Hadoop Energy Consumption Reduction with Hybrid HDFS

Ivanilton Polato
Dept. of Computer Science
Federal University of Technology – Paraná
Campo Mourão, PR, Brazil
ipolato@utfpr.edu.br

Denilson Barbosa, Abram Hindle
Dept. of Computer Science
University of Alberta
Edmonton, AB, Canada
{denilson, abram.hindle}@ualberta.ca

Fabio Kon
Dept. of Computer Science
University of São Paulo
São Paulo, SP, Brazil
fabio.kon@ime.usp.br

ABSTRACT

Apache Hadoop has evolved significantly over the last years, with more than 60 releases bringing new features. By implementing the MapReduce programming paradigm and leveraging HDFS, its distributed file system, Hadoop has become a reliable and fault tolerant middleware for parallel and distributed computing over large datasets. Nevertheless, Hadoop may struggle under certain workloads, resulting in poor performance and high energy consumption. Users increasingly demand that high performance computing solutions address sustainability and limit energy consumption. In this paper, we introduce $HDFS_H$, a hybrid storage mechanism for HDFS, which uses a combination of Hard Disks and Solid-State Disks to achieve higher performance while saving power in Hadoop computations. $HDFS_H$ brings to the middleware the best from HDs (affordable cost per GB and high storage capacity) and SSDs (high throughput and low energy consumption) in a configurable fashion, using dedicated storage zones for each storage device type. We implemented our mechanism as a block placement policy for HDFS, and assessed it over six recent releases of Hadoop with different architectural properties. Results indicate that our approach increases overall job performance while decreasing the energy consumption under most hybrid configurations evaluated. Our results also showed that, in many cases, storing only part of the data in SSDs results in significant energy savings and execution speedups.

CCS Concepts

•**Hardware** → **Power estimation and optimization;** •**Computing methodologies** → *Parallel computing methodologies;*

Keywords

Hadoop; HDFS; Hybrid Storage; Energy Consumption;

SAC 2016,April 04 - 08, 2016, Pisa, Italy
Copyright 2016 ACM 978-1-4503-3739-7/16/04…$15.00
http://dx.doi.org/10.1145/2851613.2851623

1. INTRODUCTION

Currently there are 2 predominate perspectives relevant for big data analysis: the 3 "Vs" – volume, variety, and velocity [13, 26]; and hardware and software infrastructures capable of processing all the collected data. These processing infrastructures now encounter new performance challenges: energy consumption, power usage, functionality, and environmental impact. Over the last years, the volume and speed of data creation consistently increased. A recent study estimates that 90% of all data in the world was generated over the last two years [1]. The International Data Corporation predicted that from 2005 to 2020, the digital universe will grow by a factor of 300, from 130 exabytes to 40,000 exabytes [22]. The same study also predicted that the "digital universe" will roughly double and the storage market will grow 55% every two years. As a result, they expect that the discovery and analytics software market will grow 33% until 2016, leading to an 8 billion-dollar business [22]. In terms of infrastructure, the storage server market has benefited from continually decreasing disk prices and higher performance solid state drives. Over the last three years, the cost for hard disks has ranged between $0.03 and $0.05 per GB [14], with SSDs costing around 20 times more.

Within this context, this paper presents the results of a hybrid storage approach for the Hadoop Distributed File System (HDFS), called $HDFS_H$, which seamless integrates both storage technologies – HDs and SSDs – to create a highly-efficient hybrid storage system. Our results indicate that in most configurations this approach promotes overall job performance increase, while decreasing the energy consumption. The key contributions of this paper are: **(1) An adaptable hybrid storage approach for HDFS that takes into account the performance profiles of HDs and SSDs.** Since SSDs are faster and use less power than HDs but are much more expensive, these characteristics must be adequately treated for different workloads; **(2) An HD- and SSD-aware block placement policy that is optimized for heterogeneous storage.** This policy was designed to accommodate a pre-defined percentage of the total number of blocks in the SSDs, and the remainder in the HDs; and **(3) An evaluation of the technique over multiple versions of Apache Hadoop.** Our research showed that each Hadoop branch has a unique energy consumption profile; we detail these results and show how our system behaves in each situation. We show that regardless of Hadoop architecture/version, $HDFS_H$ provides improvements over the default HDFS settings.

Our hybrid storage model splits the file system into storage zones, wherein a block placement strategy directs file blocks to zones according to predefined rules. This enables the use of different storage configurations for different workloads, thereby achieving the desired tradeoff between performance and energy consumption. For now, our goal is to allow the user to choose the best configuration for the available infrastructure, by setting how much of each storage devices should be used during MapReduce [2, 4] computations. Although the cost of SSDs could still be considered prohibitive for general storage purposes, they can provide unique performance enhancements to data analysis by reducing energy consumption if they receive adequate support from processing platforms.

Our motivation is that as data analysis increases and expands so does the associated energy consumption. The number of data centers has consistently grown, increasing the availability of computing nodes and storage space, and demanding more power. Data centers' maintenance costs and environmental impacts have consistently increased with the demand for more energy to power and cool them. In fact, energy accounts for 30% of the Total Cost of Ownership (TCO), a major and continuous cost for data centers [5]. This makes energy consumption as one of the most important topics regarding big data processing.

2. APACHE HADOOP AND HDFS

Hadoop was developed based on Google's MapReduce parallel approach [2,4], using the same ideas: hiding complexity from users and thereby allowing them to focus on programming the paradigm's two primitive functions, Map and Reduce. Hadoop uses the HDFS file system [20], a block direct storage system capable of storing, managing, and streaming large amounts of data in a reasonable time to user applications. As mentioned earlier, most HDFS releases lacks differentiation of the different storage devices attached to a Hadoop cluster node; consequently, they cannot properly exploit the features provided by such devices to decrease a cluster's energy consumption or increase job performance in a custom fashion. Our approach tackles this specific issue, creating storage zones according to the device types connected to the cluster nodes. To the best of our knowledge, this is a novel approach to represent and manage storage space for Hadoop jobs.

Throughout its history, Hadoop experienced more than 60 releases in several development branches. As of now, Hadoop has three main development branches: 1.x, 0.23.x and 2.x. Our research focused on recent releases of these branches. We reported on the genealogy tree of the Hadoop project using release logs from each project branch and its releases presenting the project's evolution, from version 0.20.0 up to the latest releases [17].

3. HDFS HYBRID STORAGE

The HDFS_H approach controls the number of blocks that are kept in each storage zone using a block placement policy. At first, we create two independent storage zones using the HDs ($HDzone$) and SSDs ($SSDzone$). The policy sends blocks to each storage zone guaranteeing that a predefined percentage of the total number of blocks is stored in the SSDs and the rest in the HDs. To avoid bias in the distribution, the policy uses a round-robin fashion list to distribute evenly the blocks between the zones. This assures that each zone receives blocks from every portion of the dataset files. Due to space restrictions, the complete model will not be presented, focusing on the results obtained during the experimentation phase.

4. EXPERIMENTAL METHODOLOGY

With HDFS_H and the block placement rules properly designed, we then selected the Hadoop releases and benchmarks for the experiments. Six releases from the current branches were selected: 1.x (1.1.1 and 1.2.1), 0.23.x (0.23.8 and 0.23.10) and 2.x (2.3.0 and 2.4.0). The 0.23.x and 2.x branches include the YARN resource manager. The difference between them is that 2.x releases include the High Available NameNode for HDFS – an effort focused on the automatic failover of the NameNode – whereas the 0.23.x releases exclude such a feature. The 1.x Hadoop releases do not include the YARN feature and have the limitation that they are more tightly coupled to the MapReduce paradigm and mostly designed to run batch jobs, making them less flexible than the other two branches.

The testing infrastructure is a 9-node commodity cluster (Quad-core processor; 8GB of RAM; 1TB HD; 120GB SSD; Red Hat (4.4.7-4) GNU/Linux). One node is dedicated to the Hadoop NameNode and JobTracker daemons and the other eight nodes run Hadoop DataNodes and TaskTracker daemons. For the energy consumption measurements, we instrumented the cluster with four *Watts Up? Pro*, connecting the DataNodes in pairs on each one of the power meter devices.

To cover different situations, our final selection includes the following benchmarks and dataset sizes: Sort (I/O-bound): 10, 48, and 256GB; Join (CPU-bound): 20GB; and K-Means Clustering (CPU- & I/O-bound) from HiBench[7]: 3×10^7 samples. Even though our approach focuses on storage, we performed experiments using CPU-bound benchmarks to analyze the behavior of the hybrid storage under these workloads.

Each benchmark uses a specific dataset that was generated and stored for experimental reuse and replication. For the Sort experiments, the datasets were generated using the *RandomWriter* job. The Join benchmark performs a join between two datasets, in a database fashion. These experiments used datasets generated with DBGEN from the TPC-H benchmark [21]. Finally, we used an implementation of the K-Means clustering algorithm using the Mahout Library from HiBench in our experiments. We chose our experimentation benchmarks based not only on their I/O or CPU characteristics, but also on prior research [7, 16].

For each benchmark, we ran a batch job as follows: for small datasets, a 10-job batch; and for medium and large ones, a 5-job batch. Each selected Hadoop release ran one batch for each benchmark/dataset pair. A configured daemon records one reading every second from each power meter on separate files. To demonstrate the use of our block placement policy, we set five predefined proportions to split the data into the storage zones. The first one, named HD, keeps all the data in the $HDzone$ and does not use the $SSDzone$. The three intermediate ones vary the amount of data stored in each zone: 80% in the $HDzone$ and rest in the $SSDzone$, named 80/20; the 50/50 configuration uses an equal distribution between zones; the 20/80 keeps most of the data on the $SSDzone$ (80%). The last one (SSD) uses only the $SSDzone$ to store the data.

5. RESULTS AND ANALYSIS

We are particularly concerned with the impact of hybrid storage systems on energy consumption. Our first finding was the large difference in both performance and power needs among different Hadoop releases. Due to architectural changes in the middleware, Hadoop releases that included the YARN resource manager performed worst and consumed more energy when compared to the 1.x releases. In the following, we detail our experimental results. In terms of performance, we consider makespan as the time difference between the start and conclusion of Hadoop jobs.

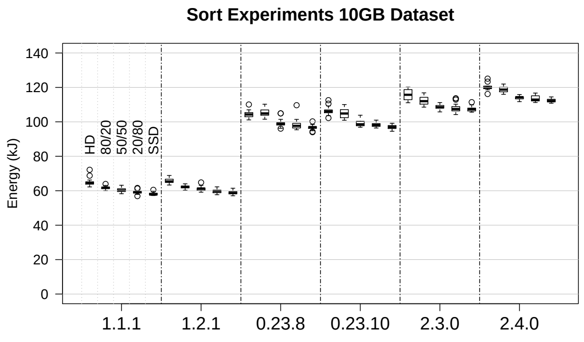

Figure 1: Energy Consumption: Sort 10GB

5.1 Results On I/O-Bound Benchmark

Starting with the 10GB Sort benchmark, Figure 1 shows the differences between two $HDFS_H$ configurations: HD and SSD. The differences among the three branches are easily identified. Whereas releases 0.23.x consumed on average 65% more energy than releases 1.x, releases 2.x consumed on average 85% more energy than the 1.1.1 and 1.2.1 releases. This increase is partially explained by the performance loss: jobs running on 0.23.x releases were 27% slower than on 1.x releases. The same is observed in the 2.x branch, which was around 35% slower than 1.x releases, as Figure 2 illustrates. The significant difference in energy consumption can also be explained by the introduction of new functionality in recent Hadoop branches. The YARN component brought flexibility to the framework, allowing other types of jobs to be executed, in addition to the original MapReduce. YARN also enabled the instantiation of multiple JobTrackers and NameNodes. Our experiments demonstrated that all this flexibility comes at a price: loss in performance and an even larger increase in energy consumption when executing MapReduce jobs.

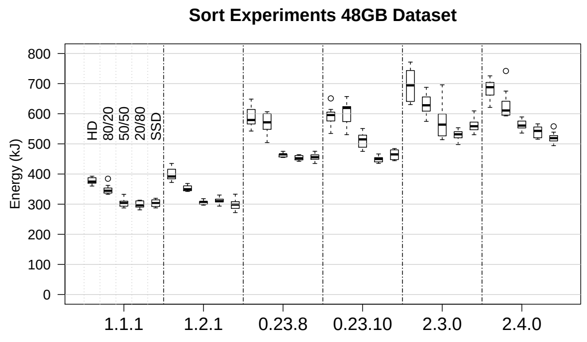

Figure 3: Energy Consumption: Sort 48GB

Further considering the 10GB Sort experiments, Figure 1 presents the results for all the releases using the five configurations we tested: HD, 80/20, 50/50, 20/80, and SSD. We can also observe, in Figure 1, the expected tendency in energy savings when moving data to the configurations that favor SSD use.

Next, we moved on to the experiments with larger datasets. Figure 3 shows the results for the 48GB Sort experiments. The results support the tendency towards energy savings when using SSD. Analyzing these two initial experiments, we noticed that increasing the dataset size shifts the tendency of power saving towards the balanced configurations: 50/50 and 20/80. This indicates that, by storing only a fraction of the data on SSDs with these specific hybrid con-

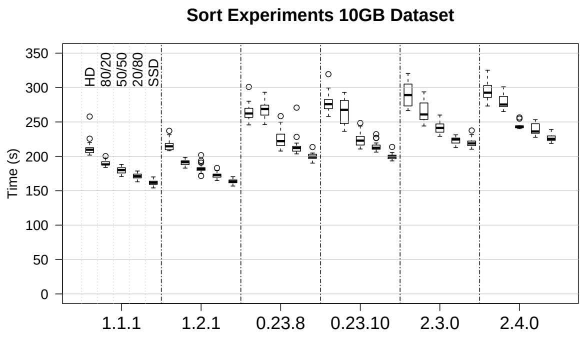

Figure 2: Job Makespan: Sort 10GB

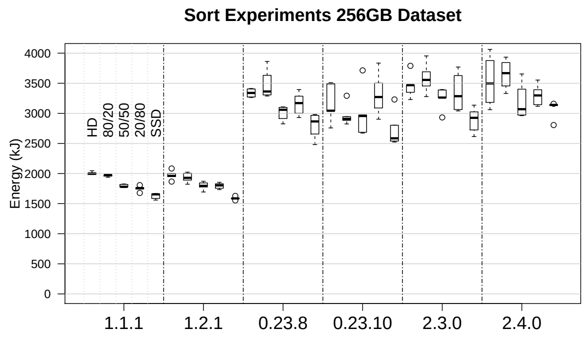

Figure 4: Energy Consumption: Sort 256GB

Table 1: Sort Benchmarks: Energy consumed (kJ)

Dataset size	Release	HD	80/20	50/50	20/80	SSD
10GB	1.1.1	64	62	60	59	58
	1.2.1	66	62	61	60	59
	0.23.8	108	105	99	98	97
	0.23.10	106	105	99	98	97
	2.3.0	117	112	109	108	107
	2.4.0	120	119	114	113	112
48GB	1.1.1	378	348	303	299	304
	1.2.1	398	352	306	311	298
	0.23.8	588	566	463	454	456
	0.23.10	593	596	510	449	464
	2.3.0	696	630	579	529	564
	2.4.0	682	631	563	540	521
256GB	1.1.1	2005	1967	1795	1750	1626
	1.2.1	1972	1934	1796	1795	1588
	0.23.8	3339	3469	3001	3157	2790
	0.23.10	3168	2971	3000	3320	2736
	2.3.0	3461	3587	3248	3345	2885
	2.4.0	3530	3650	3212	3301	3077

figurations, we achieve a significant increase in performance and reduction in energy consumption. Our results indicate that, if all data is processed from the *SSDzone*, there is an average reduction of 20% in energy consumption. Additionally, a similar reduction can also be observed in the 50/50 and 20/80 configurations in Figures 1 and 3. The energy consumption results from the Sort benchmarks are listed in Table 1.

The results from the Sort benchmark using the 256GB dataset (Figure 4) also corroborate the previous statements. With the increase in the dataset size, the energy consumption rates decrease in the balanced configurations, favoring the use of less SSD storage to achieve results similar to the SSD-only configurations. Table 1 presents these results. The energy savings when using more SSD storage is clearly noticeable.

5.2 Results On CPU-Bound Benchmarks

To evaluate whether our approach favors I/O-bound jobs only, we performed two sets of experiments using CPU-bound jobs. The differences between the hybrid HD and SSD configurations were insignificant, thus we present only the results for the two extreme configurations using the Join and K-Means benchmarks, since the intermediate results did not present anything new. Except for the differences between branches, the Join benchmark did not present any significant difference among all the 5 configurations tested, as seen in Figure 5, presenting only the HD and SSD ones.

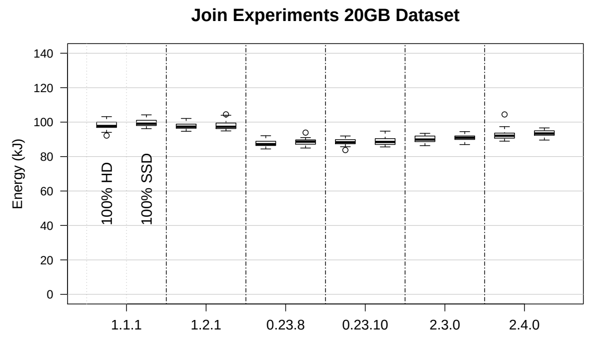

Figure 5: Energy Consumption: Join 20GB

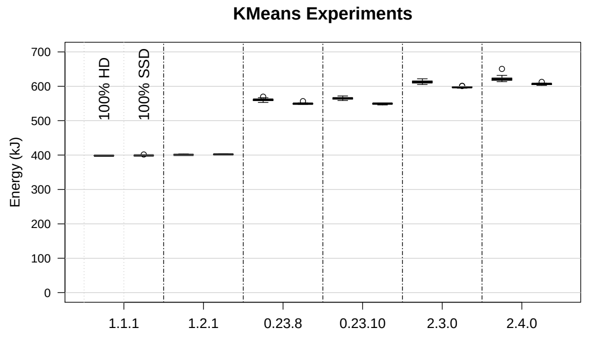

Figure 6: Energy Consumption: K-Means

The Mahout K-Means is a hybrid benchmark that is CPU-bound in the iterations, and I/O-bound in clustering. With our setup (3 iterations), 3/4 of the execution in this benchmark was CPU-bound, while the rest was I/O-bound. As can be seen in Figure 6, again, there is no significant difference between the configurations, except for the differences among branches. We thus conclude that, except for the difference between branches and releases, there are no significant differences in terms of performance and energy consumption when running CPU-bound jobs with our approach.

5.3 Speedup

Regarding job makespan performance, we confirmed the hypothesis that storing more data in the SSDs enabled the jobs to run faster, since SSDs provide higher throughput. The novelty here is the non-linear behavior of the job makespan as we increase the dataset size. With the 10GB dataset, the 80/20 configuration was on average 7% faster than the HD configuration, with only 20% of the data processed from the SSD; in the 50/50 configuration, jobs were on average 17% faster than the HD configuration; and, in the 20/80 configuration, 22% faster; finally, jobs running with the SSD-only configurations were on average 26% faster than purely running on HD. In the 48GB Sort, the observed speedups compared to the HD configuration were: 80/20, 8% faster; 50/50, 27% faster; 20/80, 31% faster; and SSD, 30% faster. The average makespan from the Sort benchmarks can be seen in Figure 7.

5.4 Time versus Power and Cost

On a single computer, typically, time and energy are related since energy is defined as the integration of power over time ($e = p \cdot t$). Thus, we conducted an analysis of the relation between time and energy, and asked the following research question "Is energy consumption explained completely by execution time?", since Hadoop tasks run on multiple computers. Regardless of the Sort dataset size or configuration, over all tests, a Pearson correlation of 0.9884 was achieved, indicating high linear correlation between time and energy, as expected. The novelty is that releases from 0.23.x and 2.x branches demand much more energy compared to 1.x releases, even though their job makespans are close. This has a great influence on the smaller sort benchmarks, observed in Figures 1 and 2, and on Figures 3 and 7 in the 48GB results. Therefore, besides time, there is an external factor playing in the energy consumption in the 0.23.x and 2.x releases. This influence increases even more if we consider the Sort 256GB results (Figures 3 and 7).

Additionally, besides energy and time, cost must play as a role in our approach. Assuming that the $HD_{cost/GB} = \$0.05$ and the $SSD_{cost/GB} = \$1.00$ (20 times larger), we plotted the ratio between job makespan and job storage cost for the sort jobs, using the tested releases. We observed, in Figure 8, that, for each release, the general behavior is a pareto-optimal configuration. The plot shows a clear trade-off between hardware cost and performance: the more SSD is used, the more expensive the hardware cost is, but with less hours spent to perform the job, less energy is used. On the other hand, by using more HD in the configurations, jobs take more hours to finish, demanding more energy, but the hardware cost decreases greatly. This behavior can be consistently observed in three key configurations: HD, 50/50, and SSD, marked with rectangles in Figure 8. Consequently, the 50/50 configuration is the one that optimally shows the key point in our analysis for every tested release: with a storage composed of 50% of SSDs, we can achieve results with a performance close to the use of SSD-only configurations with a fraction of the cost of using only SSDs. Yet, this approach guarantees a decrease in energy consumption, as presented before. This behavior can be observed in every release on the 10GB (omitted in the plot), 48GB and 256GB datasets tested in the Sort experiments. Furthermore, the plot shows once more the clear difference among 1.x, 0.23.x, and 2.x releases in performance, and the major difference when scaling up the dataset sizes.

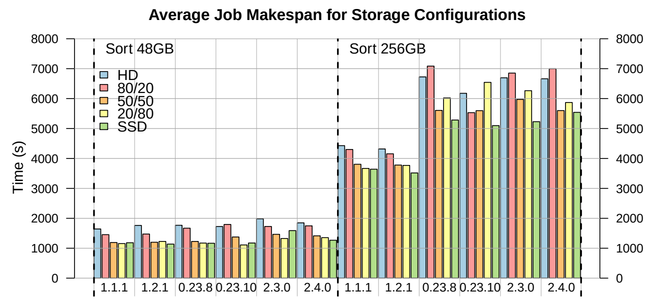

Figure 7: Hadoop performance: multiple releases with different configurations

6. RELATED WORK

A comprehensive literature search [18] indicated that HDFS has been modified to increase its performance in multiple ways: tuning the I/O mode [9, 19], solving data placement problems [24], and adapting it to support small files processing [3, 9], since HDFS was not originally designed for such purposes. Yet, these approaches only targeted the HDFS' performance without considering energy consumption or using a combination of different storage devices.

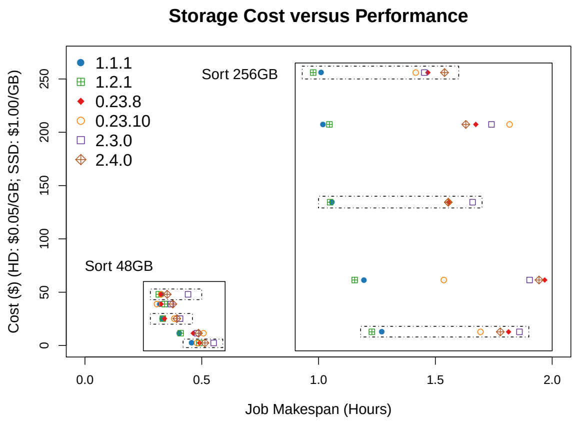

Figure 8: $HDFS_H$ storage cost versus performance

The use of SSDs as a storage solution on clusters is such a recent trend in the industry that the first proposals for MapReduce clusters only recently appeared. Most of the research up to date tends to analyze whether MapReduce can benefit, in terms of performance, when deploying HDFS using SSDs [8, 12]. Other researchers focus on incorporating SSDs into HDFS using caching mechanisms to achieve better performance [10, 23, 25]. A few works also discuss SSDs' impact on Hadoop [11, 15], sometimes focusing on using SSDs as the sole storage device under HDFS. Our approach tends to be more affordable, since we developed a hybrid file system that seamlessly couples the best from HDs (affordable cost per GB, high storage capacity, and, to some extent, endurance) and SSDs (high performance and low energy consumption rates) in a configurable fashion. In [17], we applied the Green Mining methodology [6] analyzing the projects' source code in several dimensions, correlating the results presented in this paper with the architectural changes throughout the development branches, corroborating the results that later versions of Hadoop can suffer from serious energy consumption performance regressions.

The Hadoop team of developers implemented an approach to incorporate hybrid storage into the HDFS. On the latest releases – 2.7.0 and 2.7.1 – the project made available the implementation of new storage policies. Two of the new policies make use of SSDs, but contrary to our approach, they only allow the use of all data on SSDs (All_SSD); or the use of one block replica stored in the SSDs, keeping the other replicas in the HDs (One_SSD). To the present, these newly implemented policies do not allow the controlled hybrid storage of blocks guaranteeing the uniform distribution of blocks over HDs and SSDs as we have presented. Also, there are no mentions of performance increase or energy consumption testing of such policies.

7. CONCLUSIONS AND FUTURE WORK

We presented an approach to seamlessly integrate HD and SSD technologies into $HDFS_H$. Our approach shows reduction in energy consumption even when only a fraction of the data is stored in the modified HDFS $SSDzone$. We showed that, with larger datasets, the reduction in energy demand can be significant, achieving up to a 20% savings under certain hybrid configurations. The general use of $HDFS_H$ affords immediate benefits since it increases MapReduce jobs' performance and reduces energy consumption. Yet, for now, users must manually define these configurations, but we encourage users to insert 50% of SSD storage space in the Hadoop cluster as a mean to increase performance by reducing the job makespan, and more important as a mean to achieve reduction in the overall energy consumption. Moreover, these benefits can come at a fraction of the cost of using only SSDs as storage space in Hadoop clusters. As future work, we intend to investigate autonomic heuristics that would analyze the workflow to automatically and dynamically configure $HDFS_H$ for optimal performance and energy consumption.

Internal and external validity are threatened by the focus on the Hadoop project. What happens to Hadoop might not happen to other projects thus we cannot generalize much beyond the Hadoop project. Furthermore not all Hadoop releases were tested. The range of workloads threatens external validity but is tempered by the focus on I/O and CPU bound workloads. External validity is threatened by the scale and the limitations of our 9-node infrastructure as our datasets are limited to less than 1TB by 8 120GB SSDs. Additionally, hybrid storage device, known as SSHD, which combines in the same device both SSD and HD technologies were not tested in our approach.

This research was funded by CAPES, Fundação Araucária, CNPq proc. 305566/2013-0, and by the Emerging Leaders in the Americas Program (ELAP) from the Government of Canada.

8. REFERENCES

[1] P. B. Brandtzæg. Big data - for better or worse, May 2013.

[2] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In *6th OSDI*, volume 6, pages 10–10, Berkeley, CA, USA, 2004. USENIX.

[3] B. Dong et al. A novel approach to improving the efficiency of storing and accessing small files on Hadoop: A case study by PowerPoint files. In *IEEE SCC*, pages 65–72. IEEE, 2010.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google File System. *ACM SIGOPS OSR*, 37(5):29–43, 2003.

[5] J. Hamilton. Overall data center costs. http://perspectives.mvdirona.com/2010/09/, 2010.

[6] A. Hindle. Green mining: a methodology of relating software change and configuration to power consumption. *Empirical Software Engineering*, pages 1–36, 2013.

[7] S. Huang et al. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *IEEE ICDEW*, pages 41–51, March 2010.

[8] H. Jeon, K. El Maghraoui, and G. B. Kandiraju. Investigating hybrid ssd ftl schemes for hadoop workloads. In *ACM CF*, pages 20:1–20:10, New York, NY, USA, 2013. ACM.

[9] D. Jiang et al. The performance of MapReduce: an in-depth study. *Proceedings of the VLDB Endowment*, 3(1-2):472–483, Sept. 2010.

[10] K. Krish et al. Venu: Orchestrating ssds in hadoop storage. In *IEEE Big Data*, pages 207–212, Oct 2014.

[11] K. Kambatla and Y. Chen. The truth about mapreduce performance on ssds. In *LISA14*, pages 118–126. USENIX, 2014.

[12] Y. Kang et al. Enabling cost-effective data processing with smart ssd. In *IEEE MSST*, pages 1–12, 2013.

[13] D. Laney. 3D data management: Controlling data volume, velocity, and variety. Technical report, META Group, February 2001.

[14] J. C. MacCallum. Disk drive prices. http://www.jcmit.com/diskprice.htm, 2015.

[15] S. Moon, J. Lee, and Y. S. Kee. Introducing ssds to the hadoop mapreduce framework. In *IEEE CLOUD*, pages 272–279, June 2014.

[16] F. Pan, Y. Yue, J. Xiong, and D. Hao. I/O characterization of big data workloads in data centers. In J. Zhan, R. Han, and C. Weng, editors, *Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, volume 8807 of *LNCS*, pages 85–97. Springer International Publishing, 2014.

[17] I. Polato, D. Barbosa, A. Hindle, and F. Kon. Hadoop branching: Architectural impacts on energy and performance. In *6th International Green & Sustainable Computing Conference*. IEEE, 2015.

[18] I. Polato, R. Ré, A. Goldman, and F. Kon. A comprehensive view of Hadoop research - A systematic literature review. *Journal of Network and Computer Applications*, 46:1 – 25, 2014.

[19] J. Shafer, S. Rixner, and A. Cox. The Hadoop Distributed Filesystem: Balancing portability and performance. In *IEEE ISPASS*, pages 122–133, 2010.

[20] K. Shvachko et al. The Hadoop Distributed File System. In *IEEE Storage*, pages 1–10, Washington, DC, USA, 2010. IEEE.

[21] Transaction Processing Performance Council. TPC Benchmark H (Decision Support) Standard Specification, Jun 2002.

[22] D. Vesset, A. Nadkarni, C. W. Olofson, and D. Schubmehl. Worldwide big data technology and services 2012-2016 forecast. Technical report, IDC Corporate USA, 2012.

[23] B. Wang et al. mpCache: Accelerating mapreduce with hybrid storage system on many-core clusters. In *Network and Parallel Computing*, volume 8707 of *LNCS*, pages 220–233. Springer, 2014.

[24] J. Xie et al. Improving MapReduce performance through data placement in heterogeneous Hadoop clusters. In *International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, pages 1–9. IEEE, 2010.

[25] D. Zhao and I. Raicu. Hycache: A user-level caching middleware for distributed file systems. In *IEEE IPDPSW*, pages 1997–2006, 2013.

[26] P. Zikopoulos and C. Eaton. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. Mcgraw-hill, 2011.