2016 IEEE International Parallel and Distributed Processing Symposium Workshops

Performance Analysis of an I/O-Intensive Workflow executing on Google Cloud and Amazon Web Services

Hassan Nawaz, Gideon Juve, Rafael Ferreira da Silva, Ewa Deelman

University of Southern California, Information Sciences Institute, Marina Del Rey, CA, USA
{*hassan,gideon,rafsilva,deelman*}*@isi.edu*

***Abstract*—Scientific workflows have become the mainstream to conduct large-scale scientific research. In the meantime, cloud computing has emerged as an alternative computing paradigm. In this paper, we conduct an analysis of the performance of an I/O-intensive real scientific workflow on cloud environments using makespan (the turnaround time for a workflow to complete its execution) as the key performance metric. In particular, we assess the impact of varying the storage configurations on workflow performance when executing on Google Cloud and Amazon Web Services. We aim to understand the performance bottlenecks of the popular cloud-based execution environments. Experimental results show significant differences in application performance for different configurations. They also reveal that Amazon Web Services outperforms Google Cloud with equivalent application and system configurations. We then investigate the root cause of these results using provenance data and by benchmarking disk and network I/O on both infrastructures. Lastly, we also suggest modifications in the standard cloud storage APIs, which will reduce the makespan for I/O-intensive workflows.**

***Keywords*-Scientific Workflow; Cloud Computing; I/O Performance Modeling**

I. Introduction

Scientific workflows have been extensively used by scientists to perform complex simulations and process large amounts of data [1]. Traditionally, scientific workflows are executed on campus clusters and national cyberinfrastructures systems. However, the emergence of cloud computing has opened a new avenue for scientists [2]. The benefits of running workflows on these environments include predictable performance, quality of service, on-demand resource provisioning, the ability to store virtual machines (VMs) in the form of virtual images, and resource monitoring. In addition, workflow developers can have full control of the execution environment, something that is typically limited in traditional systems (and may impose difficulties for the workflow execution). On the other hand, cloud systems were not designed for the execution of complex simulations and I/O-intense applications. Furthermore, computational resources may not be free (in case of commercial clouds). Therefore, there is an incentive for researchers to explore avenues to reduce the cost of executing workflows, while increasing their efficiency. Extensive work has been performed [2]–[4] to understand the efficiency of cloud environment for scientific workflows. However, most of these works focus on performance (in terms of processing speed) and monetary optimizations.

In this paper, we assess the performance of an I/O-intensive scientific workflow on two widely used commercial cloud environments: Google Compute Engine [5] and Amazon Web Services [6]. We initially focus on the performance difference when storage configurations are varied to quantify the impact of storage bottlenecks. To this end, we use the Montage workflow [7], a well-known astronomy application, as a benchmark to quantify application performance. The Montage workflow is composed of thousands of computing jobs and manages over 20,000 data transfers. We execute instances of Montage on different storage deployment configurations in both cloud systems. We then collect performance metrics (e.g., makespan—workflow turnaround time) to compare the efficiency of these systems. This comparison unveils significant performance differences among configurations revealing the impact of the bottlenecks in the storage configuration. We also notice a remarkable difference between the application's performance on both cloud systems despite the similarity of the execution environments (in terms of VM types and software) and configurations used. We then investigate the performance of these systems and available data transfer tools by benchmarking network and disk I/O. Finally, we discuss opportunities to improve current cloud APIs, which can lead to significant impact on the workflow performance.

The main contributions of this paper include:

1) An evaluation of the impact of varying storage configurations on the performance of an I/O-intensive workflow;
2) A quantitative analysis of application performance on popular cloud systems using provenance data;
3) A comprehensive analysis of benchmarking file transfer times of different sizes using different cloud tools;
4) A discussion on indicators that would significantly improve the performance of I/O-intensive workflows on cloud environments.

This paper is organized as follows. Section II describes the Montage workflow, and the execution environment for the experiments. Section III presents the performance analysis of the I/O-intensive workflow on both cloud environments

978-1-5090-3682-0/16 $31.00 © 2016 IEEE
DOI 10.1109/IPDPSW.2016.90

under different storage configurations. Section IV presents an exploration of performance discrepancies, using benchmarking of data transfer times, for various file sizes using standard cloud transfer tools. The efficiency of multi-threaded transfers are evaluated in Section V. Section VI discusses related work, and Section VII summarizes our findings and identifies future work.

II. Experiment Conditions

In this section, we introduce the I/O-intensive workflow application and its main characteristics, the workflow management system, and the different storage deployment configurations.

A. Scientific Application

Montage [7] is an astronomy application that creates astronomical image mosaics using data collected from telescopes. The workflow (Figure 1) can be set up with different sizes depending upon the area of the sky (in square degrees) covered by the output mosaic. In this paper, we used Montage to generate an 8-degree square mosaic. The resulting workflow is composed of 10,429 jobs, which reads 4.2 GB of input data, and produces 7.9 GB of output data (excluding temporary data). A workflow instance operates over about 23K intermediate files, where most of them have a few MBs. Figure 2 shows the distribution of intermediate file sizes produced during the workflow execution. The write pattern is sequential and each file is written once only and never modified later. The read pattern is mostly sequential with a few jobs accessing files at random locations. We consider Montage to be I/O-bound because it spends more than 95% of its time waiting on I/O operations. Note that the `mConcatFit` job (Figure 1) has several incoming edges, and all subsequent jobs depend on it. In the workflow instance used in this paper, `mConcatFit` transfers 6,173 small files with average size of 0.3 KB as input (a total of 1.9 MB). Therefore, a poor execution performance of this job (in particular for data movement) may significantly impact the workflow makespan. A detailed characterization of the Montage workflow can be found in [8], [9].

B. Execution Environment

Workflow runs use the Pegasus workflow management system [10]. Pegasus automates the task of mapping, clustering, scheduling, and executing computing jobs in a wide range of execution environments. Typically, scientific workflows are composed of multiple layers of jobs where each layer performs specific computation using data from previous layers. In cloud deployments, this data is written as a file to a shared scratch directory by a parent job, and is consumed by a child job for its computation. These files are called *intermediate data files*.

The execution environment consists of a submit host (located at our lab at the USC Information Sciences Institute,

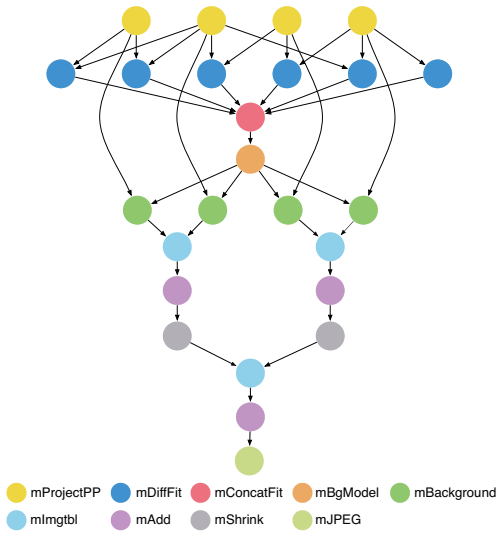


Figure 1. An illustrative representation of a Montage workflow.

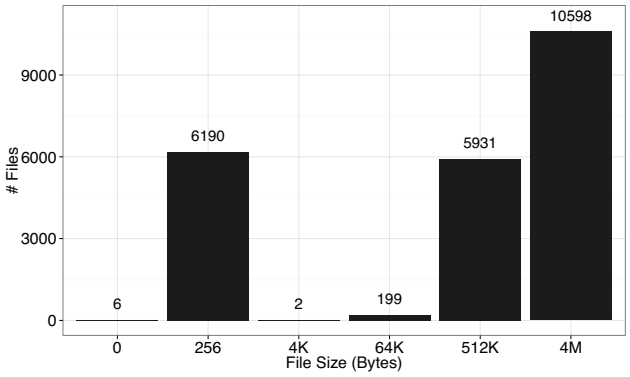


Figure 2. Distribution of intermediate file sizes for one instance of the Montage workflow.

Los Angeles, CA) which plans and submits the workflow for execution. The jobs are executed inside the cloud on VMs. The experiments used VMs that are meant for jobs with both memory and computation requirements. For Amazon, `m3.2xlarge` VM instance types were used, while `n1-standard-8` instances were used for Google. Both VM types provide 8 cores per node and 30 GB of memory. The experiments conducted in this paper were limited to one VM on each cloud.

In Pegasus, data movement in cloud environments is performed through standard cloud tools. For Google, we use the `gsutil` [11] client, and for Amazon we use the `pegasus-s3` [12] client, which is built on top of standard Amazon APIs. Amazon also provides its own client (`aws-cli` [13]), however it is not the standard tool used in Pegasus. In Section IV, we evaluate the performance of each of these tools.

In order to measure the actual overhead involved on data transfers, we limit the transfer mechanisms to a single-threaded mode. In Section V, we conduct a multi-threaded experiment to demonstrate the performance gain of transferring data in parallel.

C. Storage Configuration Deployments

In order to quantify the performance overhead of the I/O-intensive workflow on cloud environments, we conducted workflow runs in three distinct real production scenarios where the storage configuration varies:

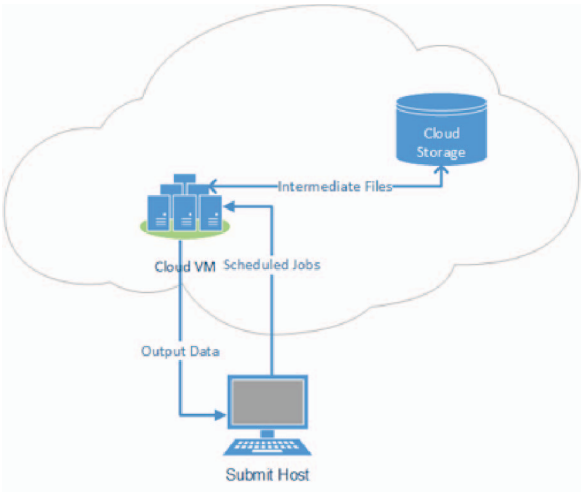
Cloud storage. In this configuration, intermediate files are stored into object storage (Figure 3a). In particular, we use the Amazon S3 storage [14], and Google Cloud Storage [15] for Amazon and Google VMs respectively. This configuration is expected to be the most commonly used in cloud environments due to its simplicity (e.g., there is no need for a shared file system) and to the ability to permanently store intermediate results (e.g., workflow steering). However, storing all intermediate files in a storage service may be costly.

VM storage. In this configuration, intermediate files are stored locally to disks attached to the VM (Figure 3b). For the experiments, each VM had a 70 GB SSD drive available for storage. Although this configuration may reduce the monetary cost, it may not be scalable for very large workflow executions (e.g., each VM will process thousands of jobs, and as a result the machine may run out of disk space [16]). However, this configuration helps quantify the overhead of other storage configurations.

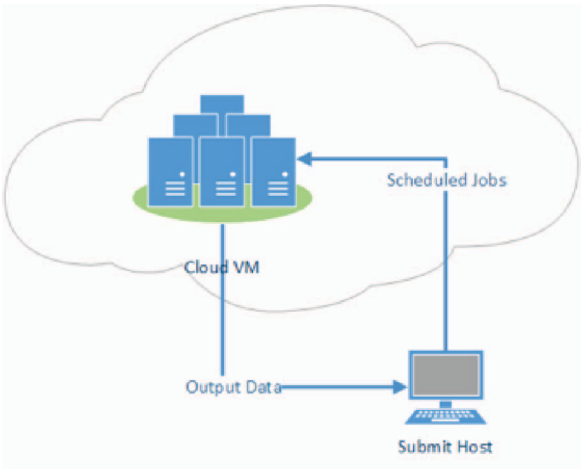
Submit host. In this configuration, intermediate data is stored at the submit host (Figure 3c). It is unlikely this configuration will be used in real production workflows due to the high latency in transferring data to the submit host (which is often outside the cloud network). However, this configuration may be useful in low cost scenarios, or when local data analyses should be performed in the intermediate results. In the experiments, we use the standard linux SCP [17] command to perform transfers between the submit host and the VMs. Note that other standard protocols (e.g., SFTP [18]) could also have been used, since the purpose is only to create a homogenous environment for both clouds for comparisons.

III. Overall Performance Evaluation of an I/O-intensive Workflow

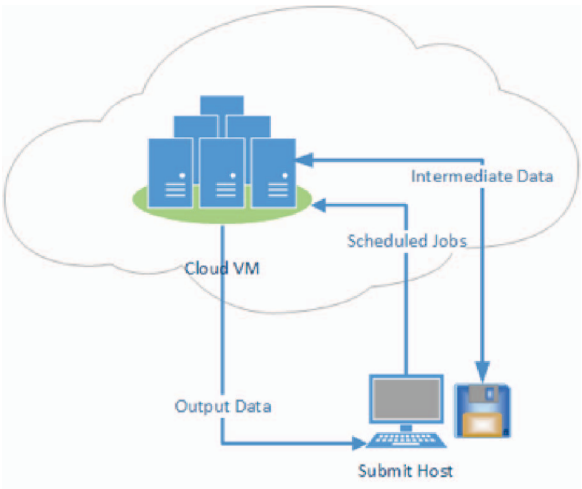
For this experiment, we conducted three runs of the Montage workflow for each configuration on both cloud infrastructures to quantify the overhead of data transfers. Figure 4 shows the average makespan for the runs on both Amazon and Google. Not surprisingly, the *VM storage* configuration outperforms all other configurations due to the absence of transferring intermediate files. When the



(a) Using cloud storage for intermediate storage.



(b) Using VM Storage for intermediate files.



(c) Using submit host for intermediate files.

Figure 3. Different cloud storage configuration deployments evaluated in this work.

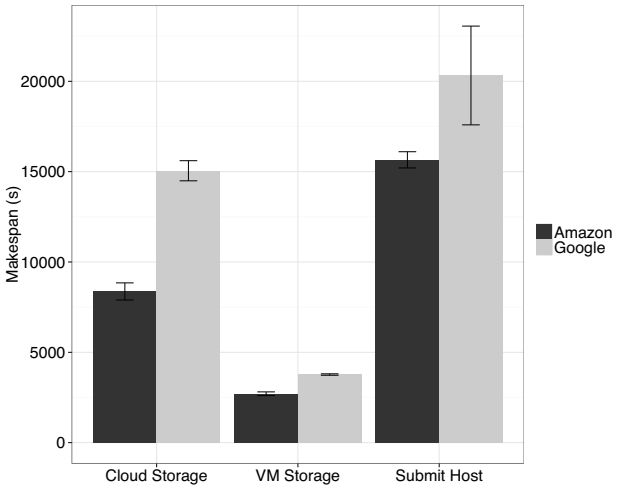
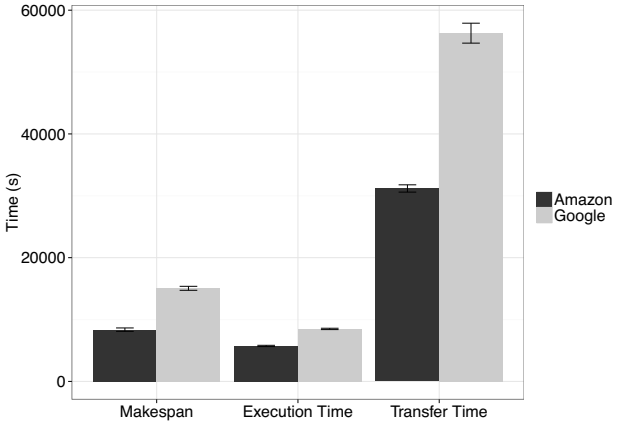


Figure 4. Average makespan values for 3 runs of the Montage Workflow for different storage configurations on Amazon and Google cloud environments.

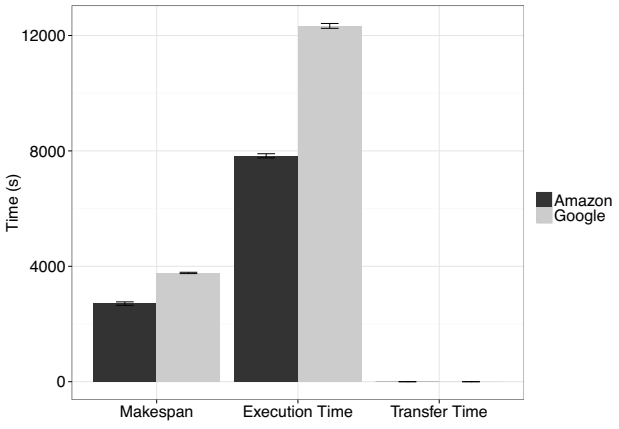
intermediate data is stored within the cloud, i.e., in an object storage, the network overhead is significantly smaller when compared to external transfers to the submit host. In both clouds, the performance gain on storing data locally is up to about 400% when compared to the *Cloud Storage* configuration, and up to about 580% in relation to the *Submit Host* configuration. Several aspects may influence the performance of a workflow execution, including data staging time overhead, disk I/O overhead for intermediate files, a difference in computational power, and external loads. However, since we only vary storage configurations, we argue that these differences are mostly related to overhead caused by data movements.

Figure 5 shows the amount of time spent on jobs execution and files transfer on both clouds and for all storage configurations, reconstructed by provenance information collected by Pegasus. Execution and transfer times are computed as the sequential cumulative execution time for individual jobs, and data transfer operations respectively. Note that these sequential execution times may represent larger values when compared to the workflow turnaround time, since the makespan measures the difference between the earliest start time and the latest finish time. In both *Cloud Storage* (Figure 5a) and *Submit Host* (Figure 5c) configurations, the execution time is substantially smaller than the workflow makespan which confirms that data transfer operations become a bottleneck in the workflow execution. In addition, measurements shown in Figure 5 also quantify performance differences when using a remote storage (either in an object storage or on the submit host).

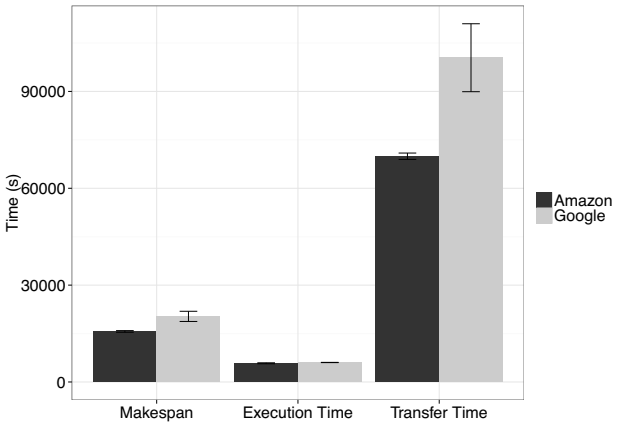
In contrast, jobs execution time prevails in the *VM storage* configuration (Figure 5b) as there are no data transfers. The execution times for the later configuration is larger than for



(a) Cloud Storage



(b) VM Storage



(c) Submit Host

Figure 5. Average makespan, cumulative job execution time, and intermediate data file transfer time per configuration.

the other configurations. This performance loss is mostly due to the execution of data movement operations to store intermediate files locally. The intermediate files produced by the parent jobs are written to the SSD drive attached to

the VM. The child jobs then read these intermediate files from the filesystem, and do not have to incur any data transfer overhead. However, the number of I/O operations is significantly increased. As a result, jobs spend more time waiting for I/O operations to complete.

In the case of *VM Storage*, the only data transfers are data staging jobs (transferring of input/output data from/to the submit host). Average data staging times for this configuration are 2,680s for Google runs, and 2,262s for workflows executed on Amazon. This difference in data staging times indicates a difference of performance in the two systems.

Note that there is a consistent difference between the workflow makespan values obtained from the execution on both clouds in all storage configurations (Figures 4 and 5). The performance measures obtained from workflow runs on Amazon are up to 44% faster than runs conducted on Google. This discrepancy is visible in the *Cloud Storage* (Figure 5a) and *Submit Host* (Figure 5c) configurations, where intermediate transfer times are significantly impacted on Google runs. For instance, for the *Cloud Storage* configuration the cumulative average transfer time for Google is 56,287s, and for Amazon is 31,186s. This result indicates that data transfer strongly impacts the workflow makespan. In such configurations, this difference is mostly due to (1) poor performance of the storage system (e.g., limited bandwidth or high load), or (2) performance issues with the transfer tools. In the next section, we analyze the performance of the transfer tools and the systems through benchmarking. The difference in the execution times for the *VM Storage* configuration (Figure 5b) is mostly due to the performance of the SSD disks as aforementioned. In the next section, we also investigate whether the performance of the attached storage driver degrades the workflow makespan.

As presented in Section II-A, the `mConcatFit` job of the Montage workflow may significantly impact the workflow makespan if its performance is very low. For instance, the time required to stage in data for this job on the *Cloud Storage* configuration is 1,862s for Amazon, and for Google is 4,387s, which represents 22% and 29% of the workflow makespan respectively. We investigate the cause of this massive difference in the analyses presented in the following section.

IV. Benchmarking Storage Performance

In this section, we investigate the causes of different cloud performance by benchmarking network and disk throughput.

A. Network I/O

Experiments conducted in this subsection use the *Cloud Storage* configuration, where intermediate files are stored into an object storage. We downloaded and uploaded files of various sizes from Amazon and Google VMs to their respective cloud storage every hour for a week (from May 12. 2015 to May 18, 2015). Figure 6 shows the time series

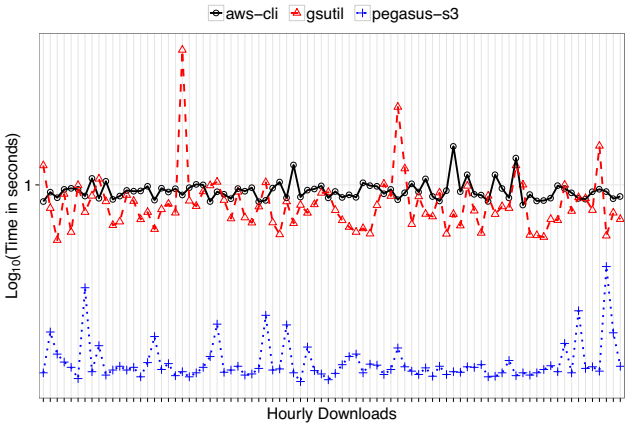


Figure 6. Time series download times of an empty file (0 Bytes) from an object storage to the VM for Amazon and Google clouds. The values were measured hourly during the time period of a week (May 12, 2015 to May 18, 2015).

of the download times for an empty file (0 Bytes) from the cloud object storage to the virtual machine. The goal in transferring an empty file is to measure the overhead induced by the system. Since `mConcatFit` stages in over 6K small files with average size of 0.3 KB, the performance of these operations are of utmost importance. The large overhead for small file transfers appears as the major cause for the low performance of the `mConcatFit` job, and hence, the workflow. For Amazon, we collected measurements from transfers performed with the `pegasus-s3` client, as well as the *aws-cli* tool (which is the standard Amazon client). For Google, we used their standard command line tool: `gsutil`. Intriguingly, `pegasus-s3` outperforms both standard tools provided by the cloud environments. Nevertheless, Amazon and Google standard tools perform similarly. Note that due to the dynamic nature of the system measurements may vary due to, for example, network contention or internal load balancing. However, such variations are not sufficient to mask the large performance difference between the tools.

Figure 7 shows the average upload (top) and download (bottom) times of different file sizes for the transfer tools. For upload operations (Figure 7-top), `pegasus-s3` has better performance for small file sizes (less or equal to 10MB), while the performance difference is mitigated for larger files (100MB and 1GB). Similar behavior is observed for download operations (Figure 7-bottom), except that `pegasus-s3` yields better performance up to 100MB files.

In order to identify the reasons why `gsutil` and `aws-cli` yield poor performances, we ran the transfer tools in the debug mode and evaluated all request operations performed by each tool. From this analysis, we noticed that all tools generate a `HEAD` and/or a `GET` request. We then used `tcpdump` [19] and `wireshark` [20] to trace TCP packets (Figure 8). The Amazon client (`aws-cli`) uses two TCP

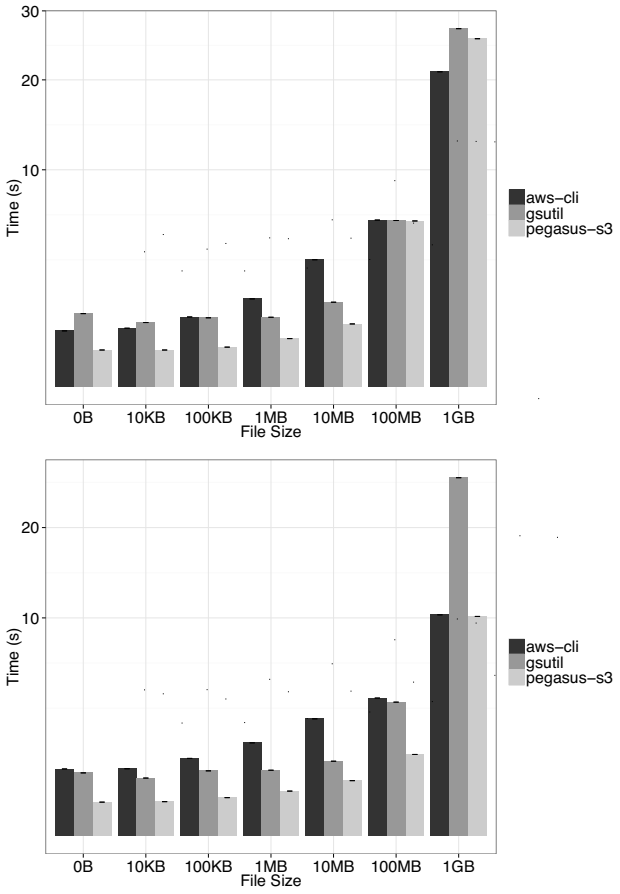


Figure 7. Average upload (top) and download (bottom) times of a file size transferred every hour for a week between a VM and a Cloud Storage.

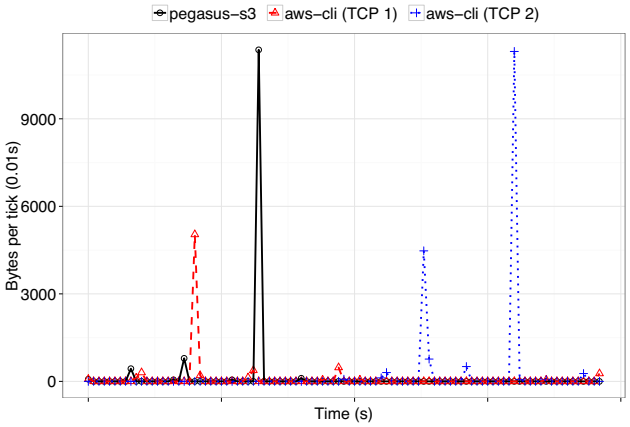


Figure 8. Bytes per 0.01s transferred per TCP connection. It shows that `aws-cli` creates two sequential TCP connections, while `pegasus-s3` reuses the same connection. The ∼1200 peaks represent the `GET` call to the object storage (Amazon S3), and the ∼4000 peaks represent the TLS overhead.

connections to perform a *copy* command, as opposed to only one connection used by `pegasus-s3`. This leads to the overhead of a TCP connection establishment and termination once the `HEAD` request returns. Additionally, `aws-cli` uses `https` over all operations, which generates an additional overhead of Transport Level Security (TLS) [21] for each of the connections. `pegasus-s3`, on the other hand, uses the same TCP connection connection to perform both `HEAD` and `GET` requests. It also allows to skip TLS by allowing operations over `http` instead of `https`, i.e., it bypasses the TCP handshake [22].

In Figure 7, for most file sizes `pegasus-s3` performs significantly better than `gsutil`. We conducted the same analysis as for `aws-cli`, and noticed that `gsutil` also establishes two TCP connections during the course of a request. However, unlike `aws-cli` it uses two `GET` requests: one to fetch the location of the data, and another to copy the data itself. This additional overhead has a significant impact on data movement operations over small files. Note that as `gsutil` performs two `GET` requests using different TCP connections, it is expected that file transfers of an empty file would take longer than with `pegasus-s3`. Analysis results shown in Figures 6 and 7 clearly demonstrate that this is the case. Since in the Montage workflow most of the jobs consume/produce small files (up to 10MB, Figure 2), we do not investigate the reasons of the poor performance of the `gsutil` tool when transferring a 1GB file. However, typical issues may include network latency and increased load, among others.

The performance differences showed by these tools explains the differences observed on transfer times, and thereby makespan, for the workflow executions performed with the *Cloud Storage* and *Submit Host* configurations shown in Figure 5.

B. SSD I/O

The analysis conducted for the *VM Storage* configuration (Figure 5b) showed a significant performance difference between the jobs execution times. As suggested, this discrepancy may be caused by the performance of the disk drive attached to the virtual machine. In this subsection, we evaluate the I/O throughput of the SSD disk used for each cloud environment through benchmarking. For Amazon, we used the Amazon General Purpose SSD disk, and for Google we used the Google Persistent SSD disk. For the benchmarking experiment, we used the linux `dd` [23] command-line tool to benchmark the read/write throughput. We executed `dd` with a block size of 4MB and one thousand blocks. We performed one hundred sequential iterations of the command on both cloud resources and measured the I/O throughput (Figure 9).

Amazon SSD provides a much higher throughput (128 MB/s) during the first 60 iterations, however the throughput drastically drops to under 9 MB/s for the remaining iterations. This reduction in the throughput is triggered by

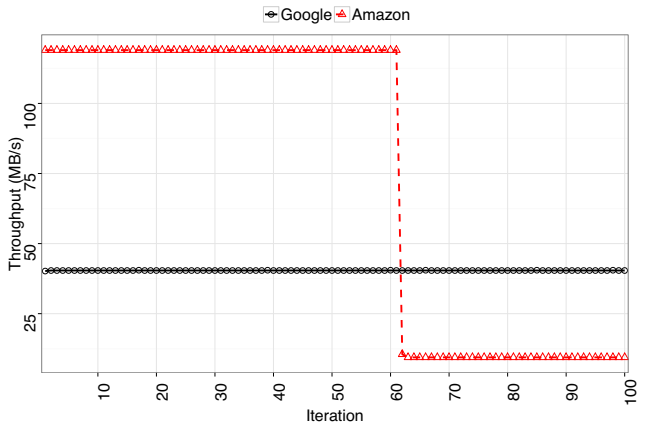


Figure 9. I/O throughput comparison between SSD disks from Amazon and Google clouds.

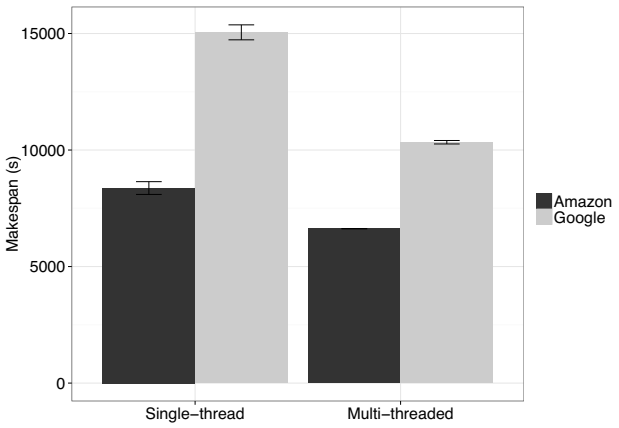


Figure 10. Average workflow makespan for 3 runs of the Montage workflow using singe- and multi-threaded mode for data transfer.

Amazon's burst tolerance policy. Amazon provides a consistent baseline throughput of 3 IOPS (Input/Output Operations Per Second) per GB and handles bursts up to 3000 IOPS per volume. These bursts are based on I/O credits for a volume. Initially, all volumes begin with sufficient credits to allow bursts of 3000 IOPS for a time period of 1,800s. This explains why the SSD I/O throughput for Amazon falls back to the baseline after the 60th iteration. Volumes earn I/O credits every second at a baseline performance rate of 3 IOPS per GB of volume size, and credits can be accumulated up to 1800 seconds of burst. For example, the 70 GB SSD attached to the VM requires 25,714 seconds to accumulate credits to completely refill the bucket of 30min burst. Since the *VM Storage* configuration writes all intermediate data to the SSD disk and the Montage workflow operates over thousands of small files, I/O credits were eventually consumed and the I/O throughput significantly dropped. This result explains why execution times for the *VM Storage* configuration are much longer. Amazon also provides a Provisioned IOPS SSD Volume, in which a user can define a Volume with a specific IOPS rate for applications that have high I/O requirements. The drawback of this solution is that the workflow execution cost may significantly increase.

Google, on the other hand, provides 30 IOPS per GB without any burst tolerance. In our experiments, the 70GB hard drive has a 33.6 MB/s sustained throughput. Although Amazon's burst policy may substantially reduce I/O throughput, it still provides a much higher overall I/O throughput for the workflow execution (this statement holds if the workflow I/O patterns are bursty enough). For this reason, Amazon yields better performance than Google for the *VM Storage* configuration, where intermediate files are stored on the attached SSD.

V. Multi-Threaded Data Transfer

The set of experiments conducted in the previous sections focused on the analysis of the impact of overheads on workflow executions. In these experiments, all data transfers where performed in a single-thread mode, which facilitates the detection and evaluation of performance issues. However, in real production executions, data transfers are often performed in a multi-threaded mode. Therefore, we performed workflow runs in a *Cloud Storage* configuration scenario where file transfers were realized with multiple threads. We chose this configuration since it has been the most popular among Pegasus' users.

First, we performed several executions of workflows using the Montage workflow. For each execution of the workflow, we gradually increased the number of threads used to execute the transfer operations. Overall, a reduction in the makespan was observed by increasing threads until the number of threads was greater than 5. Thus, we set 5 as the default number of threads per data transfer operation and conducted 3 runs of the Montage workflow on both cloud platforms. Figure 10 shows the average workflow makespan for the Montage workflow for both transfer modes. Not surprisingly, the multi-threaded mode yields smaller makespan values. In the multi-threaded mode, the makespan for Amazon is about 21% lower, while for Google the improvement is of about 32%.

Workflow runs on Google had more significant reduction in the makespan when compared to the single-threaded mode. This decrease is due to the use of multiple transfers at a time, which masks the overhead incurred in a single operation. To illustrate the impact of using a multi-threaded transfer mode on the workflow execution performance, we measure the performance gain of data transfer operations. The cumulative data transfer time (multi-threaded) for Amazon is 25,167s, and for Google is 42,573s, which represents

a decrease of about 19% and 24% (Figure 5), respectively. We also observe a decrease on the average runtime of the `mConcatFit` job (Figure 1) from 1,862s to 790s on Amazon, and from 4,387s to 937s on Google, which represent 12% and 9% of the workflow makespan respectively.

A. Discussion on Potential Improvements and Research Directions for Running I/O-intensive Workflows on Cloud Computing Environments

The analyses performed in this paper showed that standard cloud environments may present performance issues for running I/O-intensive workflows. In particular, for workflows that operate over a large number of (small) files, the performance may be poor, as noticed for the Montage workflow. The current model provided by cloud storage APIs includes a GET request per object. Although multi-threading data transfers in Pegasus substantially reduce the overhead due to network communications, the performance loss is still high when several small files are involved. For instance, the total time required to transfer the input data for the `mConcatFit` job (a total of 1.9MB) from the object storage to the VM is 790s for Amazon, and 937s for Google. It is easy to notice that the overhead incurred by these data transfers significantly slows down the application. Since the Montage instance used in this paper is small (to avoid spending several cycles of computing resources), this overhead can exponentially grow with very-large runs of the Montage workflow (in the order of millions of jobs [24]).

A possible solution to mitigate this overhead is to use a bulk mechanism to concatenate and manage files within a single transfer request. As a result, the overhead added by transfer operation for each file can be masked. The resulting overhead for transfers would be negligible compared to the time to transfer the data. This approach is similar to the multipart strategy to get/put files from cloud storage services, where files are broken into chunks and transferred in parallel [25]. However, this approach should be used sparingly, since it could slow down the workflow execution as it may include (artificial) barriers for job execution (i.e., subsequent jobs will have to wait the entire transfer to start executing).

Our analyses also showed that the performance difference among data transfer clients is mostly due to the number of connections established to perform a transfer operation. Our findings show that if secured connections is not a requirement, not using them could significantly increase the performance. For example, `pegasus-s3` (which does not perform secured data transfers) yields better performance than `aws-cli` (which enforces secured connections). Low data transfer performance may also be related to the performance of the local disk used as a storage mechanism. In our experiments, we identified that Montage runs have a bursty access pattern, which explains why even though Amazon provides lower baseline I/O per GB, it still performs better than Google.

VI. Related Work

In the past decade, several researches have been conducted to understand the applicability of the cloud environment for the execution of scientific workflows [26]–[28]. For instance, in [2] the authors describe the experiences with deploying a scientific workflow on various cloud environments. An evaluation of the feasibility of cloud systems to meet the performance requirements of scientific applications at a reasonable price is conducted in [29]. In [3], [30], authors explored various data sharing options for scientific workflows on EC2, and also analyzed the issues in deployment, performance, and costs of running these workflows in a cloud environment [31]–[33]. A plethora of studies have been dedicated to cost- and deadline-aware scheduling and resource provisioning in cloud systems. The goal of these studies is to minimize the cost of executing workflows in clouds while meeting the user deadlines [4], [34]–[36]. In this paper, however, we aim to understand the bottlenecks involved in the execution of I/O-intensive workflows in clouds. We compare two widely used commercial clouds under different storage configurations.

In I/O performance analysis, studies have focused on measuring and analyzing the effect of provisioned network bandwidth on overall workflow execution time and I/O performance [37], [38]. In [39], authors analyze the I/O performance of HPC applications in cloud environments by varying disk storage per VM. However, their benchmarking does not involve an object storage (in their case Amazon S3). There have been studies on I/O performance isolation in cloud environment [40], [41], but they are limited to a fixed file size. In this paper, we conduct benchmarks on different file sizes, using different storage configurations and cloud specific storage clients.

Studies have been done to assess the performance of Amazon for high-performance computing and scientific applications. These studies have shown that the cloud environment requires significant improvement to match the needs of the scientific community [42], [43]. Consequently, we focus on understanding the overheads involved on I/O operations to identify measures of improvements where these overhead could be minimized. Unlike previous work on benchmarking Google Compute Engine [44], we compare a fixed set of resources between Amazon and Google using cloud specific clients. Our purpose is to identify performance issues from a scientific application's perspective that could lead to performance improvements.

VII. Conclusion

In this paper, we evaluated the performance of an I/O-intensive workflow on two widely used commercial clouds (Amazon and Google). We compared workflow execution

makespans in different cloud storage deployments to explore and quantify the impact of storage bottlenecks. Experimental results show that the overall performance loss in using *Cloud Storage* for intermediate files is about 400% when compared to storing data locally. We also observed that the overall workflow makespan on Google runs is higher than on Amazon for all configurations. We then conducted network and disk I/O benchmarking to identify the causes of these performance differences. We identified the overhead incurred on individual file transfer to be the culprit.

In configurations where the workflow data is stored externally to the VM, the transfer client provided by the workflow management system outperforms standard cloud tools. Further investigation showed that this performance gain is due to the overhead generated by additional TCP connections established by the cloud standard clients. When the workflow data is stored locally to a disk attached to the virtual machine (i.e., no external data transfer is performed), the I/O performance of the disk (in our case an SSD driver) has significant impact on the workflow makespan. Amazon appears as a preferable platform for bursty applications, since its policy allows I/O bursts up to half an hour. We also performed an evaluation of multi-threaded data transfers. As expected, the workflow execution performance significantly increases since the overheads are masked by the parallel transfers. However, for large workflows composed of jobs that operate over very small files, the overhead will still be a burden. Therefore, we suggested the exploration of a bulk transfer mechanism that would mask the overhead of individual transfers.

The cloud computing environment is constantly evolving, e.g., software and hardware are often updated/replaced, and new techniques are continuously emerging. Therefore, we acknowledge that performance measures vary and the conclusions derived from this paper may change accordingly. However, our methodology is still applicable. Figure 11 shows the variance in transferring an empty file every hour for over a month in *Cloud Storage* configuration. The evolving difference of transfer times for small files for `gsutil` is an indicator of change in the system overhead. However, since as a client, we have very limited knowledge about the underlying system, it is hard to pinpoint the cause. Nevertheless, the purpose of this paper is to identify bottlenecks and quantify the impact of these bottlenecks on I/O-intensive scientific applications. Therefore, the methodology and issues identified in this study provide new insights to evaluate the performance and requirements of I/O-intensive workflows in cloud environments. Future work include the continuous monitoring and analysis of cloud platforms to evaluate the impact of performance changes on the execution of scientific workflows, and the evaluation of other I/O-intensive applications from different science domains. We will also pay particular attention to improvements related to cloud data transfer tools.

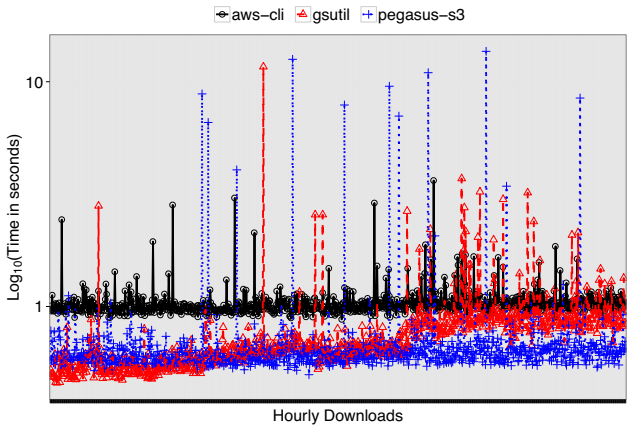


Figure 11. Time series download times of an empty file (0 Bytes) from an object storage to the VM for Amazon and Google clouds. The values were measured hourly during the time period of over a month (Oct 28, 2015 to Dec 02, 2015).

ACKNOWLEDGEMENTS

This work was funded by DOE under contract #DESC0012636, “Panorama - Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows”. We thank Karan Vahi and Mats Rynge for their valuable help.

REFERENCES

[1] D. A. Brown, P. R. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb, “A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis,” in *Workflows for e-Science*, 2007, pp. 39–59.

[2] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, and B. Berriman, “Experiences using cloud computing for a scientific workflow application,” in *Proceedings of the 2nd international workshop on Scientific cloud computing*. ACM, 2011, pp. 15–24.

[3] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, “Data sharing options for scientific workflows on amazon EC2,” in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2010, pp. 1–9.

[4] S. Abrishami, M. Naghibzadeh, and D. H. Epema, “Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, 2013.

[5] “Google compute engine,” https://cloud.google.com/compute.

[6] “Amazon elastic compute cloud (ec2),” http://aws.amazon.com/ec2.

[7] D. S. Katz, J. C. Jacob, E. Deelman, C. Kesselman, G. Singh, M.-h. Su, G. B. Berriman, J. Good, A. C. Laity, T. Prince *et al.*, “A comparison of two methods for building astronomical image mosaics on a grid,” in *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*. IEEE, 2005, pp. 85–94.

[8] R. Ferreira da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, and M. Livny, “Toward fine-grained online task characteristics estimation in scientific workflows,”

in *8th Workshop on Workflows in Support of Large-Scale Science*, ser. WORKS '13, 2013, pp. 58–67.

[9] R. Ferreira da Silva, G. Juve, M. Rynge, E. Deelman, and M. Livny, "Online task resource consumption prediction for scientific workflows," *Parallel Processing Letters*, vol. 25, no. 3, 2015.

[10] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny *et al.*, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.

[11] "gsutil," https://github.com/GoogleCloudPlatform/gsutil.

[12] "Pegasus-s3," https://pegasus.isi.edu/wms/docs/latest/cli-pegasus-s3.php.

[13] "Aws command line interface," https://aws.amazon.com/cli.

[14] "Amazon simple storage service (S3)," http://aws.amazon.com/s3.

[15] "Google cloud storage," https://cloud.google.com/storage.

[16] S. Srinivasan, G. Juve, R. Ferreira da Silva, K. Vahi, and E. Deelman, "A cleanup algorithm for implementing storage constraints in scientific workflow executions," in *9th Workshop on Workflows in Support of Large-Scale Science*, ser. WORKS'14, 2014, pp. 41–49.

[17] "Scp," http://linux.die.net/man/1/scp.

[18] "SFTP," http://linux.die.net/man/1/sftp.

[19] "Tcpdump," http://www.tcpdump.org.

[20] "Wireshark," https://www.wireshark.org.

[21] T. Dierks, "The transport layer security (tls) protocol version 1.2," 2008.

[22] J. Postel, "Rfc–793 transmission datagram protocol," *Information Sciences Institute, USC, CA*, 1981.

[23] "dd," http://linux.die.net/man/1/dd.

[24] M. Rynge, G. Juve, J. Kinney, J. Good, B. Berriman, A. Merrihew, and E. Deelman, "Producing an infrared multiwavelength galactic plane atlas using montage, pegasus and amazon web services," in *23rd Annual Astronomical Data Analysis Software and Systems, ADASS, Conference*, 2013.

[25] "Uploading objects using multipart upload api," http://docs.aws.amazon.com/AmazonS3/latest/dev/uploadobjusingmpu.html.

[26] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Scientific workflow applications on amazon ec2," in *E-Science Workshops, 2009 5th IEEE International Conference on*. IEEE, 2009, pp. 59–66.

[27] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*. IEEE, 2009, pp. 4–16.

[28] Y. Zhao, X. Fei, I. Raicu, and S. Lu, "Opportunities and challenges in running scientific workflows on the cloud," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*. IEEE, 2011, pp. 455–462.

[29] G. Juve, E. Deelman, G. B. Berriman, B. P. Berman, and P. Maechling, "An evaluation of the cost and performance of scientific workflows on amazon ec2," *Journal of Grid Computing*, vol. 10, no. 1, pp. 5–21, 2012.

[30] R. Agarwal, G. Juve, and E. Deelman, "Peer-to-peer data sharing for scientific workflows on amazon ec2," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*. IEEE, 2012, pp. 82–89.

[31] G. Juve, M. Rynge, E. Deelman, J.-S. Vockler, and G. B. Berriman, "Comparing futuregrid, amazon ec2, and open science grid for scientific workflows," *Computing in Science & Engineering*, vol. 15, no. 4, pp. 20–29, 2013.

[32] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the use of cloud computing for scientific workflows," in *eScience, 2008. eScience'08. IEEE Fourth International Conference on*. IEEE, 2008, pp. 640–645.

[33] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 50.

[34] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 22.

[35] M. Malawski, K. Figiela, M. Bubak, E. Deelman, and J. Nabrzyski, "Cost optimization of execution of multi-level deadline-constrained scientific workflows on clouds," in *Parallel Processing and Applied Mathematics*. Springer, 2014, pp. 251–260.

[36] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," *Future Generation Computer Systems*, vol. 48, pp. 1–18, 2015.

[37] J. Wang and I. Altintas, "Early cloud experiences with the kepler scientific workflow system," *Procedia Computer Science*, vol. 9, pp. 1630–1634, 2012.

[38] A. Mandal, P. Ruth, I. Baldin, Y. Xin, C. Castillo, M. Rynge, and E. Deelman, "Evaluating i/o aware network management for scientific workflows on networked clouds," in *Proceedings of the Third International Workshop on Network-Aware Data Management*. ACM, 2013, p. 2.

[39] D. Ghoshal, R. S. Canon, and L. Ramakrishnan, "I/o performance of virtualized cloud environments," in *Proceedings of the second international workshop on Data intensive computing in the clouds*. ACM, 2011, pp. 71–80.

[40] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of i/o workload in virtualized cloud environments," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 51–58.

[41] J. Shafer, "I/o virtualization bottlenecks in cloud computing today," in *Proceedings of the 2nd conference on I/O virtualization*. USENIX Association, 2010, pp. 5–5.

[42] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 159–168.

[43] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of ec2 cloud computing services for scientific computing," in *Cloud computing*. Springer, 2010, pp. 115–131.

[44] Z. Li, L. OBrien, R. Ranjan, and M. Zhang, "Early observations on performance of google compute engine for scientific computing," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. IEEE, 2013, pp. 1–8.