Exploiting Load Imbalance Patterns for Heterogeneous Cloud Computing Platforms

Eduardo Roloff, Matthias Diener, Luciano P. Gaspary and Philippe O. A. Navaux
Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

Keywords: Cloud Computing, Cost Efficiency, Load Imbalance, Heterogeneity.

Abstract: Cloud computing providers offer a variety of instance sizes, types, and configurations that have different prices but can interoperate. As many parallel applications have heterogeneous computational demands, these different instance types can be exploited to reduce the cost of executing a parallel application while maintaining an acceptable performance. In this paper, we perform an analysis of load imbalance patterns with an intentionally-imbalanced artificial benchmark to discover which patterns can benefit from a heterogeneous cloud system. Experiments with this artificial benchmark as well as applications from the NAS Parallel Benchmark suite show that the price of executing an imbalanced application can be reduced substantially on a heterogeneous cloud for a variety of imbalance patterns, while maintaining acceptable performance. By using a heterogeneous cloud, cost efficiency was improved by up to 63%, while performance was reduced by less than 7%.

1 INTRODUCTION

Executing large parallel applications in cloud environments is becoming an important research focus in cloud computing. Among the three service models offered by the Cloud Computing model, the most suitable for High-Performance Computing (HPC) is the IaaS model, since it provides instances that can be customized according to the users' needs and combined to build a cluster system. Currently, many cloud providers offer a large number of instance types that target applications from the HPC domain. In contrast to traditional cluster systems, which usually consist of homogeneous cluster nodes, cloud systems offer the possibility to flexibly allocate different types of instances and interconnect them, thus creating large *heterogeneous* platforms for distributed applications. Normally the heterogeneity of the cloud is explored near the hardware level (Dong et al., 2017) and is most related to the provider side. However, there are few studies that aim to explore the heterogeneity in public clouds.

One way in which such a heterogeneous system can be interesting is by matching the demands of the application to the underlying hardware. In the context of this paper, we focus on matching the *load* of the different processes of a parallel application to different instance types, thus providing a way to mitigate the *load imbalance* that is common in such applications. Previous work (Roloff et al., 2017a; Roloff et al., 2017b) has shown that such a matching can be beneficial for the cost efficiency of an application, by executing processes with a larger load on faster but more expensive instances, while executing processes with a lower load on slower but cheaper instances. In this way, the overall cost of the execution can be reduced, while maintaining the same performance. Due to the *principle of persistence* (Kalé, 2002), which says that load imbalance will stay the same during execution and between several executions for most applications, profiling and matching has to be done only once per application.

In this paper, we study the benefits of heterogeneous clouds, focusing on which type of load imbalance is most suitable for these cloud systems. We introduce an MPI-based benchmark, *ImbBench*, that can simulate several types of load imbalance patterns in parallel applications. With this benchmark, users can profile cloud instances to choose the instances that best meet their needs. We use the benchmark on the Microsoft Azure Cloud provider to prove that is it possible to save money while not reducing the performance of an unbalanced application across a wide range of imbalance patterns, using up to three different instance types. Furthermore, we also validate our approach using the MPI version of the NAS Parallel Benchmarks (NPB). All of our evaluations were done using the IaaS service model.

Roloff, E., Diener, M., Gaspary, L. and Navaux, P.
Exploiting Load Imbalance Patterns for Heterogeneous Cloud Computing Platforms.
DOI: 10.5220/0006807502480259
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 248-259
ISBN: 978-989-758-295-0

2 THE IMBBENCH BENCHMARK

The HPC field uses many benchmarks that focus on specific performance aspects, such as the LINPACK benchmark (Dongarra et al., 2003), which is used to measure performance for the TOP500 list; the NAS suite (Bailey et al., 1991), which consists of Computational Fluid Dynamics (CFD) benchmarks; and the Rodinia suite (Che et al., 2009), which was designed for benchmarking CPU+GPU systems, among others. However, to the best of our knowledge, there is a lack of benchmark suites designed for cloud computing. Moreover, there are few tools that help the users to exploit the heterogeneity of cloud computing instances, both in terms of price and performance.

In most cases, HPC applications are executed on *homogeneous clusters* (or clouds), which are clusters that consist of several nodes with the same configuration, number of cores, memory, and disk. It is natural that the available benchmarks were designed with a homogeneous behavior. However, cloud computing provides a large variety of heterogeneous resources, in which users can build clusters of nodes with different characteristics (Chohan et al., 2010), thus creating a *heterogeneous cluster*.

To better evaluate heterogeneous systems, we developed the *Imbalance Benchmark (ImbBench)*, which is a set of MPI-based applications that simulate several behaviors in terms of process loads. ImbBench was designed with the heterogeneity of the cloud in mind. Its goal is to help the user to choose the most suitable configuration to execute an application in the cloud. ImbBench distributes the load among all the available processes according to a preselected imbalance pattern. The patterns that can be simulated by ImbBench are illustrated in Figure 1. In the figure, the y-axis indicates the relative load of each rank (normalized such that the maximum load equals 100), while the x-axis shows the MPI ranks (there are 64 ranks in total). These patterns were chosen as they represent common types of imbalance in parallel applications.

In the *Balanced* pattern, all MPI ranks execute the same load, simulating the most desirable behavior for an HPC application. The *Multi-level* pattern shows distinct load levels, between two and eight levels, simulating an application with several different loads among the processes. The *Two-level* pattern is a special case of the Multi-level pattern. We include it because is a fairly common imbalance pattern of parallel applications. The *Amdahl* pattern simulates an application that has one process that executes much more work than all the others processes, normally a behavior where an application needs a central process to distribute all the tasks and collect the results. Finally,

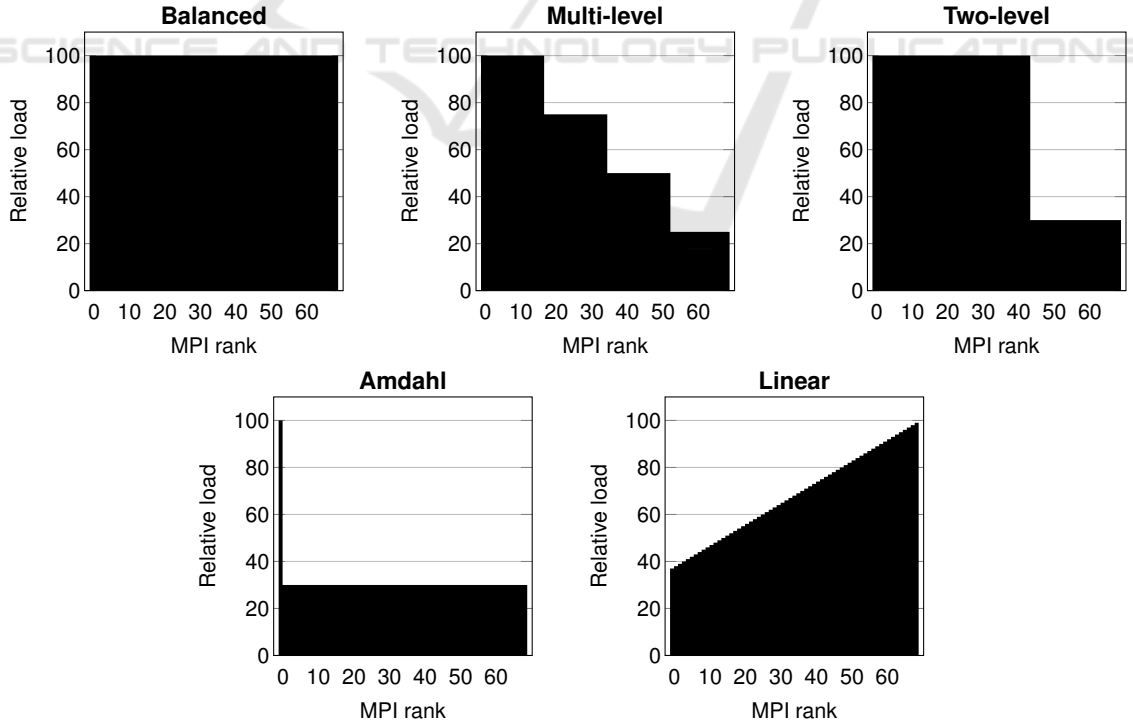

Figure 1: Overview of the load imbalance patterns that ImbBench can create. The y-axis indicates the relative load of each rank (0–100), while the x-axis contains the MPI ranks (0–63).

the *Linear* pattern simulates an application where all processes have a different load, starting from a low load on rank 0 and linearly increasing up to rank n.

For the load simulation, ImbBench uses random number generation. When a process needs to execute a higher load than another, it simply generates more random numbers. In a future version, we intend to experiment with other load creation strategies, such as other algorithms, or strategies that focus on floating point calculations or the memory subsystem, among others. ImbBench can create these patterns for any number of MPI ranks. The current version scales up to 512 ranks.

3 EXPLORING LOAD IMBALANCE PATTERNS WITH IMBBENCH

Public cloud providers offer a wide range of instance configurations. Two of the main cloud providers, Microsoft Azure and Amazon Web Services, provide several configurations to specific purposes. They offer machines with a focus on Storage, Memory, GPU/FPGA, among others. In terms of number of cores, they vary from an instance that shares a core with other users, up to instances with 128 exclusive cores. The price per hour varies from an instance that costs less than a cent up to an instance that costs US$ 14.00 per hour. Microsoft Azure has more than 60 different instance configurations while Amazon AWS offers more than 50 different configurations.

These large numbers of instance configurations indicate that there are many possibilities to create a heterogeneous cloud, but it is necessary to compare instances for their performance and cost efficiency in order to find the most suitable combination for a given application load imbalance. In this section, we use ImbBench to analyze Azure instances and determine the benefits of heterogeneous environments for different imbalance patterns.

3.1 Measuring Cost Efficiency with the Cost-delay Product

An important aspect when evaluating cloud instance types is comparing their price and performance tradeoff. Several basic metrics can be employed to measure this tradeoff. In our previous work (Roloff et al., 2017a), we introduced a new metric, the *cost-delay product (CDP)*, that can be used to compare the cost efficiency of an application being executed in a given environment. As it will be used in the rest of the paper, this section gives a brief overview of this metric.

The *CDP* metric is defined with the following equation:

$$CDP = cost\,of\,execution \times execution\,time \quad (1)$$

The *cost of execution* represents the price of the cloud environment used to execute the application. Most public cloud providers base their price model on hours of use, and the cost is then the price per hour (in US$) of all allocated instances. The *execution time* is the application's execution time in the allocated environment. Lower values of the *CDP* metric indicate a better cost efficiency for a certain application in a cloud environment.

The *CDP* metric can be extended with a weighted approach, depending on whether cost or performance is more important to the user. These metrics, C^2DP and CD^2P respectively, are defined as follows:

$$C^2DP = (cost\,of\,execution)^2 \times execution\,time \quad (2)$$

$$CD^2P = cost\,of\,execution \times (execution\,time)^2 \quad (3)$$

Cloud users can calculate these metrics for their target environments to compare the cost efficiency of an application.

3.2 ImbBench Evaluation

In our experiments, we use Microsoft Azure for the evaluation, which has shown a good performance for HPC applications (Roloff et al., 2012).

3.2.1 Performance and Price of Azure Instance Types

Among all the available instances in Azure, we selected the instances with 16 cores, because this size of instances has seven different configurations offering a multitude of heterogeneous choices. The instances used in our evaluation were: D16, D5, E16, F16, H16, G4, and L16. They are configurable with different memory and disk sizes and processor types.

To profile the instances, we executed the High-Performance Linpack benchmark (Dongarra et al., 2003) to measure the processing capacity of each instance in GigaFlops, the same methodology used to create the TOP500 rank. With the Linpack results we can organize the instances into three different groups of processing power: High, Medium, and Low. The High group contains only the H16 instance

Table 1: Characteristics of the Azure Instance Types.

Name	Price/hour	Linpack (GFlops)	GFlop/US$
D16	US$ 0.936	247.54	264.46
D5	US$ 1.117	280.04	250.71
E16	US$ 1.186	254.97	214.98
F16	US$ 0.997	263.51	264.31
G4	US$ 3.072	417.32	135.85
H16	US$ 1.941	657.64	338.81
L16	US$ 1.376	406.63	295.52

type, which achieved more than 650 GigaFlops and was the most powerful machine in our experiments. The Medium group consists of the L16 and G4 instances, both achieved slightly more than 400 GigaFlops. The Low group is composed of the other four machines types: D16, D5, E16, and F16, with a Linpack performance of about 250 GigaFlops.

The price of the instances can also be classified into the same three groups. The G4 and H16 instances are in the High and Medium groups, respectively, with the G4 instance costing around three dollars per hour and the H16 instance costing around two dollars. All the other instances are in the Low cost group, with the price near one US dollar per hour.

Another aspect is the cost efficiency of the instances, which helps the users understand how efficient an instance is, in terms of processing power per dollar. To compare this characteristic of the instances, we calculate how many GigaFlops an instance delivers per dollar. We can calculate this metric for the Linpack benchmark as follows:

$$GigaFlop/US\$ = \frac{Linpack\,result}{Price\,per\,hour} \quad (4)$$

Using Equation 4, we can determine that the instance H16 presents the best relation between price and performance. The L16 instance is slightly lower than H16, but still presented a good relation. The G4 instance shows the worst price-performance relation, because it has the highest price among all the instances in our evaluation. The other instances presented a price-performance relation near the level of 250 GigaFlops per US$. Table 1 summarizes the results of the Linpack benchmark, the cost efficiency, as well as the price of the profiled instances.

3.2.2 ImbBench Performance and Cost Efficiency

After completing the profile of the instances, we executed the ImbBench to evaluate if it is possible to benefit from the heterogeneity presented in these instances of Microsoft Azure. For the evaluation, we built clusters with 64 cores, using four Azure instances. The nodes are running with Ubuntu 16.04 (kernel 4.13), GCC version 5.4, and Open MPI version 1.10.2. We executed each ImbBench pattern 30 times, the results are the average execution time of the 30 executions. As the baseline, we use the most powerful instance type, H16, which also presented the best cost efficiency. We built a homogeneous cluster with four H16 instances and compare the heterogeneous clusters against it. We created several heterogeneous configurations until we found the configuration most similar to the baseline cluster in terms of total execution time. Since we only identified three different performance levels of the Azure instances, we executed only the Two-Level and Four-Level patterns from the Multi-Level patterns of ImbBench. The Amdahl and Linear patterns were also used in the evaluation.

The Amdahl Pattern. The first analyzed pattern is the Amdahl pattern. Figure 2 shows the results of Amdahl pattern of ImbBench using the cluster made of four H16 instances. As can be seen in the figure, rank 0 presented a significantly higher execution time (approx. 5 seconds) than the other ranks (approx. 1 second), as expected.

With these results, we determined that a heterogeneous hardware environment can be beneficial for this load imbalance pattern, by using a powerful instance together with less powerful and less expensive instances. Thus, we build a cluster with one H16 instance and three D16 instances. The reason to choose the D16 instances was that the D16 instance is the cheapest instance among all the instances used in our evaluation. The results of the heterogeneous execution of Amdahl pattern are shown in Figure 3.

As we can see in the figure, the total execution time remains the same, because the process with the highest demand was executed in the same instance as before, and the other processes, even when executed in D16 instances, do not reduce the total execution time. It is possible to note that the processes from 0 to 15 executed faster than the processes 16 to 63. The reason is that the processes 0 to 15 were executed on the H16 instance, and took 1 second to finish, and the processes 16 to 63 were executed in the D16 instances, and took approx. 3 seconds to finish. We can conclude that, even with 3/4 of the processes executed on instances with less processing power, the total execution time was not affected, because they executed faster than the process with the highest load.

In terms of price, the execution of an Amdahl-like application in a heterogeneous environment is very advantageous. The price per hour of the H16 cluster is US$ 7.764 and the price per hour of the heteroge-

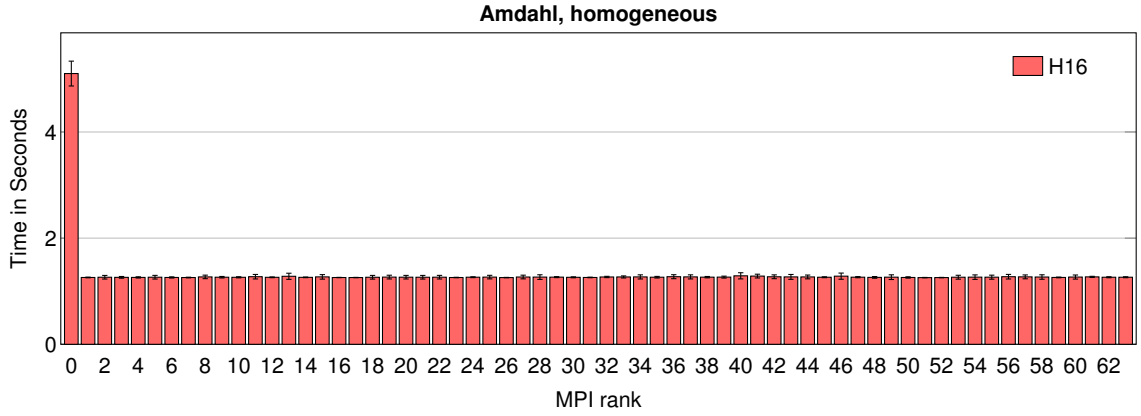

Figure 2: Execution time of ImbBench using the Amdahl pattern executed in the homogeneous cluster of H16 instances.

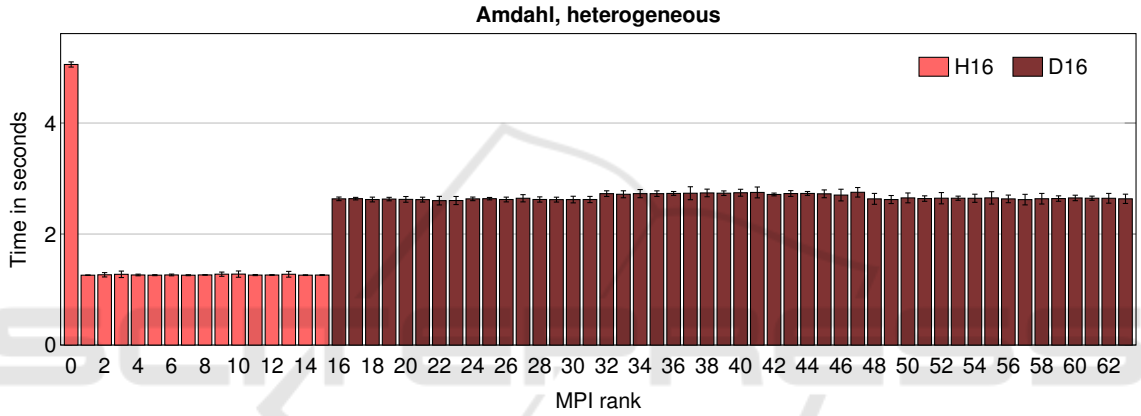

Figure 3: Execution time of ImbBench using the Amdahl pattern in the heterogeneous cluster of one H16 instance and three D16 instances. Ranks 0-15 are running on the H16 instance, while 16-63 are running on the D16 instances.

neous cluster is US$ 4.749, a saving of US$ 3.015 per hour, corresponding to a 38% total cost reduction.

The Two-level Pattern. The next pattern is the Two-Level, where the application is divided into two levels of demand. The higher level computes a certain amount of work and the low level computes exactly half of the high level. Figure 4 shows the results of the Two-Level pattern of ImbBench using the cluster made of four H16 instances. As seen in the figure, the odd processes are the high demand processes and they took around 5 seconds to perform their work. The even numbers are the low demand processes and they fulfilled their work in around 2.5 seconds.

Analyzing the results, we can conclude, as in the Amdahl pattern, that a user of an application with this behavior can take advantage of the heterogeneous cloud computing instances by mixing two instance types. Using an Instance with high processing power to execute the processes with high demand and a instance with less processing power to execute the processes with less demand. We build a cluster with two instance types and with two instances of each type. We chose the H16 and D16 instances, the H16 because it is the baseline instance and the D16 because it is the cheap instance in our chosen group. The results of the Two-Level pattern execution on the heterogeneous cluster are shown in Figure 5.

As we can see in the figure, the total execution time was slightly higher than the execution on the H16 cluster. We can observe that the even processes, which are processes with less demand, now took more time to perform their work, because they were executed in D16 instances. We can conclude that, even with half of the processes being executed in instances with less processing power, the total execution time presented only a small increase.

In terms of price, the execution of such an application in a heterogeneous environment is very advantageous. The price per hour of the H16 cluster is US$ 7.764 and the price per hour of the heterogeneous cluster is US$ 5.754, a saving of US$ 2.01 per hour or a 25% reduction of the total cost.

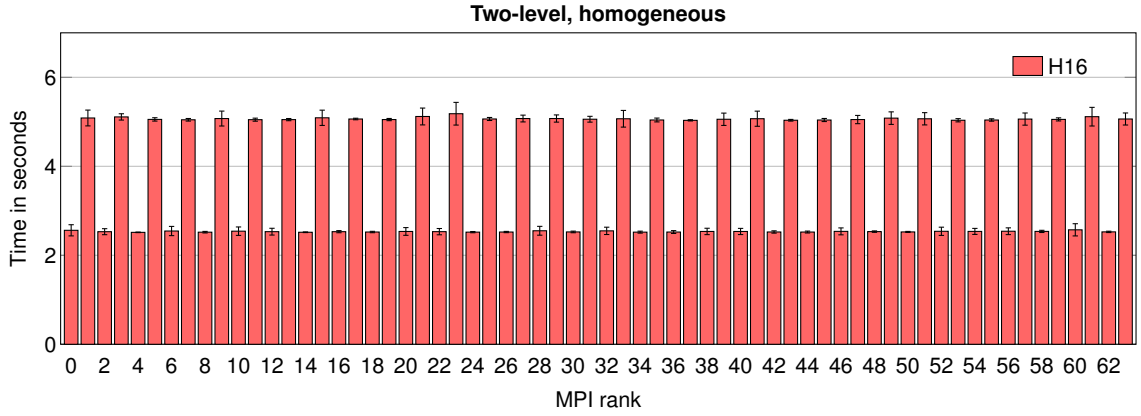

Figure 4: Execution time of ImbBench using the two-level pattern for application load in the homogeneous H16 cluster.

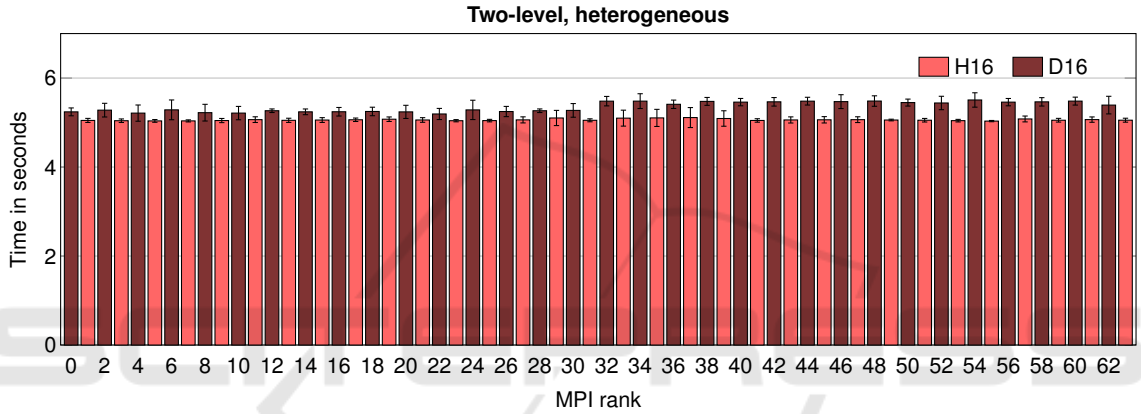

Figure 5: Execution time of ImbBench using the two-level pattern for application load in the heterogeneous cluster. Even ranks are running on D16 instances, odd ranks on H16 instances.

The Four-level Pattern. In the Four-Level pattern, the application load is divided into four levels of demand. The difference between the levels is constant, the lowest level computes X instructions, the next level computes $2X$ and so on. Figure 6 shows the results of the Four-Level pattern of ImbBench using the homogeneous cluster of four H16 instances. The figure shows that the processes' demands are divided into groups of four, where the fourth process in each group is the one that made more computation than the others. In practical terms, the time spent by every fourth process is the time spent by the application, approximately 5 seconds.

By analyzing the results, we can see that the first, second and third processes can be executed on machines with less power without increasing the total execution time. After a few simulations, we determined that the most suitable configuration for this pattern was to create a cluster with three different instance types. The processes with less demand were executed on two D16 instances, the processes with intermediate demand were executed on an L16 instance and the processes with high demand were executed on an H16 instance. The results of the execution of the Four-Level pattern on the heterogeneous cluster are shown in Figure 7.

We can observe in the figure that the execution time was slightly higher than the execution on the H16 cluster. The group of processes with less demand executed in D16 instances increased their time, but do not affect the total time. The group of processes that were executed in L16 instance increased their time as well and the processes executed in a H16 instance keep their execution time. We can conclude that, even with 3/4 of the processes being executed on instances with less processing power, the total execution time was increased only slightly.

In terms of cost efficiency, the execution of an application similar to the Four-level pattern is advantageous. The price per hour of the H16 cluster is US$ 7.764, and the price per hour of the heterogeneous cluster is US$ 5.189, a saving of US$ 2.575 per hour or a 33% total execution cost.

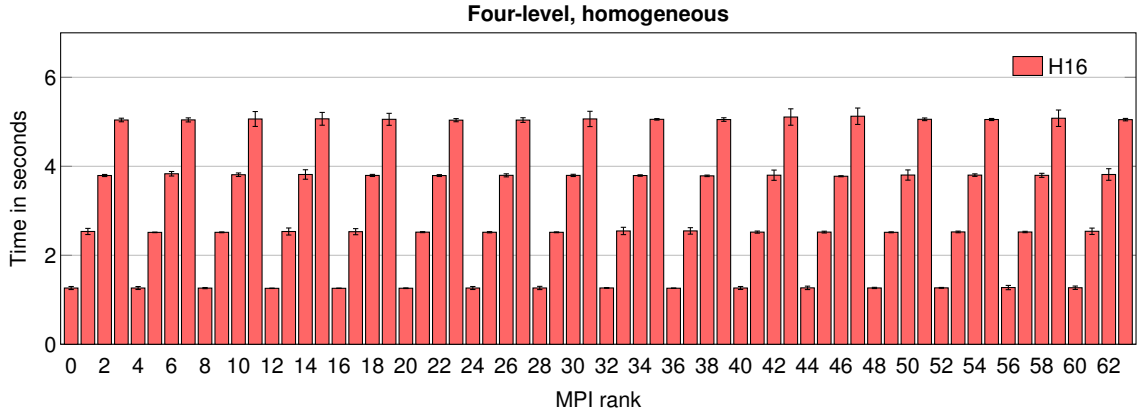

Figure 6: Execution time of ImbBench using the four-level pattern for application load in the homogeneous H16 cluster.

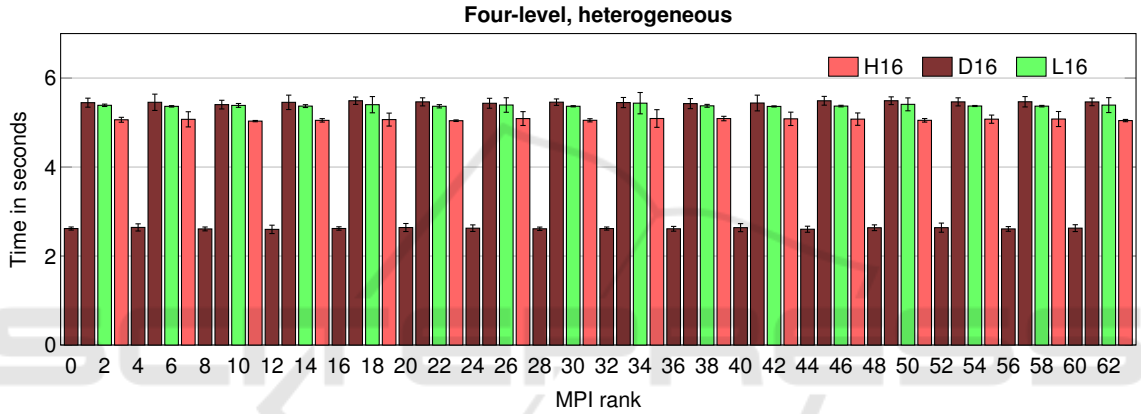

Figure 7: Execution time of ImbBench using the four-level pattern for application load in the heterogeneous cluster. Ranks 0-1,4-5,8-9,... run on D16 instances, ranks 2,6,10,14,... run on an L16 instance , ranks 3,7,11,15,... run on an H16 instance.

The Linear Pattern. The last pattern is the Linear pattern, where each process of the application has a different load. The process with rank 0 has the lowest computational demand and the demand increases in a linear way when the rank increases. Figure 8 shows the results of the Linear pattern of ImbBench using the cluster of four H16 instances. The figure shows that the processes' execution time increases as expected when the process rank increases. The total execution time of the application is the execution time of the process with the highest rank, which is around 5 seconds.

To create a configuration with mixed instances that does not affect the execution time, a few simulations were made and the most suitable configuration was the same as in the Four-Level pattern; a cluster with two D16 instances, one L16 instance and one H16 instance. The first 32 processes were executed on the D16 instances, the processes from 32 up to 47 were executed on the L16 instance and the processes from 48 up 63 were executed on the H16 instance. The results of the execution of the Linear pattern in the heterogeneous cluster are presented in Figure 9.

As seen in the figure, the first 32 processes increased their execution time, but this does not affect the total execution time, because their execution time was below the processes with high demand. The processes executed on the L16 instance, increased their time as well, and were slightly higher than the execution time on the H16 cluster. The processes that were executed on the H16 instance kept their execution time. The conclusion is the same as for the Four-Level pattern, even with 3/4 of the processes being executed in instances with less processing power, the total execution time presented an increase up less than 7%.

In terms of price, the execution of an application with a linear behavior presents advantage in an heterogeneous environment. The price per hour of the H16 cluster is US$ 7.764 and the price per hour of the heterogeneous cluster is US$ 5.189, a saving of US$ 2.575 per hour or a 33% reduction of the total cost.

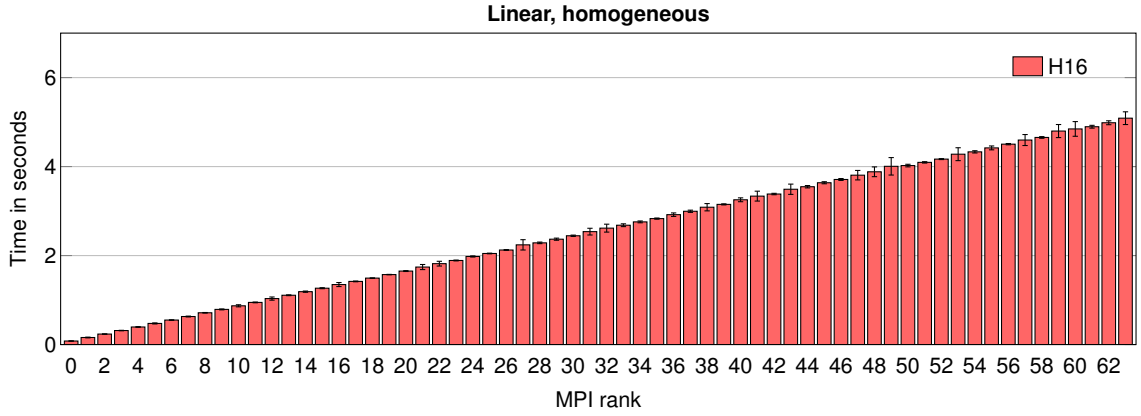

Figure 8: Execution time of ImbBench using the Linear pattern for application load in the homogeneous H16 cluster.

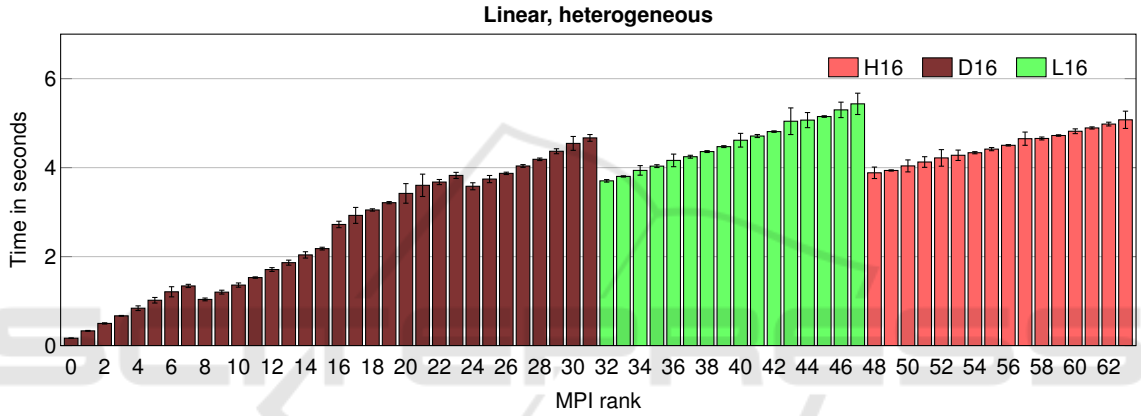

Figure 9: Execution time of ImbBench using the Linear pattern for application load in the heterogeneous cluster. Ranks 0-31 run on D16, ranks 32-47 on L16, ranks 48-63 on H16.

Summary of Results. Table 2 summarizes the results of ImbBench using both homogeneous and heterogeneous clusters. As we can see in the table, the performance loss when using the heterogeneous configuration was less than 7% for all the cases. The only exception is the Amdahl Pattern, where we observed a small performance gain. When analyzing the cost efficiency, the *CDP* results are better with the heterogeneous configurations, with a minimum gain of 21.23%, which means that this configuration is 21.23% more cost efficient than the homogeneous one. If we use the C^2DP to focus on the cost of execution, then the results are much more favorable for the heterogeneous configurations, with cost efficiency gains of up to 62.92% compared to the homogeneous configuration. When the CD^2P is used to focus on execution performance, the results of the heterogeneous configurations presented results up to 39.92% better than the homogeneous configuration. In terms of costs, the heterogeneous configurations were between 25% and 38% cheaper than the homogeneous one, and in a cloud scale these results can have a huge impact in the user's budget.

Table 2: Summary of the ImbBench results.

Pattern	Execution Time			CDP			C^2DP			CD^2P		
	Hom.	Het.	gains	Hom.	Het.	gains	Hom.	Het.	Gains	Hom.	Het.	gains
Two-Level	5.18s	5.51s	-5.92%	1.12	0.88	21.23%	2.41	1.41	41.62%	5.79	4.85	16.27%
Four-Level	5.13s	5.49s	-6.65%	1.11	0.79	28.40%	2.38	1.14	52.15%	5.67	4.35	23.29%
Amdahl	5.10s	5.05s	0.90%	1.10	0.67	39.38%	2.37	0.88	62.92%	5.61	3.37	39.92%
Linear	5.09s	5.43s	-6.35%	1.10	0.73	33.60%	2.37	1.13	52.30%	5.58	4.26	23.78%

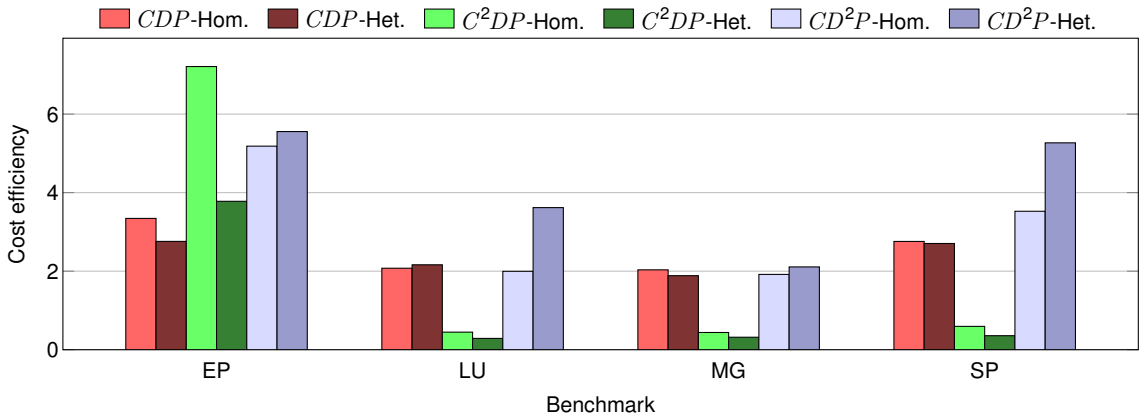

Figure 10: Cost efficiency results of the NAS benchmarks. Lower values indicate higher cost efficiency.

4 EVALUATION OF THE NAS PARALLEL BENCHMARKS

After evaluating heterogeneous clouds with ImbBench, we tested whether the heterogeneous configurations are suitable for real HPC applications. We chose the MPI version of the NAS suite of parallel benchmarks (NPB) (Bailey et al., 1991), version 3.3.1, which is a widely used workload in the HPC community. Most applications of NAS show imbalance patterns similar to the Amdahl and Two-level patterns discussed in Section 2. From the NAS suite, we selected the applications with the highest imbalance for our evaluation: EP, LU, MG, and SP. We measure the cost efficiency of the execution on the H16 homogeneous cluster and several mixed instances, using 64 cores and MPI ranks in all cases, as before. All the NAS applications were executed 30 times, and we show the average values for the results.

Table 3: Summary of the NAS results.

Name	Metric	Gains	Best het. configuration
EP	*CDP*	17%	1-H16 3-F16
EP	*C²DP*	47%	1-H16 3-F16
EP	*CD²P*	-7%	1-H16 3-F16
LU	*CDP*	-4%	1-H16 3-L16
LU	*C²DP*	35%	1-H16 3-D16
LU	*CD²P*	-38%	1-H16 3-L16
MG	*CDP*	7%	1-H16 3-L16
MG	*C²DP*	27%	1-H16 3-D16
MG	*CD²P*	-9%	1-H16 3-L16
SP	*CDP*	2%	1-H16 3-D16
SP	*C²DP*	39%	1-H16 3-D16
SP	*CD²P*	-49%	2-H16 2-L16

The results of the four homogeneous H16 instances and the best heterogeneous configurations are shown in Figure 10. As we can see in the figure, the applications EP, MG and SP presented better *CDP* in the heterogeneous execution than the homogeneous. This means that these applications executed in an acceptable time with a lower cost. As expected the *C²DP* was better in all the four heterogeneous executions, because its dominant factor is the cost.Regarding to the *CD²P*, the homogeneous configurations were slightly better in EP and MG and much better in LU and SP.

The best heterogeneous configuration for LU regarding the three metrics was one H16 instance and three F16 instances. LU performed better for *CDP* with a cluster composed of one H16 instance and three F16 instances. For the *C²DP*, the best result was achieved using a configuration with one H16 instance and three D16 instances, and for the *CD²P* the best result was achieved with one H16 and three F16 instances. The MG application performed better for both *CDP* and *CD²P* using a configuration with one H16 instance and three L16 instances, and for the *C²DP* metric the best result was achieved by using a configuration with one H16 and three D16 instances.

SP performed better for the *CDP* and *C²DP* by using a configuration with one H16 and three D16 instances. For the *CD²P*, the best result was achieved with a configuration of one H16 and three L16 instances. Table 3 summarizes the results of *CDP*, *C²DP*, and *CD²P* of NAS. We can conclude that it is possible to benefit from the cloud heterogeneity and mix of different instance types to create a more suitable execution environment. As a side conclusion, we performed several simulations mixing three and four instance types, but the results were not better than the ones with two instances for these benchmarks.

5 RELATED WORK

The work of Yeo et al. (Yeo and Lee, 2011) focused on the provider side. They analyzed how the constant hardware upgrades, made by the providers, introduce heterogeneity in the datacenter. They investigated how the providers could mitigate this impact to not affect the cloud users, in terms of performance and overall quality of service. However, their work does not allow the cloud tenants to exploit the information about the underlying infrastructure to improve the cost/efficiency of their applications.

From the multiple sources of heterogeneity in the cloud, Crago and Walters (Crago and Walters, 2015) analyzed how the usage of accelerators such as GPUs and Xeon Phi coprocessors could be challenging for the service stack of a cloud datacenter. Their work showed that heterogeneity may need to be exposed to users to help them adapt their applications to it and to improve the quality of service.

Mitigating the load imbalance of parallel and distributed applications is a well-studied topic (Pearce et al., 2012). Most solutions focus on software-level approaches in the application or runtime to distribute and migrate work among the tasks of the application (Blumofe and Leiserson, 1994; Min et al., 2011; Zhuravlev et al., 2012; Diener et al., 2015a). Such software approaches are however not possible for all types of applications, and often incur a runtime overhead during execution. In this paper, we use the opposite approach to mitigate load imbalance, by adapting the hardware environment to the imbalance present in the application. Apart from load imbalance, another common type of imbalance is the usage of memory controllers (Dashti et al., 2013; Diener et al., 2015b).

Su et al. (Su et al., 2013) developed a cost-efficient task scheduling algorithm for executing large programs in the cloud. Their strategy is to map tasks to cost efficient VMs while preserving the expected performance of the application. Their algorithm decides which instance produces the best ratio, but it is limited because it does not select heterogeneous VM instances, rather only a single type offered by the provider. They were able to improve the scheduling time, but validated their approach using only simulation.

Zhang et al. (Zhang et al., 2014; Zhang et al., 2015) aim to find a deployment with better cost/performance for MapReduce applications. To achieve this, they explored the cloud heterogeneity. They made their validation in Amazon AWS using MapReduce jobs with no data dependencies between them. The simulation was made using three different instance sizes and they try to obtain the same performance by reducing the cost of the allocations. Their results presented a difference in cost when using homogeneous or heterogeneous deployments. In some cases, the cost reduction was significantly high. Our work differs, because it includes MPI applications, and benchmarks with communication between instances.

Cheng et al. (Cheng et al., 2017) introduced Ant, a mechanism that adapts MapReduce tasks in heterogeneous clusters. The main purpose of Ant is to customize the MapReduce tasks with different configurations to be more suitable for execution in heterogeneous nodes, according to their configuration. However, the authors only took into account the performance and not the different costs of heterogeneous instances. Also, their mechanism was developed to work only with MapReduce tasks. Our work considers the instance prices and is suitable for a wide range of applications.

Carreño et al. (Carreño et al., 2016) developed a mechanism that maps the tasks of an application to cloud instances, taking into account the communication between the tasks and the network speed between instances. When a system with several instances is allocated in the cloud, their mechanism performs a profiling of all the instances, analyzing the network latency among all of them. This information is used to map the application processes that have a larger amount of communication and execute them on instances that have faster network interconnections. However, their work used homogeneous clusters, made of instances of the same type, for each execution and they did not take computational performance into account. In our work, we compare the performance when mixing different types of VMs.

Our previous work (Roloff et al., 2017b; Roloff et al., 2017a) introduced the concepts of heterogeneous clouds and the CDP metric, and showed their benefits for several NAS applications. In this paper, we extend this concept by performing an investigation of which types of imbalance patterns can benefit from heterogeneous clouds by using an intentionally-imbalanced benchmark. We also evaluate the advantages of using more than two different instance types in a single heterogeneous cloud system.

6 CONCLUSIONS AND FUTURE WORK

When developing HPC applications, a goal is to distribute the work equally among the tasks. However, this goal can not always be achieved and a certain amount of imbalance can be observed in most parallel applications. Heterogeneous clouds that are compo-

sed of different instance types are an interesting way to execute parallel, load imbalanced applications. In such a heterogeneous system, tasks with lower computational demands execute on slower but cheaper machines, while tasks with higher demands execute on faster but more expensive machines, thus increasing the overall cost efficiency of the application.

In this paper, we introduced the *Imbalance Benchmark (ImbBench)*, whose main purpose is to help users to profile their environment in terms of the heterogeneity of the instances, and to discover opportunities for heterogeneous clouds. During the evaluation of ImbBench we discovered that, depending on the application imbalance pattern, it is possible to improve the cost efficiency of the cloud environment without or with only a small increase of the execution time. Our results shown that were possible to increase the cost efficiency up to 63% with less than 7% of performance reduction. Experiments with the NAS Parallel Benchmarks showed that these gains can also be observed with traditional distributed applications. These results show us that it is possible for users to take advantage of the heterogeneity offered by cloud providers.

In the future, we will increase the capabilities of ImbBench, adding more features, such as a measurement capability of the network performance, memory operations, and floating point operations, so that the environmental profile will be more accurate. We will extend our evaluation to cover a more diverse environment as well, by using other cloud providers, a private cloud, and even the instances with variable costs, such as the AWS Spot instances. Furthermore, we intend to develop a mechanism feature to help users take advantage of cloud heterogeneity in an automated way, by analyzing instance options and application behavior and providing a recommendation of the most suitable environment for the users.

ACKNOWLEDGEMENTS

This research received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the HPC4E project, grant agreement no. 689772. This research received partial funding from CYTED for the RICAP Project, grant agreement no. 517RT0529. Additional funding was provided by FAPERGS in the context of the GreenCloud Project.

REFERENCES

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., et al. (1991). The nas parallel benchmarks. *The International Journal of Supercomputing Applications*, 5(3):63–73.

Blumofe, R. D. and Leiserson, C. E. (1994). Scheduling multithreaded computations by work stealing. In *Symposium on Foundations of Computer Science (FOCS)*, pages 1–29.

Carreño, E. D., Diener, M., Cruz, E. H. M., and Navaux, P. O. A. (2016). Automatic communication optimization of parallel applications in public clouds. In *IEEE/ACM 16th International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2016, Cartagena, Colombia, May 16-19, 2016*, pages 1–10.

Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S. H., and Skadron, K. (2009). Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*, pages 44–54.

Cheng, D., Rao, J., Guo, Y., Jiang, C., and Zhou, X. (2017). Improving performance of heterogeneous mapreduce clusters with adaptive task tuning. *IEEE Transactions on Parallel and Distributed Systems*, 28(3):774–786.

Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A. N., and Krintz, C. (2010). See spot run: Using spot instances for mapreduce workflows. *HotCloud*, 10:7–7.

Crago, S. P. and Walters, J. P. (2015). Heterogeneous cloud computing: The way forward. *Computer*, 48(1):59–61.

Dashti, M., Fedorova, A., Funston, J., Gaud, F., Lachaize, R., Lepers, B., Quéma, V., and Roth, M. (2013). Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 381–393.

Diener, M., Cruz, E. H. M., Alves, M. A. Z., Alhakeem, M. S., Navaux, P. O. A., and Heiß, H.-U. (2015a). Locality and Balance for Communication-Aware Thread Mapping in Multicore Systems. In *Euro-Par*, pages 196–208.

Diener, M., Cruz, E. H. M., and Navaux, P. O. A. (2015b). Locality vs. Balance: Exploring Data Mapping Policies on NUMA Systems. In *International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 9–16.

Dong, D., Stack, P., Xiong, H., and Morrison, J. P. (2017). Managing and unifying heterogeneous resources in cloud environments. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, pages 143–150. INSTICC, SciTePress.

Dongarra, J. J., Luszczek, P., and Petitet, A. (2003). The linpack benchmark: past, present and future. *Concurrency and Computation: practice and experience*, 15(9):803–820.

Kalé, L. V. (2002). The virtualization model of parallel programming : Runtime optimizations and the state of art. In *LACSI 2002*, Albuquerque.

Min, S.-J., Iancu, C., and Yelick, K. (2011). Hierarchical work stealing on manycore clusters. In *Conference on Partitioned Global Address Space Programming Models*, pages 1–10.

Pearce, O., Gamblin, T., de Supinski, B. R., Schulz, M., and Amato, N. M. (2012). Quantifying the effectiveness of load balance algorithms. In *ACM International Conference on Supercomputing (ICS)*, pages 185–194.

Roloff, E., Diener, M., Carissimi, A., and Navaux, P. O. (2012). High performance computing in the cloud: Deployment, performance and cost efficiency. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 371–378. IEEE.

Roloff, E., Diener, M., Carreño, E. D., Moreira, F. B., Gaspary, L. P., and Navaux, P. O. A. (2017a). Exploiting price and performance tradeoffs in heterogeneous clouds. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing, UCC 2017, Austin, TX, USA, December 5-8, 2017*, pages 71–76.

Roloff, E., Diener, M., Diaz Carreño, E., Gaspary, L. P., and Navaux, P. O. A. (2017b). Leveraging cloud heterogeneity for cost-efficient execution of parallel applications. In Rivera, F. F., Pena, T. F., and Cabaleiro, J. C., editors, *Euro-Par 2017: Parallel Processing: 23rd International Conference on Parallel and Distributed Computing, Santiago de Compostela, Spain, August 28 – September 1, 2017, Proceedings*, pages 399–411. Springer International Publishing.

Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., and Wang, J. (2013). Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Computing*, 39(4–5):177 – 188.

Yeo, S. and Lee, H. H. (2011). Using mathematical modeling in provisioning a heterogeneous cloud computing environment. *Computer*, 44(8):55–62.

Zhang, Z., Cherkasova, L., and Loo, B. T. (2014). Exploiting cloud heterogeneity for optimized cost/performance mapreduce processing. In *Proceedings of the Fourth International Workshop on Cloud Data and Platforms*, CloudDP '14, pages 1:1–1:6, New York, NY, USA. ACM.

Zhang, Z., Cherkasova, L., and Loo, B. T. (2015). Exploiting cloud heterogeneity to optimize performance and cost of mapreduce processing. *SIGMETRICS Perform. Eval. Rev.*, 42(4):38–50.

Zhuravlev, S., Saez, J. C., Blagodurov, S., Fedorova, A., and Prieto, M. (2012). Survey of Scheduling Techniques for Addressing Shared Resources in Multicore Processors. *ACM Computing Surveys (CSUR)*, 45(1):1–32.