology used to validate our proposal, as well as the obtained results.

For our experiments, we use the same Azure instance types as before, D4 and F8. All experiments use eight instances for a total of 64 cores. We vary the mix of instances between:

- fully homogeneous: all eight instances are of the same type, D4 or F8.
- heterogeneous to varying degrees: 1–7 instances of each type, totaling eight instances.

We evaluate all possible combinations of the D4 and F8 instances.

Machines of all instance types were only allocated once and not reallocated between executions. Further experiments after deallocating and allocating new instances of the same types (not shown in the paper) resulted in quantitatively and qualitatively very similar behaviors. Results show the average values of 10 executions. We begin with a discussion of the NAS benchmarks, followed by the *BRAMS* and *Alya* applications.

4.1 The NAS Benchmarks

The cost efficiency results of the NAS benchmarks are shown in Fig. 3. In the figures, the line represents the cost efficiency when varying the mix of instance types. To calculate the cost efficiency metric, we use the following equation [13].

$$cost\ efficiency = execution\ time \times price\ of\ execution \tag{1}$$

Lower values of the metric indicate a higher cost efficiency.

In the figures, the y axes show the values of the metric, while the x axes indicate the mix of instance types in the form a/b, where a represents the number of D4 instances and b represents the number of F8 instances. 0/8 and 8/0 are the homogeneous clouds, while 1/7 – 7/1 are heterogeneous instances. The most cost efficient instance combination is marked with a dashed circle (◌). The results of the instance combination determined by our mechanism are marked with an unbroken circle (○).

Several interesting results can be pointed out. First of all, heterogeneous clouds are the most cost efficient environments for the majority of the benchmarks. Heterogeneous environments are beneficial for five out of the eight benchmarks (*BT*, *EP*, *FT*, *MG*, and *SP*), while homogeneous environments are more appropriate for *CG*, *IS*, and *LU*. The five benchmarks have an increased cost

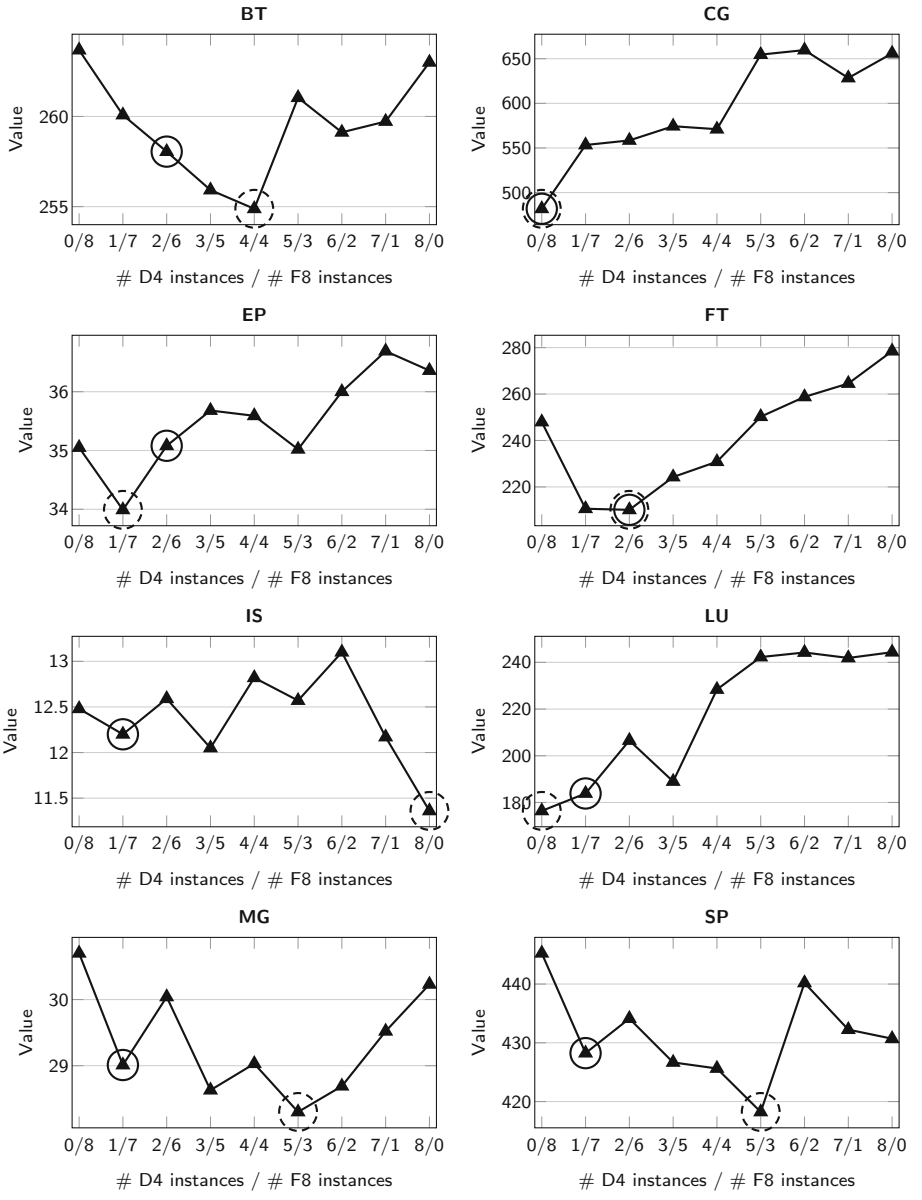

Fig. 3. Cost efficiency results for the NAS benchmarks on the D4 and F8 instances for different combinations of instances. Lower values indicate a higher cost efficiency. The highest cost efficiency is marked with a dashed circle ◌. The results of our mechanism are marked with an unbroken circle ○.

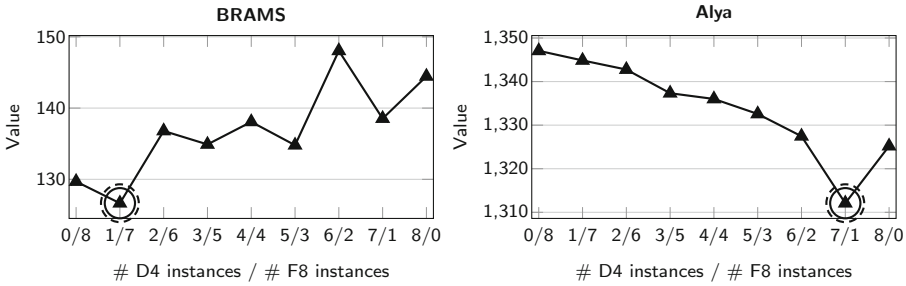

Fig. 4. Cost efficiency results for *BRAMS* and *Alya* on the D4 and F8 instances. Lower values indicate a higher cost efficiency. The highest cost efficiency is marked with a dashed circle ◌. The results of our mechanism are marked with an unbroken circle ○.

efficiency between 3.0% (*SP*) and 18.0% (*FT*) compared to the best cost efficiency of a homogeneous environment. Over all NAS benchmarks, cost efficiency was improved on average by 6.6%. These results show that cost efficiency can be improved substantially via heterogeneous clouds.

Although not achieving the optimal gains for all applications, our mechanism is able to result in substantial cost efficiency improvements close to the optimum in most scenarios, with an average improvement of 4.7%. This shows that profiling all possible instance combinations is not required in order to reduce costs. Another important result is that almost the whole spectrum of heterogeneous and homogeneous instances is the most cost efficient environment at least in one experiment. This indicates that simple policies that do not take the specific characteristics of the environment and application behavior into account can not result in optimal cost efficiency.

The performance analysis of the heterogeneous allocations is also an important aspect for the user. There are four benchmarks that presented performance losses (*BT*, *EP*, *MG*, and *SP*), between 0.2% (*SP*) to 5.5% (*MG*). It is important to remark that when comparing the performance loss with the cost efficiency gain, the heterogeneous allocations present better ratios for all the cases. This means that the performance decrease is less than the cost efficiency gain. The best ratios were obtained with *SP* (0.2% performance loss, 3.0% cost efficiency gain) and *EP* (0.7% performance loss, 3.1% cost efficiency gain). On average, performance was reduced by 0.4% in the optimal case, and 1.2% with our mechanism.

4.2 *BRAMS* and *Alya*

The cost efficiency results for the two scientific applications, *BRAMS* and *Alya*, are shown in Fig. 4. The results echo our analysis of the NAS benchmarks. Both applications can benefit significantly from a heterogeneous environment and show significant cost efficiency improvements in almost all cases. When

comparing the results of *BRAMS*, we observed that *BRAMS* presented a cost efficiency gain of 4.6%.

When analyzing the results of *Alya*, we observed a performance loss of 0.4%, while presenting a cost efficiency gain of 1.8% for the cloud tenant. Observing the load distribution of *Alya* in Fig. 2, we note that the imbalance of Alya is high, with a few processes executing much more operations than the average. The instances used in our experiments, D4 and F8, present a close performance. Due to the *Alya* load distribution, we can conclude that it could benefit from instances with higher differences between them, and from mixing more instances types.

5 Related Work

Yeo and Lee [17] analyzed how periodically upgrading hardware in datacenters introduced heterogeneity and how the service provider could mitigate its impact on performance for the end user. However, this work does not allow the cloud tenants to exploit the information about the underlying infrastructure to improve the cost/efficiency of their applications.

Zhang et al. [19] present a dynamic capacity provisioning manager that allows workload division using a heterogeneity-aware algorithm. Their work considers heterogeneity in machine hardware from production datacenters and from the workload in them. They evaluate their algorithm simulating a heterogeneous cluster. They were able to improve the utilization of the cluster and scheduling without compromising the workload. Their work takes the heterogeneity of VMs into account, but their focus is on improvements from the provider perspective.

Gupta et al. [8] propose a technique to improve the performance of parallel applications in the cloud with task placement. The authors place the tasks according to the interference between different applications by analyzing their cache memory usage, and from a description provided by the user. They do not take different types of instances into account.

Zhang et al. [20] exploited cloud heterogeneity in several MapReduce clusters to select the best cost/performance deployment. They simulate their configurations of 3 instance sizes looking to obtain the same application performance but with different provisioning costs. The validation was done on Amazon using MapReduce jobs with no data dependencies between them. Their results showed a difference in cost when using homogeneous or heterogeneous deployments. For some of the applications evaluated they obtained significant cost savings. Our work include MPI applications, and benchmarks with communication between instances.

Carreño et al. [4] created a communication-aware task mapping for cloud environments with multiple instances. Their work analyzes heterogeneity in communication between the tasks and in the network interconnections between cloud instances. They use this information to map tasks that communicate a lot to faster instances, improving inter-instance communication performance. However, their work uses the same type of VMs for each execution and they do not take

computational performance into account. In our work, we compare the performance when mixing different types of VMs.

Wang and Shi [16] developed a task-level scheduling algorithm to comply with budget and deadline constrains. They analyze heterogeneity as the variety of options of virtual machines from a provider and the underlying variations in hardware that exists for each instance. They developed a parallel greedy algorithm that improves deployment to comply with the constrains. Their work is different because it does not try to optimize the cost/efficiency of the solution but tries to respect the user constrains. Also their work was not validated using an actual public cloud infrastructure.

6 Conclusions

The cloud has become an interesting environment for the execution of parallel applications due to the easy and flexible availability of different instance types that vary in performance and price. Most current cloud deployments for parallel applications are homogeneous, that is, they are composed of a number of instances of the same type. In this paper, we motivated and analyzed a new type of deployment that is based on heterogeneous instances of different types. Since the computational demands of parallel applications are not uniform in most cases, such a heterogeneous cloud can better match the requirements of the application, improving the price and cost efficiency of the execution.

Our evaluation with MPI-based applications on an Azure cloud shows that the cost efficiency can be improved significantly, by up to 18%, depending on the load imbalance of the application, while maintaining a similar performance. Gains achieved by our mechanism were close to the optimum in most cases, showing that improvements from heterogeneous execution do not require a time-consuming evaluation of all possible instance combinations.

For the future, we plan to take communication within the parallel application into account when making placement decisions, and we will extend our analysis to consider more than two different types of instances with different numbers of cores on each type.

Acknowledgments. This research received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the HPC4E project, grant agreement no. 689772. Additional funding was provided by FAPERGS in the context of the GreenCloud Project.

References

1. 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. Int. J. Supercomput. Appl. **5**(3), 66–73 (1991)
2. Bassem, C., Bestavros, A.: Network-constrained packing of brokered workloads in virtualized environments. In: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (2015)
3. Bhatele, A., Titus, A.R., Thiagarajan, J.J., Jain, N., Gamblin, T., Bremer, P.T., Schulz, M., Kale, L.V.: Identifying the culprits behind network congestion. In: IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 113–122 (2015)
4. Carreño, E.D., Diener, M., Cruz, E.H.M., Navaux, P.O.A.: Communication optimization of parallel applications in the Cloud. In: IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid), pp. 1–10 (2016)
5. Crago, S.P., Walters, J.P.: Heterogeneous cloud computing: the way forward. Computer **48**(1), 59–61 (2015)
6. Freitas, S.R., Longo, K.M., Silva Dias, M.A.F., Chatfield, R., Silva Dias, P., Artaxo, P., Andreae, M.O., Grell, G., Rodrigues, L.F., Fazenda, A., Panetta, J.: The coupled aerosol and tracer transport model to the Brazilian developments on the regional atmospheric modeling system (CATT-BRAMS) - Part 1: model description and evaluation. Atmos. Chem. Phys. **9**(8), 2843–2861 (2009)
7. Gabriel, E., et al.: Open MPI: goals, concept, and design of a next generation MPI implementation. In: Kranzlmüller, D., Kacsuk, P., Dongarra, J. (eds.) EuroPVM/MPI 2004. LNCS, vol. 3241, pp. 97–104. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30218-6_19
8. Gupta, A., Kalé, L.V., Milojicic, D., Faraboschi, P., Balle, S.M.: HPC-aware VM placement in infrastructure clouds. In: IEEE International Conference on Cloud Engineering (IC2E), pp. 11–20 (2013)
9. Houzeaux, G., Vázquez, M., Aubry, R., Cela, J.M.: A massively parallel fractional step solver for incompressible flows. J. Comput. Phys. **228**(17), 6316–6332 (2009). http://dx.doi.org/10.1016/j.jcp.2009.05.019
10. de Melo, A.C.: The new Linux 'perf' tools. In: Linux Kongress (2010)
11. Petitet, A., Whaley, R.C., Dongarra, J., Cleary, A.: HPL - a portable implementation of the high-performance linpack benchmark for distributed-memory computers (2012). http://www.netlib.org/benchmark/hpl/
12. Roloff, E., Birck, F., Diener, M., Carissimi, A., Navaux, P.O.A.: Evaluating high performance computing on the Windows Azure platform. In: IEEE International Conference on Cloud Computing (CLOUD), pp. 803–810 (2012)
13. Roloff, E., Diener, M., Carissimi, A., Navaux, P.O.A.: High performance computing in the Cloud: deployment, performance and cost efficiency. In: IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 371–378 (2012)
14. Saad, A., El-Mahdy, A.: Network topology identification for Cloud instances. In: International Conference on Cloud and Green Computing, pp. 92–98 (2013)
15. The Open MPI project: mpirun man page (2013). http://www.open-mpi.de/doc/v1.6/man1/mpirun.1.php#sect9
16. Wang, Y., Shi, W.: Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds. IEEE Trans. Cloud Comput. **2**(3), 306–319 (2014)

17. Yeo, S., Lee, H.H.: Using mathematical modeling in provisioning a heterogeneous Cloud computing environment. Computer **44**(8), 55–62 (2011)
18. Zant, B.E., Gagnaire, M.: Performance and price analysis for Cloud service providers. In: Science and Information Conference (SAI), pp. 816–822 (2015)
19. Zhang, Q., Zhani, M.F., Boutaba, R., Hellerstein, J.L.: Harmony: dynamic heterogeneity-aware resource provisioning in the Cloud. In: Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, pp. 510–519. IEEE Computer Society, Washington, DC (2013)
20. Zhang, Z., Cherkasova, L., Loo, B.T.: Exploiting Cloud heterogeneity for optimized cost/performance MapReduce processing. In: Proceedings of the Fourth International Workshop on Cloud Data and Platforms, CloudDP 2014, pp. 1:1–1:6 (2014)