I/O-Focused Cost Model for the Exploitation of Public Cloud Resources in Data-Intensive Workflows

Francisco Rodrigo Duro(✉), Javier Garcia Blas, and Jesus Carretero

Computer Science and Engineering Department, University Carlos III, Leganes, Spain
{frodrigo,fjblas,jcarrete}@inf.uc3m.es

Abstract. Ultrascale computing systems will blur the line between HPC and cloud platforms, transparently offering to the end-user every possible available computing resource, independently of their characteristics, location, and philosophy. However, this horizon is still far from complete. In this work, we propose a model for calculating the costs related with the deployment of data-intensive applications in IaaS cloud platforms. The model will be especially focused on I/O-related costs in data-intensive applications and on the evaluation of alternative I/O solutions. This paper also evaluates the differences in costs of a typical cloud storage service in contrast with our proposed in-memory I/O accelerator, Hercules, showing great flexibility potential in the price/performance trade-off. In Hercules cases, the execution time reductions are up to 25% in the best case, while costs are similar to Amazon S3.

Keywords: Cloud · Amazon · Data-intensive · Cost model · Workflows

1 Introduction

The popularization of the cloud computing paradigm brought a new scenario to the scientific computing field. Based on the virtually limitless resources offered in a pay-per-use approach, research centers have the possibility to use cloud resources instead of the traditional HPC infrastructures. It could be even possible to combine the benefits offered by both approaches, owning an HPC cluster or supercomputer for testing and development, while deploying experiments in this HPC infrastructure augmented with as much cloud computing resources as needed or as possible given the budget of the project. This combination will lead the path to Ultrascale systems [2], large-scale complex systems that join parallel and distributed computing systems, reaching two to three orders of magnitude larger than todays systems. The research line for achieving Ultrascale systems should focus on the simplification of this scenario with mixed infrastructures, by

F. Rodrigo Duro—This work was supported by the project TIN2013-41350-P "Scalable Data Management Techniques for High-End Computing Systems" from the *Ministerio de Economía y Competitividad*, Spain.

J. Carretero et al. (Eds.): ICA3PP 2016 Workshops, LNCS 10049, pp. 244–257, 2016.
DOI: 10.1007/978-3-319-49956-7_20

transparently taking advantage of every possible computing resources available, independently of their characteristics, location, or philosophy.

However, the current technology is far from this ideal scenario. Interfaces in both HPC and cloud platforms differ and difficult its use in a combined way. The philosophy differences in cloud and HPC infrastructures, especially in the I/O subsystems, are still an unsolved inconvenience for generic applications. For tackling with this limitation, in previous works we proposed Hercules [5,9], an in-memory generic I/O architecture for data-intensive applications, as an alternative to current infrastructure-specific I/O solutions.

This lack of generic approaches does not only affect to the computing infrastructures, but also to the scientific applications. Since the introduction of the MapReduce paradigm, we have seen a change in the trends of scientific application paradigm, from the classical CPU-intensive applications (large-scale simulations, complex mathematical problems, etc.) to data-intensive applications. However, most of the technological breakthroughs emerged from the Big Data field are not fully applied to data-intensive scientific applications executed on HPC infrastructures. In the recent years, the use of workflow engines for the design, implementation, and execution of data-intensive applications in different infrastructures, is seen as the best generic approach for scientific computing.

Future Ultrascale systems will be in charge, not only of transparently offering every available resource to the users, but they will also be responsible of scheduling each computing job to the platform with the best fit for the characteristics of the application. In order to better understand when the pay-per-use cloud resources are the best option for a specific workload, it is indispensable to fully understand the incurred costs of executing an application in the cloud. In this work, we propose a model for calculating potential costs derived from the deployment of a data-intensive application over an IaaS cloud platform. This model takes especially into account the costs related with I/O operations, including the impact of deploying our proposed in-memory I/O accelerator, Hercules, as an alternative to the default cloud storage service. We have also applied this model to a study case that involves the execution of a data-intensive application, demonstrating that our solution better suits the pay-per-use philosophy for I/O operations over temporary data, flexibly adapting the performance and costs to the user requirements.

The remainder of the paper is organized as follows. Section 2 overviews previous works related to data-intensive applications in clouds. Section 3 introduces Hercules. Section 4 proposes a model for calculating the costs of deployment of an application in an IaaS cloud platform. Section 5 applies our model to the execution of a data-intensive application, comparing Amazon S3 and Hercules. Finally, Sect. 6 presents the conclusions of this work.

2 Related Work

Workflow engines, such as DMCF [8], Pegasus [1], and OmpSs [4] are software systems for designing and executing data analysis workflows. Most workflow

engines rely on the default shared storage. This implies that the I/O performance of tasks is limited by the performance of the default storage and can be greatly affected by contention. Thus, currently, the costs of working with large datasets mainly depends on infrastructures, where storage and computation resources are not completely decoupled as in the case of HDFS.

As a result, data locality-aware techniques and in-memory storage are becoming more and more important, avoiding these problems. Recent solutions like Tachyon [7] have shown the importance of data locality and in-memory storage for improving performance in data-intensive applications. Chiu et al. [3] evaluated the effects of reducing the data transfers through the use of a cooperative cache. In this work, we demonstrate that our in-memory cache solution also reduces applications production costs compared with the Amazon S3 storage system.

Yuan et al. [10] presented a novel intermediate data storage strategy for reducing the cost of the scientific cloud workflows. This strategy is based on the automatic store of the most appropriate intermediate datasets. In [11], the same authors proposed a dataset storage cost model for managing the intermediate data in a scientific cloud-aware workflow systems. Our approach differs from this one by considering both application and hardware characteristics.

3 Hercules Background

Hercules is a generic I/O architecture based on in-memory key-value stores. Hercules can be deployed as an I/O subsystem alternative to existing storage solutions such as parallel shared file system in HPC systems and cloud storage services in cloud platforms. It is especially designed for the acceleration of I/O operations over temporary data in data-intensive applications.

The main characteristics offered by Hercules are: easy deployment, flexibility, portability, scalability, and performance. Based on its easy deployment, Hercules can be flexibly configured with as many I/O nodes as necessary, depending on the requirements of the application or even depending on the requirements of each execution of the same application. A larger number of I/O nodes deployed is translated to the (several) network interfaces available, implying a greater aggregated throughput available for the applications.

Additionally, based on the generic characteristics of the architecture, Hercules can be deployed in a wide range of different infrastructures, including HPC systems [5] and cloud platforms [9], showing potential performance improvements while providing portability for existing and legacy applications. Hercules is also capable of being deployed for sharing resources with the compute nodes, enabling the possibility of exposing data locality in order to be exploited by locality-aware schedulers. However, this feature is out of the scope of this work.

4 Costs Model for In-Memory Storage on Clouds

In this section, we present a model with the objective of calculating the costs associated with the execution of a data-intensive workflow application in a public

cloud platform. We have based this model on the Amazon AWS platform, but given the similarity in billing concepts of the different existing IaaS providers, this model should be applicable to other cloud providers (i.e., Microsoft Azure). Application modeling is focused on workflow applications, represented as a Directed Acyclic Graph (DAG).

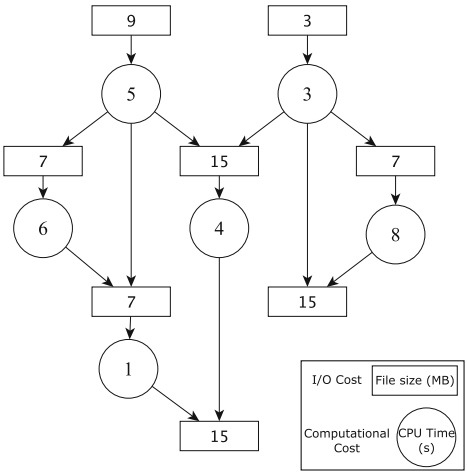

Fig. 1. Workflow model for the cost analysis.

Figure 1 shows how graph nodes (circles) represent the computational cost of each task (CPU time in seconds) while boxes represent data communication between tasks. Each box corresponds with one file, representing the file size measured in MB. Links associating tasks and files represent I/O dependencies, being write operations the links with task-to-file direction and read operations the file-to-task links. Any number of workflow instances can be executed depending on the number of existing input files.

$$C_{TOTAL} = C_{CSS} + C_{CCI} \tag{1}$$

The *total execution cost* of an application (C_{TOTAL}) is denoted as the sum of the costs of the cloud storage services (C_{CSS}) and the costs of compute instances (C_{CCI}). Both costs depend on the characteristics of the application, the characteristics of the infrastructure, and the execution time, i.e. the time needed for executing an application is lower using two computing instances than using one, but the cost of deploying two virtual machines (VMs) is greater than deploying one during the same amount of time.

4.1 Cloud Storage Service Costs

The cloud storage service costs (C_{CSS}) refers to the costs related with the I/O operations (amount of data stored, number of read/write operations, etc.) and can be calculated as:

$$C_{CSS} = \sum_{i \in F} FSize_i \cdot S_{cost} \cdot t_{ex} + \sum_{j \in E} Ein_j \cdot PUT_{cost} + \sum_{j \in E} Eout_j \cdot GET_{cost} \quad (2)$$

where F and E are respectively the sets of involved files and the in/out edges of the application. Every existing link has two different associated input and output costs. These costs will be considered depending on the nature of the I/O operation: write operations will have $IN = 1, OUT = 0$ costs (represented also as Ein), while read operations will have these values changed ($IN = 0, OUT = 1$, represented as $Eout$). Each of these costs represents one I/O operation over a specific file. $FSize$ is the *file size* of one file object in MB and S_{cost} is the store cost of files. Amazon charges the storage as USD/GB per month, so the total storage cost will depend on the total execution time (t_{ex}), introduced in Eq. 6. We only take into account the cost of execution, but any data stored before/after the execution of the workflow in the cloud storage service will be billed. Based on these parameters, the cost of storing files can be calculated. Ein and $Eout$ represent the number of input and output operations over each file, while GET_{cost} and PUT_{cost} represent the cost of every operation (a billing concept existing in most cloud storage services).

The C_{CSS} cost can be decomposed in more specific costs depending on the nature of the I/O operations. In our case, it will be especially useful to measure the cost of operations performed over input and output files ($C_{CSS_{in}}$, $C_{CSS_{out}}$) independently from the rest of I/O operations ($C_{CSS_{tmp}}$), as shown in Eq. 3.

$$C_{CSS} = C_{CSS_{in}} + C_{CSS_{tmp}} + C_{CSS_{out}} \quad (3)$$

where $C_{CSS_{in}}$, $C_{CSS_{out}}$, and $C_{CSS_{tmp}}$ take into account only storage costs and I/O operation costs related with input, output, and temporary files, respectively.

4.2 Computing Resources Costs

For the second part of the Eq. 1, the objective is to calculate the costs related with the use of computing resources. These costs include the VM instances used for executing the application and depend on the total execution time:

$$C_{CCI} = t_{ex} \cdot \sum_{i \in V} VM_{cost_i} \quad (4)$$

where V is the set of VM instances deployed during the execution of the application. In order to better represent the costs associated with the deployment of the Hercules I/O accelerator, the former equation can be decomposed differentiating the VM instances executing the application and the VM instances deployed for the Hercules I/O back-end servers:

$$C_{CCI} = t_{ex} \cdot (\sum_{i \in C} VM_{cost_i} + \sum_{j \in H} VM_{cost_j}) \quad (5)$$

where C and H are the sets of VM instances deployed during the execution of the application, for computation and Hercules purposes, respectively. t_{ex} is the *total execution time* of the application (in seconds) and VM_{cost} is the price of deploying each VM during one second (in USD/s). This cost calculation introduces the first simplification of our proposed model. In the Amazon EC2 platform, VM instances are billed for full hours, independently of being used 1 s or 59 min. In our model, we consider paying only for the useful time in seconds. This simplification can be explained by the use of the infrastructure for executing multiple batch applications. In this simplified scenario, configuration, initialization, and full hour costs can be discarded. When multiple applications are executed by the same infrastructure, these costs are diluted between all the executions. Other cloud platforms, like Microsoft Azure[1], allow per minute billing. This advanced billing model can even better fit our model.

The total execution time depends on two different factors: time spent in computation (t_{CPU}) and time spent during I/O operations ($t_{I/O}$):

$$t_{ex} = t_{CPU} + t_{I/O} \tag{6}$$

Both CPU and I/O times are affected by the characteristics of the infrastructure used during the execution of the application. t_{CPU} will be reduced depending on the number of compute instances used during the execution of the application:

$$t_{CPU} = \frac{\sum_{i \in T} t_i}{n(C)} \tag{7}$$

T is the task set of the application, while C is the set of VM instances deployed during the execution of the application for computation purposes. t_i represents the execution time of each task of the workflow and $n(C)$ is the number of VM machines deployed. The second simplification of our model consists on considering all the tasks as perfectly scalable and as executable by any instance (without taking into account dependencies), fully utilizing all the available resources, resulting in a perfect distribution of the load where the total execution time is divided by the number of VM instances. Our model suppose homogeneous VM instances where the CPU load has been previously profiled in order to measure the CPU time required by each task on this specific VM instance.

Finally, the I/O time ($t_{I/O}$) is calculated taking into account both the I/O characteristics of the application and the infrastructure used. The performance achieved for read and write operations using the Amazon S3 service greatly vary, leading to the distinction in the following equation:

$$t_{I/O} = \frac{\sum_{i \in W} FSize_i}{n(C) \cdot BW_{write}} + \frac{\sum_{j \in R} FSize_j}{n(C) \cdot BW_{read}} \tag{8}$$

[1] http://azure.microsoft.com/en-us/pricing/.

where W and R represent the sets of write operations and read operations in the application, while $FSize$ represents the size of these operations in MB. $n(C)$ represents the number of VM instances deployed and affects the total available bandwidth of I/O operations (BW is the bandwidth perceived by each node for I/O operations in MB/s). The way of considering the total available bandwidth introduces the third simplification of our model, which is the perfect scalability of the I/O operations, without taking into account network congestion and I/O contention. If one VM instance requires 10 s to write 10 files containing 100 MB of data each (1 GB total), two virtual instances will ideally perform the same operations in 5 s. Again, the simplification excludes any kind of data dependencies, dividing the total I/O work between the available compute VM instances, in a perfectly balanced scenario.

Given the fact that Hercules only affects I/O operations performed over temporary data, it is necessary to decompose the previous I/O time in three different factors:

$$t_{I/O} = t_{I/O_{input}} + t_{I/O_{tmp}} + t_{I/O_{output}} \tag{9}$$

The time needed for reading the input files of the application from Amazon S3 ($t_{I/O_{input}}$) and the time needed to write the results to persistent storage ($t_{I/O_{output}}$), is the same in the S3-only cases and the cases where Hercules is present. It can be modeled as:

$$t_{I/O_{input}} = \frac{\sum_{i \in IN} FSize_i}{n(C) \cdot BW_{read}} \tag{10}$$

$$t_{I/O_{output}} = \frac{\sum_{i \in OUT} FSize_i}{n(C) \cdot BW_{write}} \tag{11}$$

where IN is the set of read operations performed over input files and OUT is the set of write operations performed over result files during the execution of the application. The time needed for executing the I/O operations over temporary files ($t_{I/O_{tmp}}$) is modeled differently for S3 ($t_{I/O_{tmp}}(S3)$) and Hercules ($t_{I/O_{tmp}}(HER)$), as detailed in the two following equations:

$$t_{I/O_{tmp}}(S3) = \frac{\sum_{i \in TW} FSize_i}{n(C) \cdot BW_{write_{S3}}} + \frac{\sum_{j \in TR} FSize_j}{n(C) \cdot BW_{read_{S3}}} \tag{12}$$

$$t_{I/O_{tmp}}(HER) = \frac{\sum_{i \in TW} FSize_i}{MIN_{VM} \cdot BW_{write_{HER}}} + \frac{\sum_{j \in TR} FSize_j}{MIN_{VM} \cdot BW_{read_{HER}}} \tag{13}$$

$$MIN_{VM} = min(n(C), n(H)) \tag{14}$$

where W and R represent the sets of write operations and read operations performed over temporary files during the execution of the application and $n(H)$ represents the number of VM instances deployed for the Hercules infrastructure. The total available bandwidth over files stored in Hercules depends, not only

on the number of compute VM instances, but also on the number of Hercules I/O nodes available. We have selected the minimum of both values, denoted as MIN_{VM}, because it will be the limiting factor in the maximum available bandwidth. As example, in the case of a low number of compute nodes using a large Hercules infrastructure, the limiting factor will be the bandwidth available at client side: two compute instances will perform I/O operations in half the time required by one computing VM. However, this assumption is only true when the I/O nodes outnumber the computing infrastructure. In case of a lesser number of I/O nodes, the limiting factor will be the available bandwidth exposed by the Hercules infrastructure, i.e. four compute instances accessing concurrently to only one Hercules node, will share the maximum possible bandwidth offered by this node. This model is consistent with the results shown in the experimental evaluation of our solution. The rest of variables remain as described for Eq. 8.

As a summary, there are three main differences presented by the use of Amazon S3-only solutions in contrast with a hybrid solution using Amazon S3 for I/O files and Hercules for any I/O operation performed over temporary files. First, Hercules requires the deployment of a greater number of VM instances (or VMs with more RAM when deployed sharing resources with the compute nodes), incurring in a greater computation costs (pay-per-use of VM instances). Second, Hercules deployment can result in time reductions due to the acceleration of I/O operations, obtaining a reduction of the total execution time of the application, potentially lowering the costs related to the computing infrastructure (VM instances). Third, through the use of Hercules instead of Amazon S3 for I/O operations performed over temporary files, the cost of using the storage service can be lowered, which is especially important in the targeted data-intensive applications.

The next section presents the evaluation of a use case application, consistent with the data intensive target, in order to analyze this balance in different scenarios.

5 Costs Analysis of a Data-Intensive Application

In order to show the usefulness of our proposed model, we are going to define a data-intensive application with realistic characteristics. This costs analysis will show the budget impact of I/O-related operations in the execution of a data intensive application in a public cloud platform, as well as presenting the execution costs of the application performing every I/O operation over Amazon S3 in comparison with performing I/O operations over temporary data stored in Hercules.

5.1 Application Description

As a study case, we have used an I/O intensive workflow application where both computation and I/O times are balanced. Figure 2 depicts the workflow phases, where an image file is read by the *filter task*, creating three new image files. The

filtering task can be any kind of lightweight image-processing computation, such as applying three different filters and decomposing the image in RGB colors. These new image files are afterwards read by the *combine task*, combining the images or selecting one of the images (this combination/selection can be based on any criteria: quality, randomness, patterns, etc.) writing a final image file as a result of the workflow. Every image in the workflow has roughly the same size and any number of images can be used as input files.

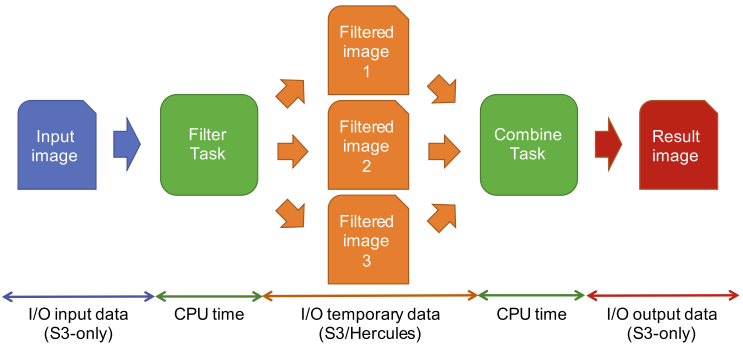

Fig. 2. Image processing data-intensive workflow used as study case for analyzing the costs derived of the deployment of Hercules over a cloud infrastructure. (Color figure online)

When Hercules is deployed and temporary files are stored in Hercules I/O nodes. In our study case, the images created by the *filter task* and read by the *combine task* (depicted in orange) are temporary files. Input and resulting files are stored in the Amazon S3 storage service in every evaluated case for durability reasons. The selected Hercules deployment consists of sharing resources with compute nodes, while deploying VM instances with a greater amount of RAM for the in-memory storage of data. Based on that logic, in the Amazon S3-only case we have deployed *m4.xlarge* VM instances (4 cores, 16 GB RAM, 0.264 USD/hour) and *r3.xlarge* memory optimized VM instances for the Hercules case, which are equivalent VM instances with more RAM (4 cores, 31.5 GB RAM, 0.371 USD/hour). In the table, the cost of VM instances for the Hercules infrastructure appears as *+0.107 USD/hour*, showing the price difference of the additional main memory required. Additionally, *r3.xlarge* come with 80 GB SSD space, which can be utilized for data.

Tables 1, 2, and 3 present the specific configuration of every configurable variable of our model, describing the characteristics of the application in Table 1, the characteristics of the architecture in Table 2, and the billing concepts and prices applied based on Amazon AWS costs in Table 3. Every variable representing costs is presented as provided by the cloud operator (Amazon) and normalized to our model when necessary. Amazon S3 and Hercules bandwidths are based on previous works [6].

Table 1. Input parameters for the costs analysis of the study case: Characteristics of the application.

Parameter	Value
Total input files	8,192
Size of input files	128 MB
Total size of input files	1 TB (8,192 * 128 MB)
Generated temporary files	24,576 (3 * 8,192)
File size of temporary files	128 MB
Total size of temporary files	3 TB (128 MB * 24,576)
Generated result files	8,192
File size of result files	128 MB
Total file size of result files	1 TB (128 MB * 8,192)
No. GET operations	32,768 (4 * 8,192)
No. PUT operations	32,768 (4 * 8,192)
Filter task CPU time	20 s
Combine task CPU time	10 s

Table 2. Input parameters for the costs analysis of the study case: Characteristics of the infrastructure running the application.

Parameter	Value
No. compute VM instances (C)	32
No. Hercules I/O nodes (H)	**4 to 32**
$BW_{read_{S3}}$	90 MB/s[a]
$BW_{write_{S3}}$	20 MB/s[a]
$BW_{read_{HER}}$	90 MB/s[a]
$BW_{write_{HER}}$	90 MB/s[a]

[a]Based on our previous work [6].

We have selected 30 s of computation time for balancing the I/O-to-computation ratio. Based on an image file size of 128 MB, reading the image from Amazon S3 at around 90 MB/s implies ∼1.4 s while writing the same image at around 20 MB/s is translated in ∼6.4 s. The workflow consists of a total of five image files, with one read and one write operation per file (with the exception of input and output files which are only read or written, not both), up to a total of 4 read operations and 4 write operations, resulting in ∼31.2 s. CPU time can be distributed as 20 s for the *filter task* and 10 s for the *combine task*. Figure 3 shows the details of this data intensive application following the workflow model presented in Fig. 1. As can be seen in Table 1, in this configuration the application processes 8,192 images, leading to 8,192 executions of the workflow that can be carried out in parallel in different computing resources with a proper scheduler.

Table 3. Input parameters for the costs analysis of the study case: billing concepts and prices in the Amazon AWS platform.

Parameter	Value
Storage cost	0.0300 USD per GB[a]
Normalized storage cost	1.13e−11 USD per MB per sec
Total CPU time	245,760 s (30 * 8,192 s)
Compute VM instances cost	0.239 USD/hour per node[b]
Normalized comp. VM instances cost	0.000066 USD/sec per node
Hercules I/O nodes cost	+0.107 USD/hour per node[b]
Normalized Hercules I/O nodes cost	0.000029 USD/sec per node
GET operations cost	0.004 USD per 10,000 op.[a]
Normalized GET operations cost	0.0000004 USD
PUT operations cost	0.005 USD per 1,000 op.[a]
Normalized PUT operations cost	0.000005 USD

[a]Based on Amazon S3 prices https://aws.amazon.com/s3/pricing/.
[b]Based on Amazon AWS prices for Amazon EC2 m4.xlarge and r3.xlarge instances https://aws.amazon.com/ec2/pricing/.

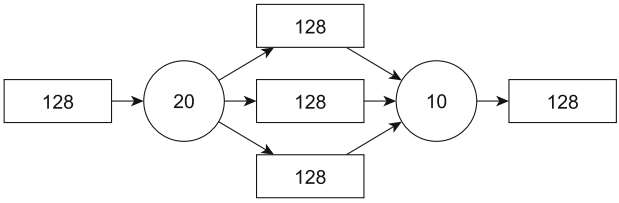

Fig. 3. Model of the image processing data-intensive workflow used as study case, including tasks, CPU cost (CPU time), I/O operations, and I/O cost (file size).

5.2 Costs Analysis

Figure 4 plots the breakdown of the total execution time of the experiment over different I/O infrastructures. *S3* case represents executions where every I/O operation is performed over the Amazon S3 storage service, while every other case rely on different Hercules deployments for temporary data accesses (using 4, 8, 16 and 32 I/O nodes deployed sharing resources with the compute nodes). The black line represents the total cost of the execution of this workflow, based on the previously presented costs model. Figure 4 clearly shows how the flexibility of Hercules can be used for finding a trade-off between cost and execution time. Using 4 I/O nodes, Hercules presents a poor execution time compared with the *S3* case. However, as the number of I/O nodes increases, the total execution time is reduced. This behavior is produced by the increased performance of I/O operations performed over temporary data, using Hercules as I/O accelerator. Every other phase of the workflow execution time remains constant for every

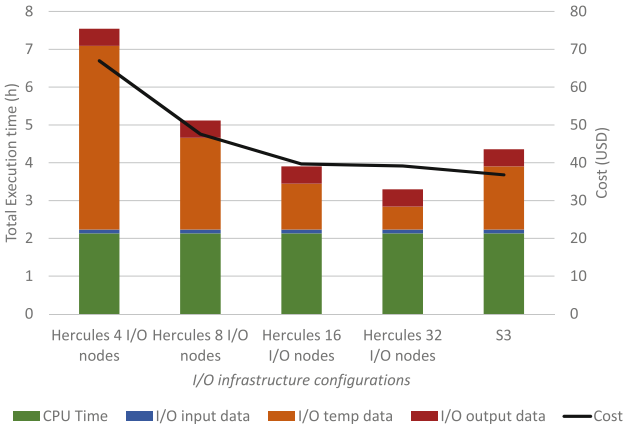

Fig. 4. Breakdown of the total execution time comparing the use of the cloud storage service (S3) for every I/O operation with the deployment of different Hercules configurations for temporary data. The black line represents the total execution cost (secondary Y axis). Computing infrastructure is 32 VM instances for every case.

experiment (including the S3 case): reading the input files, writing the results, and computation time.

The trend shown by the costs line seems counter-intuitive for two main reasons. First, total execution costs are similar in some Hercules and S3-only cases, which seems incorrect given the additional resources needed for the Hercules I/O infrastructure. Second, Hercules costs are reduced at the same pace as more Hercules I/O nodes are deployed, which again seems unrealistic given the fact that costlier VMs are necessary. Figure 5 presents a breakdown of the execution costs, detailing the cost related with three different billing concepts: S3-related costs (storage and PUT/GET operations), the costs of the VM instances deployed as computing resources. and the cost of VM instances running as Hercules backends. The combination of Figs. 4 and 5 shows how the usage of Hercules for temporary data both reduces the total execution time and reduces the amount of data stored over Amazon S3. On the one hand, the reduction of the total execution time affects the amount of time where VM instances are deployed, reducing the costs related with computation and I/O instances. On the other hand, the reduction of data stored over S3 minimizes the costs related with the use of the S3 API, both in persistent data storage and PUT/GET operations costs.

It is also interesting to highlight how the performance scales as the price is reduced in Hercules. The flexibility in the deployment of the Hercules infrastructure offers to the users the ability of trading-off execution time and cost efficiency, depending on their necessities. The specific characteristics of the application or the cloud provider used may vary these results, but we consider the study case presented as a fair example of data-intensive application (balanced CPU and

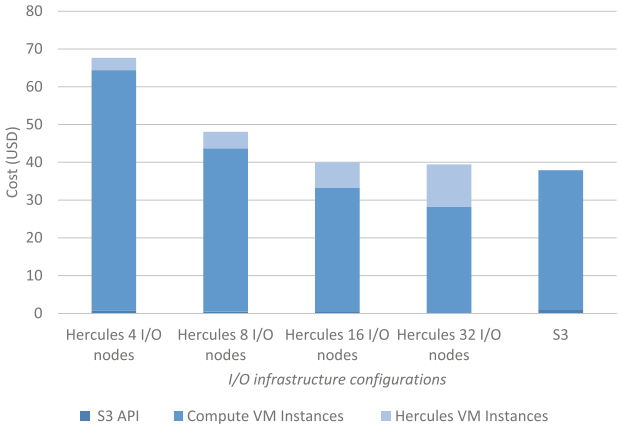

Fig. 5. Breakdown of the total execution cost comparing the use of Amazon S3 for every I/O operation with the deployment of different Hercules configurations for temporary data. Computing infrastructure is 32 VM instances for every case.

I/O time, large amount of temporary data produced) and cloud provider (being Amazon AWS one of the most used IaaS cloud providers).

Our cost analysis shows how data-intensive applications can be benefited by the deployment of Hercules, resulting in reductions in total execution time with a comparable cost. Depending on the execution time reductions achieved by applying Hercules for temporary I/O operations, it could be possible to even obtain cost reductions in applications with a great amount of temporary data in comparison with Amazon S3. Total execution time is reduced due to the increased I/O performance offered by our proposed I/O accelerator. In the costs reduction side, on the one hand, the additional costs related with the deployment of additional or costlier VM instances for the Hercules I/O infrastructure can be compensated with a reduction in total execution time (less total execution time is translated in less time using the deployed VM instances). On the other hand, the costs of storing and accessing temporary data in a persistence-oriented service like Amazon S3 can be avoided by using Hercules I/O nodes.

6 Conclusions

This work has presented a model for calculating potential costs derived from the deployment of data-intensive applications over IaaS cloud platforms. This model takes especially into account the costs related with I/O operations, including the impact of deploying our proposed in-memory I/O accelerator (Hercules) as an alternative to default cloud storage services. Additionally, we have applied the proposed model to a data-intensive image processing application, comparing the costs of execution performing every I/O operation over the default cloud storage service in contrast with deploying Hercules for temporary data.

We can conclude that the performance of data-intensive applications with a large amount of temporary data can be improved while maintaining the execution costs. The main benefit offered by our solution for future Ultrascale systems is the flexibility in configuration, targeting different objectives depending on the requirements of the application and the available budget. The user choose to save money or save time in comparison with the default cloud storage service, even beating both price and performance in balanced configurations.

In the future we should focus on the extension of the costs model for taking into account data locality issues, which should expose even better performance and costs in Hercules cases. Additionally, the model can be applied to other IaaS public cloud providers like Microsoft Azure.

References

1. Deelman, E.: Pegasus, a workflow management system for science automation. Future Gen. Comp. Syst. **46**, 17–35 (2015)
2. Carretero, J., et al.: Memorandum of understanding. In: Network for Sustainable Ultrascale Computing (NESUS), p. 30 (2014). http://www.nesus.eu
3. Chiu, D., Agrawal, G.: Evaluating caching and storage options on the Amazon Web Services Cloud. In: 11th IEEE/ACM International Conference on Grid Computing, pp. 17–24 (2010)
4. Duran, A., Ayguade, E., Badia, R.M., Labarta, J., Martinell, L., Martorell, X.: OmpSs: a proposal for programming heterogeneous multi-core architectures. Parallel Process. Lett. **21**(02), 173–193 (2011)
5. Duro, F.R., Blas, J.G., Isaila, F., Wozniak, J.M., Carretero, J., Ross, R.: Flexible data-aware scheduling for workflows over an in-memory object store. In: CCGRID 2016, pp. 321–324, May 2016
6. Duro, F.R., Garcia-Blas, J., Isaila, F., Carretero, J.: Experimental evaluation of a flexible I/O architecture for accelerating workflow engines in cloud environments. In: DISCS 2015, pp. 6:1–6:8 (2015)
7. Li, H., Ghodsi, A., Zaharia, M., Shenker, S., Stoica, I.: Tachyon: Reliable, memory speed storage for cluster computing frameworks. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 1–15. ACM (2014)
8. Marozzo, F., Talia, D., Trunfio, P.: JS4Cloud: script-based workflow programming for scalable data analysis on cloud platforms. Concurrency Comput. Pract. Experience **27**(17), 5214–5237 (2015)
9. Rodrigo Duro, F., Marozzo, F., Garcia Blas, J., Talia, D., Trunfio, P.: Exploiting in-memory storage for improving workflow executions in cloud platforms. J. Supercomputing **72**(11), 4069–4088 (2016)
10. Yuan, D., Yang, Y., Liu, X., Chen, J.: A cost-effective strategy for intermediate data storage in scientific cloud workflow systems. In: IPDPS 2010, pp. 1–12 (2010)
11. Yuan, D., Yang, Y., Liu, X., Chen, J.: On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems. J. Parallel Distrib. Comput. **71**(2), 316–332 (2011)