Faster Storage Devices Profiling with Parallel SeRRa

Jean Luca Bez, Francieli Zanon Boito, Rodrigo Virote Kassick,
Vinicius Rodrigues Machado, Philippe O. A. Navaux
Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre, Brazil
{jlbez, fzboito, rvkassick, vrmachado, navaux}@inf.ufrgs.br

Abstract

This work presents the parallel storage device profiling tool SeRRa. Our tool obtains the sequential to random throughput ratio for reads and writes of different sizes on storage devices. In order to provide this information efficiently, SeRRa employs benchmarks to obtain the values for only a subset of the parameter space and estimates the remaining values through linear models. The MPI parallelization of SeRRa presented in this paper allows for faster profiling. Our results show that our parallel SeRRa provides profiles up to 8.7 *times faster than the sequential implementation, up to* 895 *times faster than the originally required time (without SeRRa).*

1. Introduction

Hard Disk Drives (HDDs) have been the main non-volatile storage devices. For this reason, most systems were developed or adapted in order to maximize performance when accessing these devices [7].

Solid State Drives (SSDs) are a recent alternative to hard disks. These devices use flash memory to store data, which brings advantages [3]. Nonetheless, 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. Furthermore, hybrid solutions where both devices are used have been gaining popularity [6].

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.

Several assumptions about performance from HDDs do not hold when using SSDs and RAID arrays, and different requirements arise [5]. Therefore, we cannot simply classify optimizations 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 optimizations could be classified according to the *sequential to random throughput ratio* that devices must present in order to benefit from them.

However, obtaining this metric to a storage device can be a time-consuming task. In order to provide accurate results as fast as possible, SeRRa was developed in our previous work [1]. In this scenario, this paper proposes a parallel implementation of SeRRa. We evaluate our approach over clusters with HDDs, SSDs, and RAID arrays, and show performance improvements over the sequential version of the tool.

This paper is organized as follows: the next section presents the design of SeRRa. Section 3 details the parallel implementation. Section 4 discusses results obtained with the new SeRRa. Finally, Section 5 brings final remarks.

2. SeRRa: 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 storage devices. The main goals of SeRRa's project are performance - the information must be provided as quickly as possible - accuracy, and generality.

Keeping both performance and accuracy goals at the same time is a challenging problem because profiling a storage device adequately can take a long time. 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

	One execution	Total profiling
Pastel (HDD)	15.22	329.49
Graphene (HDD)	12.26	120.38
Suno (RAID-0)	4.21	61.32
Edel (SSD)	3.09	40.44

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

are time consuming, as the fastest profile took over 1.5 days, while the slowest one took almost 14 days.

The slow profiling whose times are presented in Table 1 consists of executing an I/O benchmark several times varying file sizes, request sizes and operation (sequential read, random read, sequential write, and random write). We have observed that most of the access times graphs (time to process a request as a function of the request size) 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.
2. **Benchmark:** the test is executed for the request sizes picked on the previous step. For this task, SeRRa uses the $IOzone$ benchmark [4].
3. **Linear Regression:** the complete set of access time, for each given interval, is estimated through linear regression based on the measurements from the previous step. 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 request sizes, as well as averages, maximum and minimum values. The estimated access times curves are also provided.

The tool, implemented on *Python*, is open source and available at `http://serratool.bitbucket.org/`.

3. Parallel SeRRa

Although SeRRa profiles a machine's storage device through its local file system, it can also be used in the context of a Parallel File System (PFS). The information provided by the tool can be used by I/O optimization techniques to improve performance when accessing those systems.

Therefore, optimization techniques can benefit from knowing how the PFS data servers' storage devices behave regarding access sequentiality. This information can be used, for instance, to decide between different I/O scheduling algorithms.

It is usual for HPC architectures to dedicate a set of nodes for the parallel file system deployment. This shared storage infrastructure is often homogeneous, with identical storage devices on all involved machines. These devices are expected to present the same performance behavior, and thus the same sequential to random throughput ratios.

For this reason, we have decided to create a parallel implementation of SeRRa which benefits from this characteristic to provide information faster. A faster profiling facilitates the tool's use for dynamic decision making.

SeRRa's parallel implementation, developed with MPI4PY[1], uses the master-slave paradigm for communication between processors. We have parallelized the step of benchmarking (step 2 described in the previous section), since it is the most time-consuming one. The master is responsible for all other steps, such as picking measuring points, linear regressions and reporting results. Additionally, the master sends tasks to slaves and receives their results until there are no tasks left. Each task corresponds to a request size to be tested with four access patterns: sequential write, random write, sequential read and random read.

Therefore, it makes no sense to use more processes than available machines, since this approach would lead to benchmarks being executed concurrently in the same node, compromising results. Furthermore, in this implementation the parallelism is limited by the number of intervals, measuring points per interval and benchmark repetitions.

4. Performance Evaluation

The four systems used in our tests are listed in Table 2. They were selected by their variety, aiming at representing most available devices. All of them are homogeneous clusters from *Grid'5000* [2].

The tests were executed on the *Linux* operating system, using *IOzone* version *3.397*. All tested devices were accessed through the *Ext3* 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).

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 request sizes lied within two ranges: from 8KB to 64KB with gaps of 8KB; and from 64KB to 4MB with gaps of 32KB.

1 http://packages.python.org/mpi4py

Cluster	RAM (GB)	Storage Device		
		Type	Capacity	Bus
Pastel	8	HDD	$250GB$	SATA $1.5Gb/s$
Graphene	16	HDD	$320GB$	SATA $3Gb/s$
Suno	32	RAID-0	$2 \times 300GB$	SAS $3Gb/s$
Edel	24	SSD	$64GB$	SATA $1.5Gb/s$

Table 2: Configuration of the evaluated storage devices

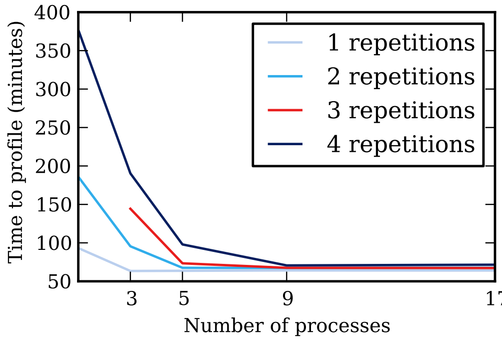
We have executed tests using two measuring points per interval, as this configuration was the one with the best results in our previous analysis [1]. We use up to four benchmark repetitions. Statistically, executing more repetitions of the benchmarks is expected to provide better results.

We compare the parallel implementation's results with SeRRa's sequential implementation and with the profile obtained without the tool. To obtain the latter, we have executed the complete profiling - without estimations, executing the benchmarks for all the specified request sizes - on all tests environments with the aforementioned parameters. 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.

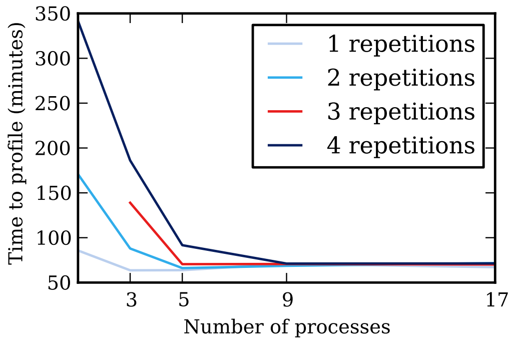
Table 3 presents the total time to perform all experiments described in the previous section by the different implementations. The numbers between parenthesis represent fractions of the originally required time (without SeRRa). The results with parallel SeRRa from the table were obtained using 17 processes (a master and 16 slaves). We show the times for sequential SeRRa with 1 and 4 benchmark repetitions. Repeating the benchmarks multiple times and taking the results' average allows for more accurate results. On the other hand, the time required to obtain a profile increases linearly with the number of repetitions.

We can see the parallel version's times for 4 benchmark repetitions were faster than (but close to) sequential SeRRa with 1 repetition. In other words, it is possible to obtain more accurate results using roughly the same profiling time.

Figure 1 presents the profiling time with SeRRa on the Pastel and Graphene clusters varying the number of benchmark repetitions. Graphs for Suno and Edel are similar to Graphene's and were omitted due to space constraints. The x axis represents the number of processes. The single-process case (the first point of each line) is the time obtained with the sequential implementation. Onward, the number of processes depicted includes the master plus the slaves – i.e. the number of processes that actually run benchmarks is the number of processes minus 1.



(a) Pastel cluster (HDDs)



(b) Graphene cluster (HDDs)

Figure 1: Time to profile disks with Parallel SeRRa

We can see that performance increases (the profiling time decreases) as we increase the number of processes. With few repetitions, performance reaches its best with few processes and does not profit from more nodes to run the profiling. This happens due to the parallelism limit of our experiment: using two intervals with two measuring points per interval, the total number of tasks is $2\,intervals \times 2\,points\ per\ interval \times N\,repetitions$. Therefore, using 2 benchmark repetitions we were expected to decrease the profiling time until 8 slaves (9 processes), and with 4 repetitions until 17 processes.

However, we can see that in most cases (in Pastel, Graphene, and Suno) there was no difference between the results with 4 benchmark repetitions using 9 or 17 processes. The same happened for results with 2 repetitions considering 5 or 9 processes. This indicates that the maximum speedup can be reached using a number of tasks which is twice the number of slave processes. This configuration leads to a better load balance, as it will be discussed later in this section.

Table 4 presents the observed speedups, comparing the best result from the parallel implementation with the sequential one. The speedup is not expected to be linear with

Cluster	No SeRRa	Sequential SeRRa		Parallel SeRRa
		1 repetition	4 repetitions	4 repetitions
Pastel (HDD)	19769.4	93.21 (1/212)	376.99 (1/52)	71.57 (1/276)
Graphene (HDD)	7222.8	85.69 (1/84)	341.50 (1/21)	70.75 (1/102)
Suno (RAID-0)	3679.2	28.12 (1/130)	109.42 (1/33)	21.56 (1/171)
Edel (SSD)	2426.4	5.92 (1/409)	23.58 (1/102)	2.72 (1/892)
Sum	33097.8	212.94 (1/155)	851.49 (1/39)	166.6 (1/199)

Table 3: Time to profile in minutes. The fractions compare SeRRa's results with the originally required time (without SeRRa).

the number of processes, since the tasks distributed to the nodes are of different sizes: considering a fixed file size (as SeRRa does), it takes longer to execute the tests with smaller request sizes. In other words, the parallel implementation profiling time will be limited by the slowest test.

Benchmark repetitions	**Pastel (HDD)**	**Graphene (HDD)**	**Suno (RAID-0)**	**Edel (SSD)**
1	1.47	1.37	1.31	2.29
2	2.76	2.59	2.6	4.51
3	2.15	1.98	2.18	3.31
4	5.34	4.83	5.08	8.71

Table 4: Speedup provided by SeRRa's parallel implementation (with the best number of processes to each case).

We can observe from Table 4 that speedups observed for the Edel cluster are the highest. This happens because Edel's devices present the lowest sequential to random throughput ratios. When storage devices present a high sequential to random throughput ratio, a task which has to perform many small, random accesses to access a fixed-size file will take much longer than a task which accesses a file of the same size with larger, random or sequential requests. This will create a situation of task imbalance which can impair speedup due to longer execution times.

The load imbalance explains why the best performance was achieved having a number of processes which is half the number of tasks for Pastel, Graphene, and Suno; and why this did not happen for Edel, where the difference between the tasks is smaller due to lower sequential to random throughput ratios. Having more available tasks than processes allows for better load balancing. Therefore, one possible solution to avoid the imbalance problem would be to break the tests in smaller units.

5. Final Remarks

This paper presented a parallel implementation of a tool for storage devices profiling regarding access sequentiality named SeRRa. Decreasing SeRRa execution time is important because it facilitates its use by I/O optimizations to dynamically adapt to storage devices' characteristics. For this reason, we have developed a parallel implementation with MPI following a master-slave paradigm. This approach is adequate for homogeneous storage nodes.

We have evaluated our approach with four clusters, using HDDs, RAID arrays and SSDs. Our results show performance improvements (decreases in the profiling time) of up to 8.71 times with the parallel implementation of SeRRa over the sequential one, up to 895 times over not using SeRRa.

References

[1] F. Z. Boito, R. V. Kassick, P. O. Navaux, and Y. Denneulin. Towards fast profiling of storage devices regarding access sequentiality. In *Symposium on Applied Computing (SAC)*, Salamanca, Spain, 2015. ACM.

[2] 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.

[3] F. Chen, D. A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *11th International Joint Conference on Measurement and Modeling of Computer Systems*, pages 181–192, New York, NY, USA, 2009. ACM.

[4] W. D. Norcott and D. Capps. *Iozone filesystem benchmark*, 2006. Available at www.iozone.org. Accessed in March 2014.

[5] A. Rajimwale, V. Prabhakaran, and J. D. Davis. Block management in solid-state devices. In *Proceedings of the USENIX Annual Technical Conference*, pages 279–284, San Jose, CA, USA, 2009. USENIX Association.

[6] W. Xu, Y. Lu, Q. Li, E. Zhou, Z. Song, Y. Dong, W. Zhang, D. Wei, X. Zhang, H. Chen, et al. Hybrid hierarchy storage system in milkyway-2 supercomputer. *Frontiers of Computer Science*, 8(3):367–377, 2014.

[7] Y. Zhang and B. Bhargava. Self-learning disk scheduling. *IEEE Transactions on Knowledge and Data Engineering*, 21(1):50–65, 2008.