Towards Fast Profiling of Storage Devices Regarding Access Sequentiality

Francieli Zanon Boito[1,2], Rodrigo Kassick[1,2], Philippe O. A. Navaux[1] and Yves Denneulin[2]
[1]Institute of Informatics – Federal University of Rio Grande do Sul – Porto Alegre, Brazil
{fzboito, rvkassick, navaux}@inf.ufrgs.br
[2]LIG Laboratory – INRIA – University of Grenoble – Grenoble, France
{yves.denneulin}@imag.fr

ABSTRACT

This work presents a tool for storage device profiling named SeRRa. Our tool obtains the sequential to random throughput ratio for reads and writes of different sizes. Several optimizations aim at adapting applications' access patterns in order to generate contiguous accesses for improved performance when accessing storage devices like hard disks. However, when considering other storage options like RAID arrays and SSDs, the access time ratio between contiguous and non-contiguous accesses may not compensate for these optimizations' cost. In this scenario, the information provided by our tool could be used to dynamically decide if optimizations are beneficial for performance, an important task to provide I/O throughput. In order to provide this information efficiently, SeRRa employs benchmarks to obtain the values for a subset of the parameter space and estimates the remaining values through a linear model. Extensive test campaigns with our tool on nine storage devices have shown median errors of approximately 5% (up to 55%) while taking only 1/168 of the originally needed time.

Categories and Subject Descriptors

D.4.8 [**Operating Systems**]: Performance

General Terms

Storage Device

1. INTRODUCTION

For years, magnetic disks (*Hard Disks Drives* - HDDs) have been the main non-volatile storage device on personal computers and supercomputers for High Performance Computing (HPC). Because of that, most systems were adapted in order to obtain good performance when accessing these devices. For instance, disk schedulers employ elevator algorithms in order to minimize head movements [13], improving data access performance. Several optimizations focus on generating sequential accesses instead of random ones, since this access pattern benefits HDDs.

Solid State Drives (SSDs) are a recent alternative to hard disks. These devices employ mainly flash memory to store data, with no mechanical parts. This characteristic makes them more resistant to falls and vibrations. Moreover, their advantages over HDDs include: size, noise generation, heat dissipation and energy consumption. However, despite the growing adoption of SSDs, their larger cost per byte still hampers their use on large-scale systems for HPC. Therefore, several parallel file system deployments on clusters still store data on hard disks. Nevertheless, hybrid solutions where both devices are used have been gaining popularity.

Another popular solution for storage on HPC systems is the use of RAID arrays that combine multiple hard disks onto a virtual unit for performance and reliability purposes. Data is striped among the disks and can be retrieved in parallel, which improves performance. Nonetheless, since both SSDs and RAID solutions are inherently different from HDDs, they should not be treated simply as "faster disks". Several assumptions about performance from HDDs do not hold when using SSDs and RAID arrays, and different requirements arise [10].

Regarding spatial locality, HDDs are known for having better performance when accesses are done sequentially. Although RAID arrays are composed by hard disks and their performance is usually better with sequential accesses, their organization complicates this behavior. The alignment of accesses to the stripe size has a great impact and must be considered. On the other hand, works that aim at characterizing SSDs' performance behavior achieve different conclusions. Despite most SSDs presenting better performance for sequential writes than for random writes, read operations' performance follows no general rule. On some SSDs, there is no difference between sequential and random accesses, but on others this difference achieves orders of magnitude [5]. The sequential to random throughput ratio on some SSDs surpasses what is observed on some HDDs [10].

Therefore, we cannot simply classify optimizations - such as I/O schedulers [2] - by saying they are only suitable for HDDs or SSDs. Approaches that aim at generating contiguous accesses (originally designed for HDDs) can greatly improve performance when used on SSDs that are also sensitive to access sequentiality. Furthermore, on any device, the performance improvement caused by the use of a specific optimization may not compensate its overhead. Hence, these

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SAC '2015, April 13 - 17 2015, Salamanca, Spain
Copyright 2015 ACM 978-1-4503-3196-8/15/04 $15.00
http://dx.doi.org/10.1145/2695664.2695701.

optimizations could be classified according to the sequential to random throughput ratio that devices must present in order to benefit from them.

In this scenario, this paper proposes a tool for storage devices profiling named SeRRa. **Our tool reports, for a storage device, the sequential to random throughput ratio for read and write operations with different requests sizes.** Since I/O profiling of storage devices is a time consuming task, SeRRa aims at providing accurate results as fast as possible, facilitating its use. For that, benchmarks are executed only over a subset of all profiled requests sizes, and the remaining values are estimated through linear regressions.

This paper is organized as follows: the next section discusses related work. Section 3 presents the design of SeRRa. The error induced by the estimation process is discussed on Section 4, as well as results obtained with our tool for nine real storage devices that include HDDs, SSDs and RAID arrays. Finally, Section 5 brings final remarks.

2. RELATED WORK

With the growing adoption of solid state drives, several works focused at characterizing these devices by evaluating their performance over several access patterns [5, 10]. These works point at SSDs' project options, their impact on performance, and illustrate common phenomena as *write amplification* and *stripe alignment*. Differently, our work aims at measuring storage performance in a generic way, and not only on modeling or explaining SSDs' performance.

El Maghraoui et al. [7] propose a detailed model of SSDs' performance. Nonetheless, the model needs low-level information that must be profiled through micro-benchmarks. Moreover, the proposed model is a low-level model, focusing on the device only and not including higher levels of the I/O subsystem. A similar approach is used by Desnoyers [6]. Such models are suitable for evaluating project options for SSDs, for instance, contrary to what our tool does. We aim at providing a high level profiling of the storage system in order to make decisions about optimizations.

For similar reasons, device simulators like *DiskSim* [4] and others [1, 8, 12] have no use on our context. They allow the evaluation of devices parameters, but not the profiling of existing systems.

Although several benchmarking tools report access time and throughput on the access to files over different access patterns, we could not find any tool that reports the sequential to random throughput ratio. Other tools also do not estimate results for a large set of parameters from a few measurements, as SeRRa does through linear regressions. These reasons motivated the development of the tool, described in the next section.

3. A STORAGE DEVICE PROFILING TOOL

This section describes SeRRa, a storage device profiling tool. Its development was motivated by the need for a fast way to obtain the sequential to random throughput ratio from devices as HDDs, SSDs and RAID arrays. The main goals of SeRRa's project are:

- Performance: the information must be provided as quickly as possible;

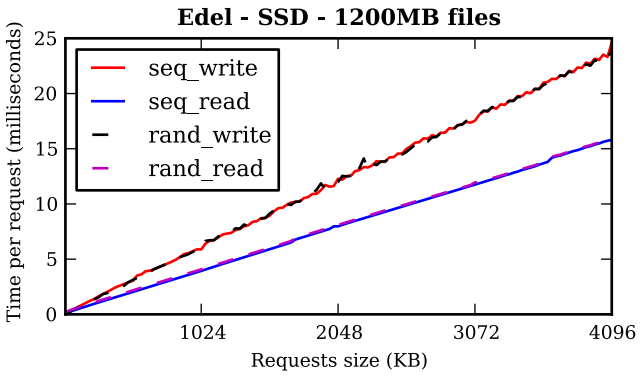

Figure 1: Profiling data from a SSD - Access time per request for several request sizes.

- Accuracy: the provided information must fairly reflect the real behavior of the profiled storage device;
- Generality: the tool must be easy to use and do not require user-provided information about the device.

Table 1: Original time in hours to profile the storage devices used in this study.

	One test execution	Total profiling time
Pastel (HDD)	15.22	329.49
Graphene (HDD)	12.26	120.38
Bali (HDD)	10.38	126.10
Chimint (RAID-0)	2.33	69.13
Suno (RAID-0)	4.21	61.32
Taurus (RAID-0)	2.43	57.95
Edel (SSD)	3.09	40.44
Graphite (SSD)	3.45	60.90
Bali (SSD)	2.67	82.49

Keeping both performance and accuracy goals at the same time is a challenging problem because profiling a storage device adequately can take several hours. Table 1 presents the time spent to profile the devices used in this study. It includes time required for one execution of the tests and for the complete profiling (that includes several executions in order to provide statistical guarantees). Details about these devices and the executed tests will be presented in Section 4. From the times reported in Table 1, it is clear that these tests are time consuming, as the fastest profile took over 1.5 days, while the slowest one took almost 14 days.

Ideally, such a complete profiling would be needed only once for each storage device and its stored results could be used for a long time. However, even considering this fact, to dedicate the environment for days can be prohibitive. Moreover, since aspects other than the storage device like local file system, disk scheduler and operating system also affect performance, the reported results are dependent on a system configuration. Changing the operating system version, for instance, could require a new profiling.

The slow profiling whose times are presented in Table 1 consists of executing an I/O benchmark several times varying file size, requests size and operation (sequential read,

random read, sequential write, and random write). Figure 1 shows an example of access times graph - obtained for 1200MB files with different requests sizes on a SSD. Similarly to what can be observed in this example, most of the access times graphs present a linear function appearance. Because of this observation, we have decided to use linear regressions on the design of our profiling tool. Therefore, the following steps compose SeRRa's execution:

1. **Monte Carlo:** request sizes inside a given interval are randomly picked. This interval is provided as a parameter, as well as the gap between every two consecutive sizes. For instance: if asked to approximate the results for the interval [8KB, 40KB] using 8KB gaps, the tool would pick points from the set {8KB, 16KB, 24KB, 32KB, 40KB}. The number of points to be selected for each interval is also defined by the user. Multiple intervals, with different gaps, could be provided. The points at the extremes of the interval (in this example, 8 and 40KB) are always included on this step.

2. **Benchmark:** the test is executed for the requests sizes that were picked on the previous step. We decided to use *IOzone* [9] because it is a widely adopted I/O benchmark, easy to install and use, and because it fits our needs. Other benchmarks that allow sequential and random accesses, like IOR [11], could be used with few modifications on the tool. The user also provides the file size to the tests.

3. **Linear Regression:** from the access times measured on the previous step, the complete set of access time results for each given interval is estimated through linear regression. Each test (read or write, sequential or random) for each interval is estimated separately.

4. **Report:** The sequential to random throughput ratios for the read and write tests are reported. The tool provides such values for all requests sizes, as well as averages, maximum and minimum values. The estimated access times curves are also informed.

The tool, implemented on *Python*, is open source and available at
`http://www.inf.ufrgs.br/~fzboito/serra.html`.

4. RESULTS

This section presents the SeRRa profiling tool's evaluation. First, we describe the test environments and methodology on Section 4.1. Then Section 4.2 discusses the error induced by the estimation of access times through linear regression. Lastly, Section 4.3 evaluates the tool's results and discusses its trade-off between performance and accuracy.

4.1 Tests Environments and Methodology

This section describes the nine systems used as our test environments, listed in Table 2. These systems were selected by their variety, aiming at representing most available devices. Aside from the system named "Bali", all environments are nodes from *Grid'5000* clusters [3].

The tests were executed on *Linux* operating system, using *IOzone* 3.397. All tested devices were accessed through the *Ext4* local file system, and the default *cfq* disk scheduler was kept. Both virtual memory's page size and file system's block size are 4KB. Caching was explicitly disabled through the *O_DIRECT* POSIX flag (*"-I"* parameter for IOzone).

For the tests described in this paper, five different file sizes were used - 40MB, 200MB, 400MB, 800MB and 1200MB. On all tested devices, we observed that the access time curves usually stabilize before 1200MB, not showing significant differences as we increase the file size further.

The requests size was increased from 8KB to 4MB. This range was divided into two intervals: from 8KB to 64KB with gaps of 8KB; and from 64KB to 4MB with gaps of 32KB. Most parallel file systems employ stripe sizes of at least 32KB for their data servers, therefore the requests they receive (and are object to optimizations) are usually multiples of this value. This happens because each large request issued by an application becomes a set of requests, that are multiples of the stripe size, to servers. We defined a smaller gap on the first interval in order to represent small requests.

4.2 Estimation error by linear approximation

This section aims at quantifying the error induced by estimating the access time curves through linear regressions.

In order to do that, we executed the complete profiling - without estimations, executing the benchmarks for all the specified request sizes - on all tests environments with the parameters discussed on Section 4.1. All tests were repeated until a 90% confidence could be achieved with a *t-student* distribution - with at least six executions. The maximum accepted error was of 10%. The necessary time to obtain this complete profile per machine, up to 14 days, was previously presented in Table 1.

A modified version of SeRRa was then executed on all devices varying the number of measuring points per interval (number of requests sizes for which the benchmark is executed) from 2 to 8. In the end, instead of reporting the sequential to random throughput ratio, the modified tool compared its estimated curves with the measured ones, obtained through benchmarking only. The graphs from Figure 4.2 present the absolute values of estimation error for the four tests. For each tested environment (represented by a different color), we present results for both intervals (interval 1 - from 8KB to 64KB - represented by circles, and interval 2 - from 64KB to 4MB - by triangles). For each interval on each number of measuring points, the graphs show five points: one per file size, increasing from left to right. Each result is the median from the errors for all tested requests size. We chose to use the median when summarizing error values because we believe it is a better central tendency estimator, since it is more resistant to outliers than the arithmetic mean.

We can see that most tests presented errors of up to approximately 20% (84% of the cases for sequential write, 86% for random write, 88% for sequential read, and 98% for random read). In general we did not observe improvements on the quality of the estimation by increasing the number of measuring points. In some cases, it is better to approximate the access time curve with only two points (the extremes of the interval) than including up to all points (Interval 1 has only 8 requests sizes). This happens because the real access time curve is not exactly a linear function, so slight variations from this linear format can impact the estimation, deviating it from the best approximation.

Since the values from the complete profiling are the average of several measurements (in order to provide statistical

Table 2: Configuration of the evaluated storage devices

Cluster	RAM	Storage Device			
		Type	Model	Capacity	Bus
Pastel	$8GB$	HDD	Hitachi HDS7225SC	$250GB$	SATA $1.5Gb/s$
Graphene	$16GB$	HDD	Hitachi HDS72103	$320GB$	SATA $3Gb/s$
Bali	$64GB$	HDD	Seagate ST91000640NS	$1TB$	SATA $6Gb/s$
Chimint	$16GB$	RAID-0	2×Seagate ST3300657SS-H	$2 \times 136GB$	SAS $3Gb/s$
Suno	$32GB$	RAID 0	2×Seagate ST9300653SS	$2 \times 300GB$	SAS $3Gb/s$
Taurus	$32GB$	RAID-0	2×Seagate ST3300657SS	$2 \times 300GB$	SAS $6Gb/s$
Edel	$24GB$	SSD	Micron C400	$64GB$	SATA $1.5Gb/s$
Graphite	$256GB$	SSD	Intel DC S3500 Series	$300GB$	SATA $6Gb/s$
Bali	$64GB$	SSD	SAMSUNG 840 Series MZ-7TD500BW	$500GB$	SATA $6Gb/s$

guarantees), the single benchmark executions made by our tool will hardly match them. We believe this inherent error is mostly responsible for the observed estimation error. This also helps to explain why including more points does not help estimation accuracy: more points potentially include more error.

From the graphs, we can observe that SSD and RAID devices were responsible for most of the cases where the error exceeded 40%. On the complete profiling, these devices took more repetitions in order to obtain statistical guarantees than the HDDs (17 repetitions on average were necessary for each test with HDDs, 43 with RAID arrays, and 26 with SSDs). Therefore, their results present more variability, and are harder to approximate. The machines with more variability were also the ones with incidence of high estimation errors (over 40%): Chimint (62 repetitions), Taurus (47), Bali-SSD (39), and Pastel (25).

Table 3: Median from estimation errors.

	Write	Random Write	Read	Random Read
HDD	10%	8%	10%	3%
RAID	10%	16%	9%	7%
SSD	17%	19%	5%	2%
All	9%	8%	7%	4%

Table 3 presents the medians of all values from the graphs of Figure 4.2 separated by type of storage device. Results show that RAID arrays' random access time curves are harder to approximate by linear functions than HDDs'. SSDs' write time curves are the hardest to estimate. On SSDs, maintenance operations may be executed by the controller during write operations, affecting performance. This phenomenon, known as "write amplification", causes higher variability between repetitions of tests in SSDs, making the approximation on these devices more difficult.

4.3 SeRRa's Results Evaluation

The previous section discussed the error induced by approximating access times curves with linear functions. The presented results show that increasing the number of measuring points (requests sizes to run the benchmarks with) does not necessarily increase the quality of the estimation. Because of that, all ratios obtained with our tool in this section used only 2 measuring points per interval. Instead of increasing execution time by running tests with more requests sizes, in this section we evaluate the number of repetitions of the benchmarks. The arithmetic average of multiple executions' results is taken for each measuring point (instead of the result of a single execution).

The graphs from Figure 4.3 present the absolute error obtained by using SeRRa to measure the sequential to random throughput ratio (compared to the ratios obtained from the complete profiling) with 2 points per interval and up to 4 repetitions of each measurement. For each number of repetitions, there are 5 results for each machine: one per file size increasing from left to right, similar to the graphs on Figure 4.2. We can see that most tests presented errors of up to $\approx 20\%$ (91% of the tests for write and 88% for read). The devices responsible for the cases with the highest errors were the same that presented the highest estimation errors on the previous section.

Table 4: Median of the errors with SeRRa.

	Write			Read		
	1 rep.	2 rep.	4 rep.	1 rep.	2 rep.	4 rep.
HDD	4.0%	5.7%	5.8%	3.4%	3.6%	3.3%
RAID	6.7%	8.2%	9.0%	10.3%	8.9%	8.4%
SSD	5.2%	3.6%	1.6%	1.5%	1.4%	2.0%
All	5.2%	5.9%	5.2%	4.9%	4.1%	3.6%

Table 5: SeRRa's time to measure (minutes).

	1 repetition	4 repetitions
Pastel (HDD)	93.21 (1/212)	376.99 (1/52)
Graphene (HDD)	85.69 (1/84)	341.50 (1/21)
Bali (HDD)	74.03 (1/102)	282.79 (1/26)
Chimint (RAID-0)	22.83 (1/181)	72.73 (1/57)
Suno (RAID-0)	28.12 (1/130)	109.42 (1/33)
Taurus (RAID-0)	16.03 (1/216)	64.54 (1/53)
Edel (SSD)	5.92 (1/409)	23.58 (1/102)
Graphite (SSD)	6.49 (1/562)	24.86 (1/146)
Bali (SSD)	5.79 (1/855)	21.52 (1/229)
Sum	338.10 (1/168)	1317.96 (1/43)

Table 4 presents the medians of all values presented on Figure 4.3 separated by type of storage device. In general, increasing the number of repetitions did not increase the results' accuracy significantly. It happened only for read tests on RAID arrays, and for write tests on SSDs. Despite SSDs presenting the highest estimation errors for write tests on the previous section, these devices presented the lowest ratio measurement errors. The time needed by SeRRa to obtain all sequential to random throughput ratios is presented in Table 5, along with its comparison to the time needed to obtain these ratios through the complete profiling discussed

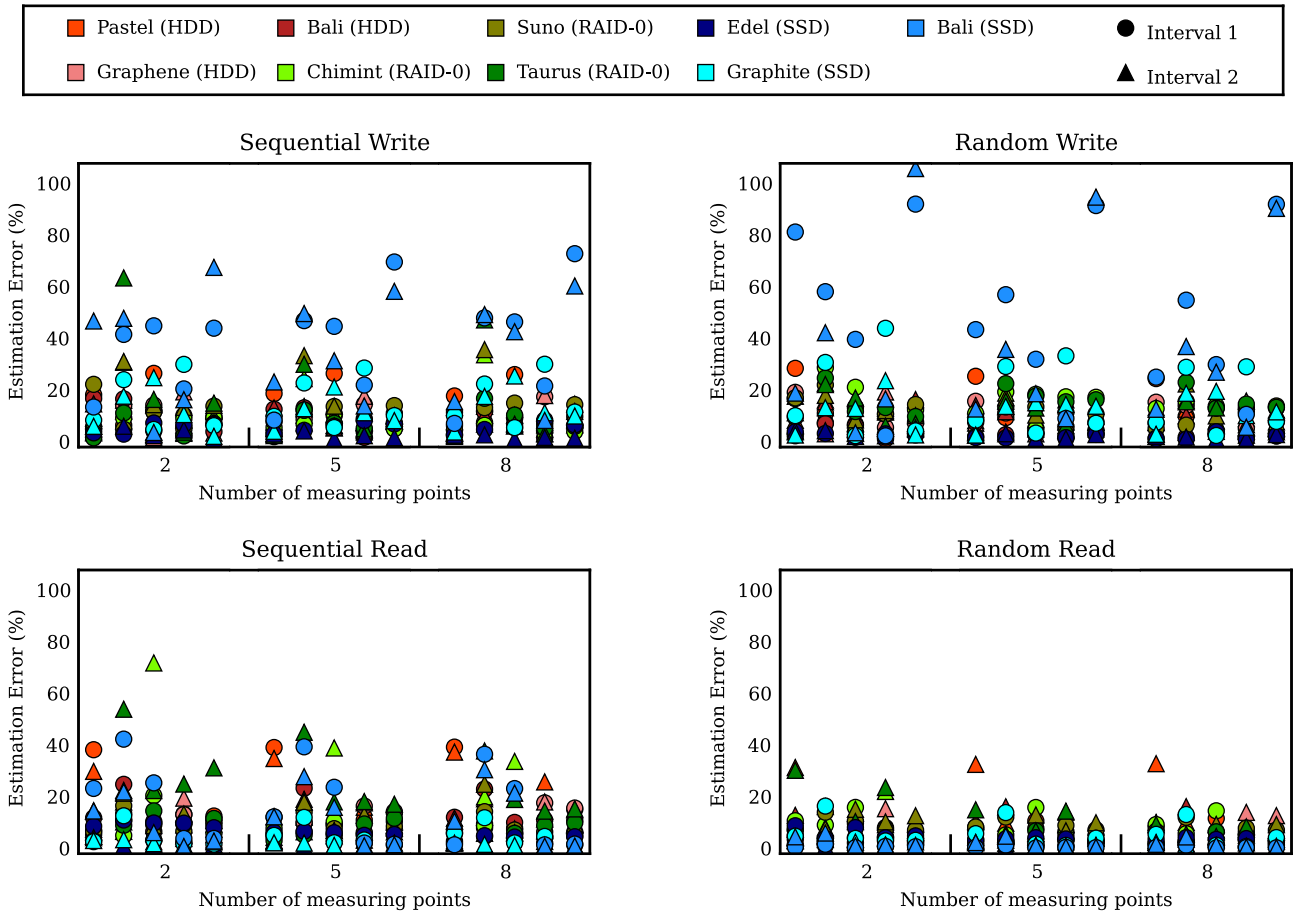

Figure 2: Absolute error induced by using linear regressions to estimate the access time curves.

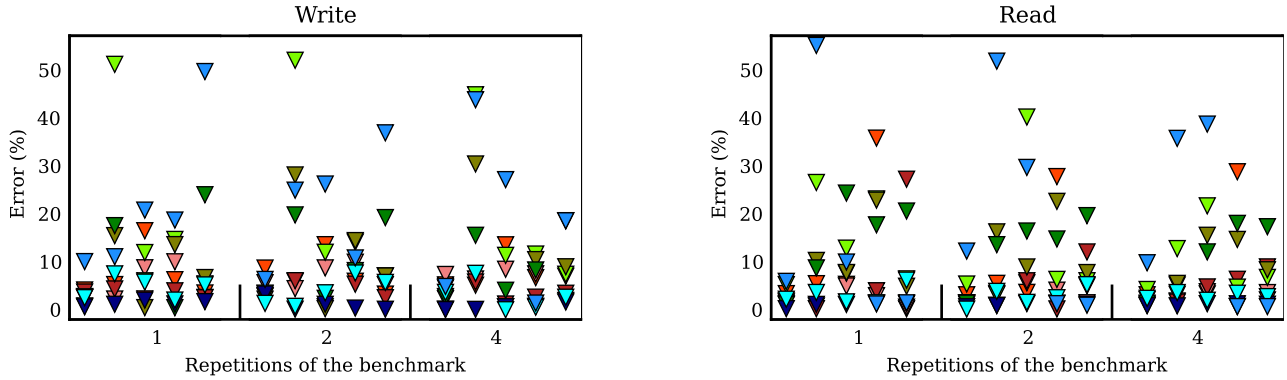

Figure 3: Absolute error induced by using SeRRa to obtain the sequential to random throughput ratio.

previously. With SeRRa, the slowest profiling (Pastel) takes 1.6 (with 1 repetition) or 6.3 (4 repetitions) hours instead of 14 days in order to provide a value whose error's median is around 5%.

We believe that more executions of the benchmark did not led to better results because 4 is still a small number of repetitions compared to what was necessary to provide the statistical guarantees defined on the complete profiling (from 13 to 62). In addition, increasing the number of executions can sometimes include outliers, perturbing the results. Since the decrease in time comes from both less repetitions of the tests and less requests sizes to be tested, running SeRRa to obtain this complete profiling with 62 repetitions, for instance, would be expected to take around 1/3 of the original time and provide results that are more accurate. The choice depends on the needed accuracy. We believe that a me-

Table 6: Sequential do random ratio with 1200MB files - measured vs. estimated with SeRRa (4 repetitions).

		Pastel (HDD)	Graphene (HDD)	Bali (HDD)	Chimint (RAID-0)	Suno (RAID-0)	Taurus (RAID-0)	Edel (SSD)	Graphite (SSD)	Bali (SSD)
Write	Measured	2.02	1.89	1.74	1.95	1.71	1.62	0.99	1.01	1.38
	SeRRa	1.98	1.75	1.68	1.80	1.55	1.32	0.97	1.04	1.64
Read	Measured	2.24	2.67	2.61	3.48	2.76	3.36	1.08	1.04	1.10
	SeRRa	2.19	2.57	2.84	3.25	2.53	2.78	1.07	1.06	1.09

dian error of approximately 5% (and up to 55%) would be enough for comparison between devices and optimizations' evaluation.

Table 6 illustrates SeRRa's results by showing the sequential to random throughput ratios obtained with it for $1200MB$ files and 4 repetitions. The values are compared with results from the complete profiling.

5. FINAL REMARKS

This paper presented a tool for storage devices profiling regarding access sequentiality named SeRRa. It quantifies the difference between accessing files sequentially and randomly for a given device. In order to obtain this information quickly, the benchmarks are executed on a small subset of the reported values and the remaining ones are estimated through a linear model.

Storage devices present different performance behaviors, and this behavior cannot be simply stated for whole classes of devices. On the other hand, optimizations that work to adapt applications' access patterns impose overheads that may not be compensated by its performance improvement. On this context, the information provided by SeRRa can be used to decide which I/O optimizations will be beneficial for performance for a given storage device. The tool is open source and available on the Internet.

We presented results for a large space of parameters on 9 different storage devices, comprising HDDs, SSDs and RAID arrays. Our evaluation shows that SeRRa provides ratios with median errors of approximately 5% while taking 1/168 of the originally needed time.

6. ACKNOWLEDGMENTS

This research has been partially supported by CNPq and CAPES-BRAZIL under the grants 5847/11-7 and Stic-Amsud 6132-13-8. The experiments presented in this paper were carried out on the Grid'5000 experimental test bed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see `https://www.grid5000.fr`). This research was accomplished in the context of the International Joint Laboratory LICIA and of the HPC-GA project.

7. REFERENCES

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy. Design tradeoffs for ssd performance. In *USENIX Annual Technical Conference*, pages 57–70, 2008.

[2] F. Z. Boito, R. V. Kassick, P. O. Navaux, and Y. Denneulin. Agios: Application-guided i/o scheduling for parallel file systems. In *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, pages 43–50, 2013.

[3] R. Bolze, F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.-G. Talbi, and I. Touche. Grid5000: A large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, 20(4):481–494, 2006.

[4] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger. The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101). *Parallel Data Laboratory*, page 26, 2008.

[5] F. Chen, D. A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 181–192. ACM, 2009.

[6] P. Desnoyers. Analytic modeling of ssd write performance. In *Proceedings of the 5th Annual International Systems and Storage Conference*, page 12. ACM, 2012.

[7] K. El Maghraoui, G. Kandiraju, J. Jann, and P. Pattnaik. Modeling and simulating flash based solid-state disks for operating systems. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pages 15–26. ACM, 2010.

[8] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar. Flashsim: A simulator for nand flash-based solid-state drives. In *Advances in System Simulation, 2009. SIMUL'09. First International Conference on*, pages 125–131. IEEE, 2009.

[9] W. D. Norcott and D. Capps. Iozone filesystem benchmark. *URL: www. iozone. org*, 2006.

[10] A. Rajimwale, V. Prabhakaran, and J. D. Davis. Block management in solid-state devices. In *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, USENIX'09, pages 21–21, Berkeley, CA, USA, 2009. USENIX Association.

[11] H. Shan and J. Shalf. Using ior to analyze the i/o performance for hpc platforms. Technical report, Ernest Orlando Lawrence Berkeley NationalLaboratory, Berkeley, CA (US), 2007.

[12] J. Yoo, Y. Won, J. Hwang, S. Kang, J. Choi, S. Yoon, and J. Cha. Vssim: Virtual machine based ssd simulator. In *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*, pages 1–14. IEEE, 2013.

[13] Y. Zhang and B. Bhargava. Self-learning disk scheduling. *IEEE Transactions on Knowledge and Data Engineering*, pages 50–65, 2008.