I/O Performance Modeling for Big Data Applications over Cloud Infrastructures

Ioannis Mytilinis *, Dimitrios Tsoumakos #, Verena Kantere $, Anastassios Nanos * and Nectarios Koziris *

* *Computing Systems Laboratory, National Technical University of Athens*
{gmytil, ananos, nkoziris}@cslab.ece.ntua.gr

Department of Informatics, Ionian University
dtsouma@ionio.gr

$ *Institute of Services Science, University of Geneva*
verena.kantere@unige.ch

***Abstract*—Big Data applications receive an ever-increasing amount of attention, thus becoming a dominant class of applications that are deployed over virtualized environments. Cloud environments entail a large amount of complexity relative to I/O performance. The use of Big Data increases the complexity of I/O management as well as its characterization and prediction: As I/O operations become growingly dominant in such applications, the intricacies of virtualization, different storage backends and deployment setups significantly hinder our ability to analyze and correctly predict I/O performance.**

To that end, this work proposes an end-to-end modeling technique to predict performance of I/O–intensive Big Data applications running over cloud infrastructures. We develop a model tuned over application and infrastructure dimensions: Primitive I/O operations, data access patterns, storage backends and deployment parameters. The trained model can be used to predict both I/O but also general task performance. Our evaluation results show that for jobs which are dominated by I/O operations, such as I/O-bound MapReduce jobs, our model is capable of predicting execution time with an accuracy close to 90% that decreases as application processing becomes more complex.

I. Introduction

The emergence of virtualization technologies has gradually brought forth a paradigm shift from traditionally managed software and service offerings towards Cloud-based ones. Traditional business application models incur high expenses and complexity: Customized and often large amounts of hardware and software are required; on top of that, configuration, management, update, security, etc, costs complicate the task of designing, deploying and successfully maintaining a service available to multiple clients. Migrating applications to the Cloud enables businesses to significantly reduce time-to-market delays and operational costs. Hardware and software management is undertaken by the infrastructure vendor, who also provides a flexible, pay-as-you-go pricing scheme: Business owners pay only for what they need, adaptively scaling resources to meet demand.

These features make Cloud infrastructures an ideal platform for *Big Data Applications* (or *BDAs*). Big Data is generally described through the well-known notion of *the four V's*: Volume, Velocity, Variety and Veracity. An ever-growing interest in analytics and data-driven decision making has made BDAs a number-one priority that allows businesses worldwide to define new initiatives and re-evaluate their current strategies.

With the size and the corresponding demand for data ever-increasing, the importance of I/O in BDAs is particularly stressed. As I/O operations take up a considerable portion of the application's operations, they can easily dominate total application performance. At a higher level, frequent (and possibly concurrent) reading and writing of large files is required, whether a real-time system or a batch processing application is concerned. Moreover, deployment of BDAs over virtualized environments is a common and popular choice. This combination increases the complexity of I/O management as well as its characterization/prediction.

Including the virtualization layer, there exists a large amount of complexity (measured in intermediate infrastructure) between the application users and the storage devices that hold the data. As Fig. 1 shows, an I/O request may have to go through the VM's main memory, some hypervisor-dependent drivers, the VM host memory and the network before it finally reaches the storage system, which, in turn, may have its own caches. Due to this level of complexity, monitoring and subsequent characterization of isolated physical components (such as storage devices, networks, etc) is not sufficient; results might be too complex to analyze or combine for a definitive I/O modeling, especially given the numerous choices of different setups, vendors and hardware involved. Rather, an *end-to-end* approach is required to more accurately depict overall I/O performance.

In this work we embark on describing such an approach: An I/O profiling scheme and its subsequent modeling for Cloud-based BDAs. We believe that such a study will offer valuable information related to which application workloads and storage characteristics affect BDA I/O performance, and thus the end-to-end application behavior. In our work, I/O performance is measured and modeled according to the following dimensions:

1) *I/O Operations*: We measure primitive *read* and *write* operations.

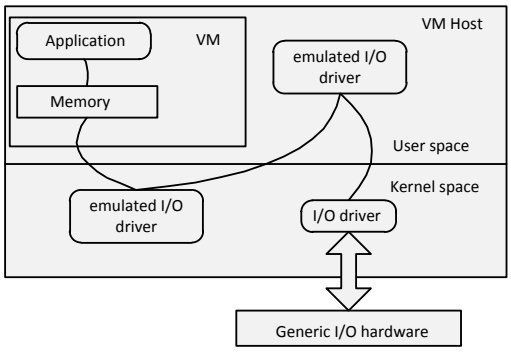


Figure 1: I/O path in a virtualized environment.

2) *Data and Access*: We utilize "raw data" (bytestreams) between several MB to several GB of no particular format. Data is accessed through well-known distributions: Uniform, sequential and skewed with varying degrees of skewness.
3) *Storage backends*: We utilize both local storage as well as distributed object stores for the VM storage device.
4) *BDA Deployments*: We consider BDAs deployed both over a single as well as multiple VMs.

The aforementioned dimensions are carefully chosen so as to capture important, modern architectural considerations in the cloud but also abstract away the complexity that the plethora of available software and hardware offerings bring into I/O characterization. This paper makes a twofold contribution:

- First, it proposes a profiling methodology for the I/O path in virtualized environments. Understanding the complexity of Cloud I/O, we model it in an end-to-end fashion.
- Second, we evaluate the usefulness of this model, not exclusively over I/O but on the total application performance as well. Implementation results from an academic public cloud deployment over multiple sample analytics jobs and platforms are thoroughly described.

Our results show a varying degree of precision that directly depends on the overhead that the application platform and the application's pure processing part (non-I/O) impose on performance. For tasks that are dominated by I/O operations, our model is remarkably accurate.

II. Problem Overview

Formally, we define the problem in its general form as:

Problem 1. *Given a multi-dimensional input space I, which contains a set of application and system characteristics, find a function $f : I \longrightarrow M$ which maps the BDA and system characteristics to an application performance space M.*

We consider I to be $I = S \times A$, where S is the space of storage and A the space of application characteristics. As we treat the I/O path as a black box, we do not examine fine-grained storage characteristics but consider S to be a set of different storage architectures. Space A can be divided to the following subspaces: Workload subspace (W), data subspace (D) and resource subspace (R). Thus, A = W×D×R. The dimensions of each of these subspaces are presented in detail in Section III.

The output space M may contain any metric that characterizes application performance. Such metrics can be related to performance, availability, etc. Nevertheless, in this paper, we consider only performance metrics and more specifically the throughput of I/O operations (MB/s) as it is perceived by the application user.

III. Application Modeling

For each of the application characteristics subspaces: W, D and R, we define the measures we are interested in for our modeling.

A. Resources

Subspace R includes all the hardware/software resources a BDA may use. Such resources are the number of VMs an application needs, the number of VCPUs, the amount of RAM VMs have, etc.

With respect to the number of VMs an application utilizes, we make the following distinction: An application can be either *single-VM* or *distributed*. A single-VM application may accept requests from zero to many clients but there is only one virtual machine which interacts with the storage.

In contrast, in a distributed application, there exist multiple VMs involved in the I/O procedure. And in this case, the application can accept requests from zero to many clients, but there are many VMs of the same application which issue concurrent I/O requests to the storage system.

B. Workload

Workloads with different characteristics may lead to different I/O performance. As the possible workload features that affect performance are numerous, we define the following measures of the W subspace: the type and locality of operations, the data access pattern and the degree of concurrency.

Since we aim at I/O performance evaluation, we are interested in primitive *read* and *write* operations. These operations may demand network access (remote read/write) or not (local read/write). Moreover, these operations may access data following different patterns. In order to examine the caching effects of main memory on the overall I/O performance, we consider two basic access patterns: sequential and x-y skewed, with variable values for x and y.

The fourth defined workload dimension is concurrency. We consider two types of concurrency:

- Client concurrency: A BDA may have many clients issuing I/O requests in parallel.
- VM concurrency: The number of nodes in the cluster of a distributed BDA.

C. Data

In the current work and in order to acquire performance profiles, we apply different workloads on raw data. By raw data, we refer to a collection of bytes which does not obey

to a specific structure. Therefore, dimensions such as data structure, mutability, etc., are not taken into account. The only measure of space D we consider is the size of data read or written. Since our modeling mainly targets BDAs, we decide for the data size to pick values from the order of MB to several GB.

IV. Storage

Storage system related information is included in the input space I. The storage system may be as simple as a single hard disk drive or an arbitrarily complex system. Cloud data centers employ a great variety of storage architectures. For example, Amazon offers local disk images, Elastic Block Store (EBS) [1] and S3 [2] storage options. As storage systems with different architectures may present different performance characteristics, in order to demonstrate the general applicability of the proposed methodology, we consider two completely different systems from an architectural point of view: *local* and *distributed*. In the first case, storage is directly attached to the virtual machine container, while in the second case, it is a distributed pool of storage nodes, available to more than one VM containers. However, our methodology can be applied to any other storage architecture chosen to be profiled.

In the local storage setup, VMs perform I/O operations on the local disk subsystem of their host. Thus, all the VMs of a single container compete for the same disk or disk array.

In a distributed storage architecture, VMs do not interact with the local disk of their container but they act as clients to a distributed storage cluster. For each I/O request issued by a VM at least one storage node of the distributed cluster is responsible to serve the request. The nodes of a distributed storage cluster can be from simple block devices to complete servers with large main memory and computing capabilities.

V. Methodology

Having described in detail the input space I, we present the proposed methodology for predicting the values of function f, which maps the input space I to the output space M ($f : I \longrightarrow M$). Our methodology can be summarized in the process presented in Algorithm 1. In a nutshell, we first select a subspace P of the application characteristics space A ($P \subseteq A$). Then, for each storage architecture, we measure performance for points $p \epsilon P$. Let $Prof\,(P, S)$ denotes the profiled performance for all storage architectures under evaluation and for all points in the application characteristics subspace P. For all storage architectures $s \epsilon S$, we use $Prof\,(P, s)$ to train a CART [3] model $\hat{f}_s$ (described in detail in Section VI) that predicts the corresponding values of function f.

What remains to be defined is which points of A should be included in subspace P. For each dimension of the application-characteristics space, we select a range of values. For dimensions of A which contain a finite set of values, such as the type of operation $\{read, write\}$, we consider all possible values. For dimensions that contain many or possibly infinite values, such as $\#VM \epsilon \{1, 2, ...\}$, we conduct preliminary exploratory tests and consider a subset of values, such that the choice of any other point along this dimension does not cause a significant performance variation. Table I summarizes the selected ranges of values in the current profiling process.

Algorithm 1 Methodology for predicting the values of f

1: Determine a subspace P of A
2: **for all** $storage \in S$ **do**
3: **for all** $p \in P$ **do**
4: $m_p = Prof\,(p, storage)$ {Measure performance for p and $storage$}
5: **end for**
6: Use $m = \bigcup m_{p,p \epsilon P}$ to create a CART model $\hat{f}_{storage}$ that predicts $f\,(storage, A)$
7: **end for**
8: **return** $\hat{f}_S$ {where $\hat{f}_S$ is the set of created models.}

Table I: The selected values that form the subspace P of the profiling process.

Dimension	Values
#VMs	$[1 - 16]$
type of operation	read, write
locality of operations	local, remote
access pattern	sequential, $[50 - 50, 90 - 10]$
concurrent clients	$[0 - 4]$
data size	$[64MB - 4GB]$

For the implementation of the profiling process, a profiling tool is developed. Our profiling tool consists of a deployment tool and a synthetic workload generator. Given an input file with the description of dimensions and the corresponding value ranges for each dimension, the deployment tool deploys on the cloud the requested VMs, configured with storage s, and on top of them, accordingly configured, the synthetic workload generator. The synthetic workload generator generates load for all $p \epsilon P$ and measures the achieved performance $Prof\,(p, s)$ for each point.

VI. Modeling

When the profiling process terminates, performance $Prof\,(P, s)$ is known for all available storage architectures $s \epsilon S$. The profiling results are then used to generate a performance prediction model $\hat{f}_s$ for each storage system.

Profiling indicated that performance presents a non-linear behavior, not easily predicted by simple models like linear regression. Furthermore, in many cases, the behavior of performance is not even monotonic. As a more generic model is required and for the ease of being appropriately tuned, we use, as a modeling tool, Classification and Regression Trees(*CART*) [3]. CART are a special case of a broad predictive models family called decision trees. In these tree

structures, the internal nodes of the tree contain conditions which involve the input parameters and the leaves of the tree contain the values of the function being modeled.

In order to illustrate the appliance of CART to our problem we give an example. Consider the case we are interested in predicting the performance of local read, of a single-VM application, for various access patterns and file sizes(in MB), when the local storage is employed. Fig. 2 depicts the generated decision tree. For example, if we have a random access pattern and a data size $\geq$ 192 MB, then according to the model, we expect a throughput of 33 MB/s.

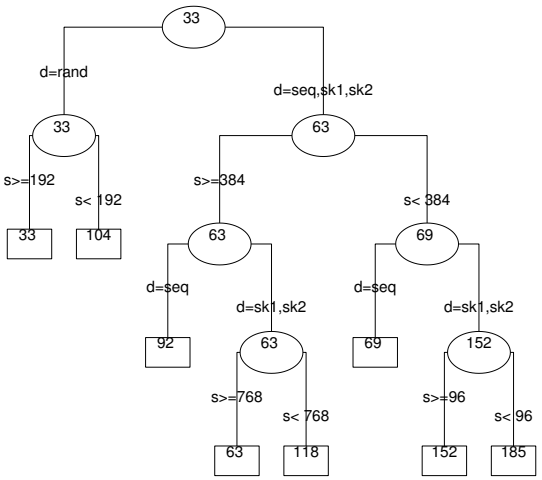


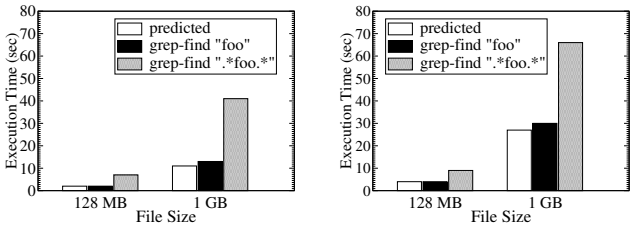
Figure 2: CART model for a single-VM application which runs on top of the local storage. The application performs local reads with various access patterns (d) and sizes (s).

VII. Experiments

The experiments were carried out in the ∼okeanos academic public cloud [4]. The deployed VMs run Ubuntu 12.04 LTS, with 2 GB of main memory, 2 VCPUs and 10 GB disk image each. As storage options, ∼okeanos offers two alternatives: *Standard* and *Archipelago*.

The *Standard* storage option is based on the *DRBD* [5] system and is the local storage architecture we use. In the DRBD model, each VM container has a locally attached disk subsystem, which is responsible to serve all the I/O requests of the hosted VMs. Nevertheless, DRBD is designed to offer high availability. Availability is achieved by implementing a network RAID-1 mechanism. Read operations access directly the local storage system. However, in write operations, data is replicated through the network to one more physical machine. In the ∼okeanos infrastructure, the network employed to the DRBD setup is 1Gb Ethernet.

Archipelago is a fully distributed storage system and is based on a *RADOS* [6] cluster, hosted on dedicated storage servers. RADOS is a distributed object store, which consists of a large collection of storage nodes, called *OSDs* and a small group of monitors responsible for managing OSDs.



(a) local storage (b) distributed storage

Figure 3: Grep for different patterns, file sizes and storage architectures.

When a client inserts data to a RADOS cluster, data is first chunked to objects of default size 4 MB, and is distributed to the OSDs according to the CRUSH algorithm [7]. RADOS objects have a default replication factor of two in favor of availability and fault tolerance.

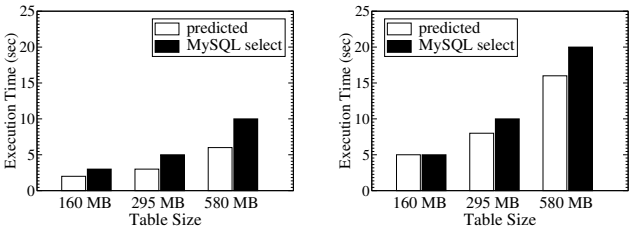
In the Archipelago storage system, there is a 10 Gb Ethernet link between the OSDs and 3×1 Gb Ethernet links between each VM container and the storage cluster.

For testing our model, we conduct experiments on some I/O-intensive applications. We make the following distinctions among I/O-intensive applications: An application may be single-VM or distributed and it may also be interactive or perform batch processing.

As single-VM batch applications, we consider the grep utility of UNIX and as single-VM interactive application a *select* operation on a MySQL table. The MySQL select workload scans the whole table and does not use any index. As distributed batch jobs, we consider some Hadoop examples and as distributed interactive applications some HBase workloads.

Fig. 3a, 3b present the execution time for the grep application when different file sizes and storage architectures are used. The grep application reads sequentially the input file and returns as output the lines of the file which match a given regular expression. We experiment with two different patterns as regular expressions: one which demands a few computations(exact match of the string *foo*) and a more compute-intensive one(find all strings containing *foo*). For the less compute-intensive case we predict execution time with an average accuracy of 90% in both storage setups. However, when more computations are involved, accuracy drops to 30% for the local setup and to 40% for the distributed one. These results validate our model, since even with the noise of a few computations, our model achieves an accuracy of 90%.

The other single-VM application used in our experiments is a MySQL query that fetches sequentially all the rows of a table. As Fig. 4a, 4b show, the accuracy of our model is in the order of 60% and 80% for the local and the distributed storage respectively. As the computational overhead of MySQL is higher than that of grep, a drop in accuracy is observed. However, since both applications have similar workloads, the trend does not change and the local



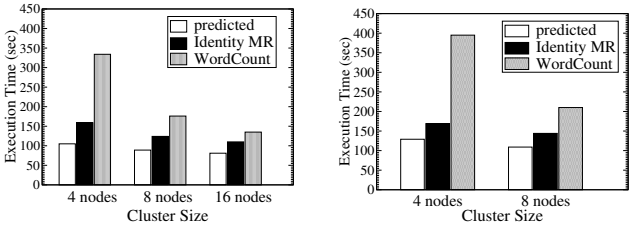
(a) local storage (b) distributed storage

Figure 4: MySQL table scan for different table sizes and storage architectures.

storage outperforms twice the distributed.

For the distributed applications, we set up Hadoop 1.2.1 with the default configuration. In Fig. 5a, 5b we predict the completion time of a Word Count job. The information our model needs to predict a MapReduce job completion time is the cluster size, the number of reducers and the size of input/output data for both map and reduce phases. We experiment with three different cluster sizes(4, 8, 16 nodes), 2 reducers and 2 GB input/output for both map and reduce phases. As MapReduce jobs, we use the Identity MapReduce and the Word Count example.

The Identity MapReduce copies a file from one HDFS path to another one. The Word Count job creates an occurrence frequency histogram for the words of a document. For the Identity MapReduce job, we predict completion time with a 70% accuracy for the local and 74% for the distributed. Even if there is no user-defined computation, the framework imposes extra overheads(parse key-values, compute partitions, sort keys, etc.) that cannot be predicted without profiling Hadoop [8]. However, we observe in both Figures that since I/O is the main bottleneck in the specific job, changes in the cluster size are captured by the model and prediction errors are kept constant.



(a) local storage (b) distributed storage

Figure 5: MapReduce job completion time when VMs are configured with different storage architectures.

For the Word Count example, where user-defined computation is included in both map and reduce phases, the prediction error is higher. Nevertheless, we observe that the accuracy of our prediction is ameliorated as the nodes of the Hadoop cluster are increased. For the local storage setup we have 31% accuracy in a 4-node cluster and 60% accuracy in a 16-node cluster. As the cluster size increases, the degree of parallelism in the map phase is increasing as well and map computation is spread across the cluster. As a result, the map phase overhead is decreasing.

The distributed interactive application we use in order to test performance for non-sequential patterns is HBase. We set up an 8-node cluster using HBase 0.94.20 with the default configuration and we load a 2 GB sized table, with YCSB [9] data as each row to have an 8-byte key and 1 KB value. The imposed workload is 1 MB-sized scans, performed concurrently from 2 or 4 clients and following different row-access distributions.

In Fig. 6 we see that we achieve an accuracy of about 60% when 2 clients issue requests simultaneously and 50% when 4 clients are used. The observed error is due to HBase specific overheads. HBase tuning leads to significantly different performance results. However, such a study is out of the scope of this paper.

Considering the HBase experiment, we make one more observation. Although the prediction error is high enough our prediction follows the trend that the application run exhibits. Thus, and in this case, even if we cannot make safe predictions, we can predict the relative deviation among different workloads and storage architectures.

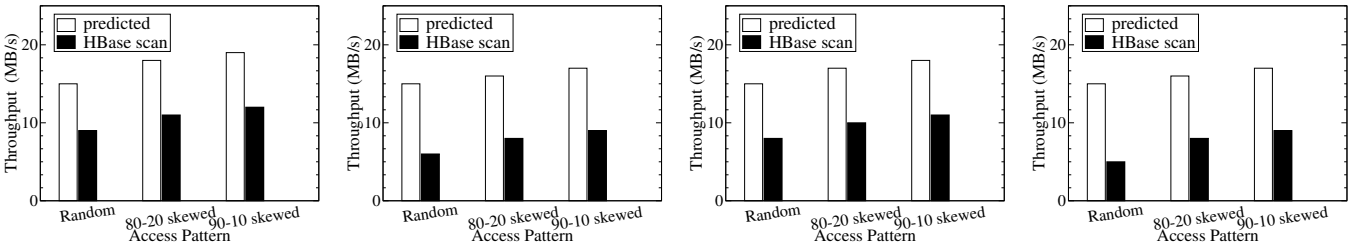
VIII. Related Work

I/O behavior modeling and prediction have been well-studied. In [10] and [11] modeling for disk drives and disk arrays is performed. Both approaches make a hierarchical decomposition of the I/O path and examine the impact of each component separately. However, this is quite impractical for large-scale, complex virtualized environments.

Other well known approaches to disk array modeling are the ones presented in [12] and [13]. While [12] models the disk array and [13] treats it as a black box, they both define workload dimensions and they sample performance for a predetermined set of points of the input space. Then a model is used to fit the measured numbers. Reference [13] also shows that CART provide better accuracy than other methods for disk array modeling. Nevertheless, as these models target only at disk arrays, the workload dimensions defined are not adequate to capture BDA workload characteristics.

Although these approaches seem to work satisfactorily enough for disk I/O prediction, it is hard to work when a long and complex I/O subsystem path is employed. To this direction, the self-scaling benchmark is presented in [14]. This benchmark measures I/O performance as it is seen by an end user issuing reads and writes. An end-to-end approach is also used in [15]. However, the system proposed there does not refer to the Cloud but to a large scale parallel multicore system and it does not predict I/O related metrics but CPU execution time.

In [16], the authors use IOR [17] in order to compare the I/O performance of HPC applications on two cloud platforms- Amazon and Magellan. In the case of Amazon, recognizing the need for evaluating different storage architectures, they also benchmark I/O performance on local,



(a) 2 clients-local storage. (b) 4 clients-local storage. (c) 2 clients-distributed storage. (d) 4 clients-distributed storage.

Figure 6: clients perform scans according different access distributions on a 8-node HBase cluster configured with different storage setups.

ephemeral devices and EBS volumes. However, their study is limited only to HPC-specific applications.

I/O characterization in virtualized environments is also carried out in [18] and [19]. Understanding I/O performance reveals opportunities for server consolidation and designing efficient cloud storage infrastructures. However, [18] is an experimental study for specific applications and does not include modeling of I/O behavior.

An alternative approach to the problem of I/O characterization in the cloud is presented in [20]. I/O traces from production servers are collected and used as a training set for a machine learning tool. During the learning process, I/O workload types are identified automatically and as output, a I/O workload generator is produced. This generator can simulate real application workloads and thus it can be used for storage systems evaluation.

IX. Acknowledgment

The research leading to these results has been partially supported by the European Commission, Seventh Framework Programme (FP7/2007- 2013) in terms of the ASAP FP7 project (grant agreement no 619706) and the CELAR 317790 FP7 project (FP7-ICT-2011-8).

X. Conclusions

In this paper we have argued towards an end-to-end modeling of I/O performance for Big Data applications running over the cloud. To achieve that we followed a three-step process: First, we identified some important BDA factors that affect I/O performance. Then, in order to measure performance we created a synthetic workload generator, based on primitive I/O operations. Finally, CART modeling was applied to fit the results of the profiling process.

Our experiments show that the created model captures with high accuracy I/O performance. A maximum accuracy of 90% for batch tasks and 60% for interactive applications has been observed. This shows that, for applications with low computational overhead, our model is sufficient to predict the total execution time. However, as more computation is involved, the prediction of the model proves to be over-optimistic. In this case, our model cannot safely predict execution time but it can accurately place the lower bound that the employed storage architecture imposes on application execution time.

References

[1] "Amazon elastic block store," http://aws.amazon.com/ebs.

[2] "Amazon s3," http://aws.amazon.com/s3.

[3] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[4] "okeanos iaas," https://okeanos.grnet.gr/home.

[5] P. Reisner, "Drbd," in *Linux Congress*, 2000.

[6] S. A. Weil, A. W. Leung, S. A. Brandt, and C. Maltzahn, "Rados:a scalable, reliable storage service for petabyte-scale storage clusters," in *PDSW*, 2007.

[7] S. A. Weil, E. L. Miller, S. A. Brandt, and C. Maltzahn, "Controlled, scalable, decentralized placement of replicated data." in *SC*, November 2006.

[8] H. Herodotou and B. Shivnath, "Profiling, what-if analysis, and cost-based optimization of mapreduce programs," in *VLDB*, 2011.

[9] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *SoCC*, 2010.

[10] C. Ruemmler and J. Wilkes, "An introduction to disk drive modeling," in *IEEE Computer*, 1994.

[11] M. Uysal, G. A. Alvarez, and A. Merchant, "A modular, analytical throughput model for modern disk arrays," in *MASCOTS*, 2001.

[12] E. Anderson, "Simple table-based modeling of storage devices," HP Labs, Tech. Rep., 2001.

[13] M. Wang, K. Au, A. Ailamaki, A. Brockwell, C. Faloutsos, and G. R. Ganger, "Storage device performance prediction with cart models," in *MASCOTS*, 2004.

[14] P. Chen and D. A. Patterson, "A new approach to i/o performance evaluation-self scaling i/o benchmarks, predicted i/o performance," in *SIGMETRICS*, 1993.

[15] E. Ipek, B. R. de Supinski, M. Schulz, and S. A. McKee, "An approach to performance prediction for parallel applications," in *Euro-Par*, 2005.

[16] D. Ghoshal, R. Canon, and L. Ramakrishnan, "I/o performance of virtualized cloud environments." in *DataCloud-SC'11*, 2011.

[17] H. Shan, K. Antypas, and J. Shalf, "Characterising and predicting the i/o performance of hpc applications using a parameterized synthetic benchmark." in *SC2008*, 2008.

[18] A. Gulati, C. Kumar, and I. Ahmad, "Storage workload characterization and consolidation in virtualized environments," in *VPACT*, 2009.

[19] S. Kraft, G. Casale, D. Krishnamurthy, D. Greer, and P. Kilpatrick, "Performance models of storage contention in cloud environments," *Software & Systems Modeling*, vol. 12, no. 4, pp. 681–704, 2013.

[20] C. Delimitrou, S. Sankar, K. Vaid, and C. Kozyrakis, "Decoupling datacenter studies from access to large-scale applications: A modeling approach for storage workoads," in *IISWC*, 2011.