Machine Learning based Performance Modeling of Flash SSDs

Jaehyung Kim
Department of Computer Science
Yonsei University
Seoul, Republic of Korea
jaehyungkim@yonsei.ac.kr

Jinuk Park
Department of Computer Science
Yonsei University
Seoul, Republic of Korea
parkju536@yonsei.ac.kr

Sanghyun Park†
Department of Computer Science
Yonsei University
Seoul, Republic of Korea
sanghyun@yonsei.ac.kr

ABSTRACT

Flash memory based solid state drives(SSDs) have alleviated the I/O bottleneck by exploiting its data parallel design. In an enterprise environment, Flash SSD used in the form of a hybrid storage architecture to achieve the better performance with lower cost. In this architecture, I/O load balancing is one of the important factors. However, the internal parallelism distorts the performance measures of the flash SSDs. Despite the criticality of load balancing on I/O intensive environments, these studies have rarely been addressed. In this paper, we examine the effectiveness of applying classification method using machine learning techniques to the I/O saturation estimation by using Linux kernel I/O statistics instead of the utilization measure that is currently used for HDDs. We conclude that machine learning techniques that we employed (Support Vector Machine and LASSO Generalized Linear Model) performs well compared to the existing utilization measure even we cannot collect the internal information of the flash SSDs.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; • **Hardware** → External storage;

KEYWORDS

Flash SSD, Machine Learning, Load balancing

1 INTRODUCTION

In recent parallel distributed systems that exploit multiple CPU cores for each server to deal with a large volume of data, the performance disparity between processor and hard disk drives (HDDs) has been widened as the number of cores in single CPU increases even more than dozens of cores in enterprise environments [1].

HDDs have been representative storage devices for a long time, but emerging storage technologies have drawn attention to the importance of alleviating the performance gap between processor and storage [2]. In particular, flash SSDs are applicable to I/O intensive environments concerning both IOPS (Input/Output Operations Per Second) and bandwidth. In the enterprise, both HDDs and SSDs are configured to disk arrays, which is called hybrid storage architecture, to improve storage performance. In multiple storage architectures, well-balanced I/O load between storages results in the performance improvement. How I/O loads are distributed among storages can be measured as the utilization [3]. The previous way of measuring utilization used only for HDDs cannot be applied to flash SSDs since HDDs can only perform single I/O request at a time whereas flash SSDs can tolerate multiple I/O requests simultaneously by exploiting the internal parallelism [4]. It 's hard to estimate the utilization of flash SSD because the parallel design involves with garbage collection and merge of blocks stored on flash memories in SSD [5]. Moreover, the previous measure is independent of I/O workload characteristics whereas the other measures will vary as I/O workload characteristics change. The observed values in the same I/O load condition fluctuate slightly because of I/O processing in different kernel layers. This means that we should find the condition that can represent whether it is saturated or not from the other kernel level I/O statistics and measures [6].

In this paper, we introduce the way of performance modeling of the flash SSD using classification method based on machine learning techniques. To learn the model structure, we generated various I/O workloads that can extract almost all meaningful observations by collecting I/O statistics and calculating measures. From these datasets, we defined an additional classifier which can determine whether each observation indicates fully utilized status or not. By using them, we constructed SVM [7] and LASSO GLM model [8] and evaluated these models with 10-folds cross-validation [9]. Our approach shows excellent quality that can be used for basis in load balancing.

The remainder of this paper is organized as follows. Section 2 represents the motivation of this work. We explain how we gather data in Section 3. We describe how we clean data and find the performance limit and model construction technique in Section 4. In Section 5, the performance of estimation model is described and present conclusions.

2 MOTIVATION

We empirically verified the problem of the HDD based measure when it applied to the flash SSDs. We used Samsung 850 PRO SATA-III SSD for all the experiments in this paper. We gradually increase the number of threads each of which performs sequential

†Corresponding author. Tel.: +82 2 2123 5714

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11...$15.00
https://doi.org/10.1145/3132847.3133120

read I/O requests with the fixed 64 KBytes block size for all tests. We confirmed that the utilization reaches 100% when 3 threads run. It looks like the saturation occurs, but the actual bandwidth and IOPS does not reach the maximum. In fact, the flash SSD is saturated when the 5 threads run. From that point, the running time to complete the job just goes up as we increase more the number of threads because I/O load hit the performance limit of the flash SSD. However, the current utilization measurement cannot reflect these phenomena.

The reason why the existing utilization measure is incorrect on flash SSD is that it is defined as the rate of CPU time used for I/O processing (referred to as I/O ticks) for a given period of CPU time. Even if the utilization measured in the previous way goes up to 100%, a flash SSD could perform more requests simultaneously due to internal parallelism, and thus, can utilize more bandwidth and IOPS [10].

3 DATASET

We generated I/O workloads using FIO benchmark [11] by fine-tuning its parameters that can change I/O workload characteristics described in Table 1.

Table 1: I/O characteristics

Name	Description
I/O operation type	The operation type of I/O requests, e.g., read-only, write-only, read/write mixed
I/O request size	The size of I/O requests, it is typically faster and more efficient to read/write a few large requests than many small ones
I/O Access Pattern	Whether the I/O access locations on storage are randomly or sequentially arranged, it is typically faster sequential access than random access pattern
Outstanding I/O Level	The number of I/O requests issued simultaneously
Read/Write Ratio	The ratio between read and write I/O requests if the mixed read/write type

1. Linux OS's default block size is 4K and the maximum request size is 512K. Thus, we gradually increased the block size in powers of two KBytes increments from 4K to 512K.
2. We configured I/O load to show both idle and the utilization status.
3. All the existing types and patterns were considered in the data set generation process.
4. In the case of mixed I/O requests, we attempted to show the common case when read/write ratio reaches to 50:50.

We collect datasets comprising the kernel I/O statistics and measures every second described in Table 2. These are commonly used for monitoring storage performance. To mitigate bias, we removed a ramp-up section to reach the intended I/O load while collecting data. The data in Table 2 is used as inputs in experiments.

Table 2: Kernel I/O statistics and measures

Name	Description
Read Ratio	Read I/Os / Total I/Os
Bandwidth	The number of Read or Write I/Os / Elapsed I/O time
Await time	CPU time / Total number of Read or Write I/Os
Number of I/Os	Number of I/O requests per unit time
Merged I/Os	Number of merged I/O requests per unit time
Average request size	Sum of the size of read and write I/Os / unit time

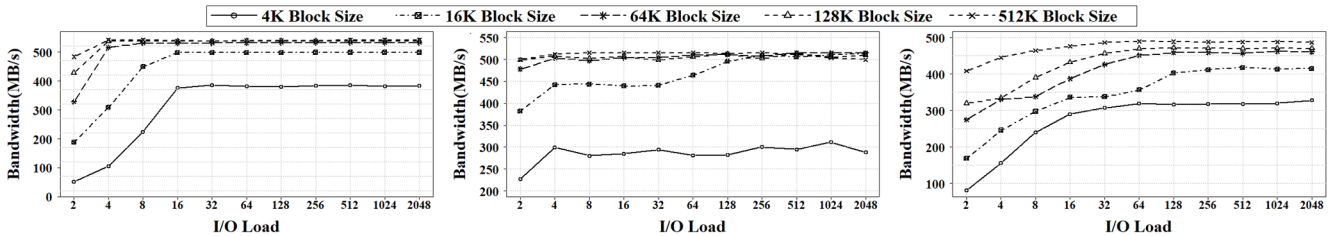
4 METHOD AND ALGORITHM

All stages including preprocessing, classification and evaluation, were performed for each I/O type separately. Each type has slightly different preprocessing step before using classification methods because performance limit changes depending on I/O types. Each preprocessing step involves defining variables and finding critical I/O load which indicates saturation. Details of preprocessing are described in the next section, 4.1. Then we could apply two machine learning techniques, LASSO GLM and SVM which are outlined in section 4.2.

4.1 Preprocessing

Since our experiments had various kernel statistics observations, we summarized the statistics to extract significant variables. First, to calculate accurate IOPS, we defined total IOPS as the summation of I/Os for read and write in mixed type. We also achieved total bandwidth by adding bandwidths for read and write.

Secondly, we carried out an investigation for each dataset based on the difference in performance as I/O load changes. When we looked at the observed bandwidth in various conditions, we found that the bandwidth has limited upper bound while block size



(a) Random Read (b) Random Write (c) Random Read/Write 50:50 Mixed

Figure 1: Performance for each block size as I/O load increases

changes. Simultaneously, we confirmed that average request queue size and await time increases constantly. Fig. 1 highlights this performance wall in different block sizes. The walls vary in upper bound depending on block size, I/O type.

Fig. 1 (a) represents that, in case of a random read I/O type, the critical I/O load which starts to be stuck in the bandwidth wall becomes smaller when the block size is relatively larger. In addition to that, the upper bounded bandwidth of wall is bigger at larger block size than one at smaller block size. However, we figured that the maximum bandwidth walls converge when both block size and I/O load increases. For example, after approximate 20 I/O loads, bandwidth walls of block size 128K and 512K converges. In case of a random write, we observed the enormous jump in performance as Fig. 1 (b) shows. After the advance in performance, the bandwidth starts to maintain the status. However, as Fig. 1 (c) illustrates, the bandwidth walls have various values of upper bounded bandwidth and I/O load in mixed type. Also, the variety of performance becomes larger than in read type.

We considered the I/O load which indicates the beginning of bandwidth wall as a critical load for saturation point. In other words, saturation appears after the critical points. To capture the critical value for each block size and I/O type, we used slightly different algorithms between data types. In the read and mixed read/write datasets where a performance gap does not exist, we used the slopes between each point to find an upper bound. We could capture the critical points when a gradient keeps under the specified threshold or zero. On the other hand, we observed two smooth paces in write type data set. To obtain second plane slope, we used the pre-described gap in the performance. We observed the presence after the huge advance in bandwidth. As we found the critical load at the corresponding block size, we marked labels at and after the I/O load as saturation.

4.2 Classification

4.2.1 LASSO. LASSO (Least absolute shrinkage and selection operator) generalized linear model is one of regression analysis methods that enable classification. LASSO is similar to standard logistic regression. The standard approach to regression modeling is to find the coefficients of regression using least squares or maximum likelihood methods. However, LASSO has a regulation term in the maximizing likelihood computation. The regulation of coefficients results in shrinkage of the impact of given variables which are not significant. Furthermore, the absolute regulative term allows shrinkage variables to zero, which results in the removal of insignificant variables or dependent variables to each other.

For given output vector y and covariate matrix X across all i, LASSO estimator can be computed through maximizing likelihood function $l(\beta)$:

$$l(\beta) = \sum_{i=1}^{N} \{y_i \ln p(X\beta) + (1 - y_i) \ln(1 - p(X\beta))\} \quad \text{subject to } \|\beta\|_1 \le t \tag{1}$$

where $p(x)$ is a logit function and β is coefficient vector.

The above equation can be expressed in so-called Lagrangian form as

$$\sum_{i=1}^{N} \{y_i \ln p(X\beta) + (1 - y_i) \ln(1 - p(X\beta))\} - \lambda \|\beta\|_1. \tag{2}$$

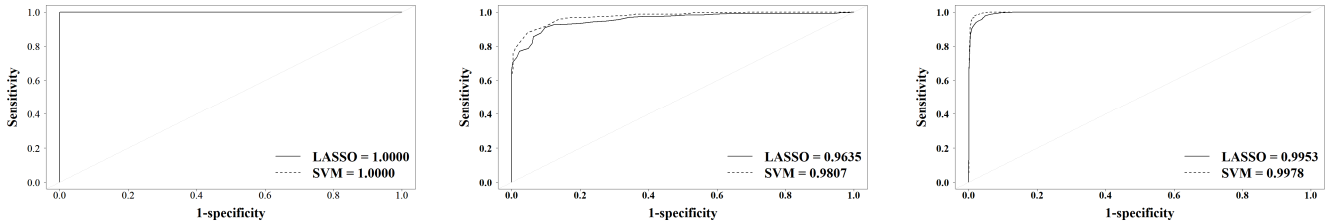
The parameter λ is called a shrinkage parameter and differs from given datasets. Therefore, we used cross-validation for training the parameter to obtain λ that minimizes the average cross validation error.

The major advantage of LASSO is that it can subset variables to drop unnecessary features or dependent variables to one another. For example, in read data set, the trained model does not require await variable. Hence, the correlation between the average queue size and await time is 0.8814. In other words, we do not need the await time since average queue size contains enough information about two features. Read/write LASSO model drops request size for the same reason. To subset significant variables leads to enhance model accuracy and interpretation.

4.2.2 SVM. Support vector machine is one of state-of-the-art machine learning techniques. An SVM model finds a hyperplane that enables to separate given data points into two regions, maximizing the distance each other. Also, we used a Gaussian kernel function to allows higher dimension computation in nonlinear manner. For given data points x_i, SVM find a hyperplane as follow:

$$\min_{w,\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{subject to } \xi_i \ge 0,\ \forall i \tag{3}$$
$$(\mathrm{w} \cdot \Phi(x_i)) \ge 1 - \xi_i, \quad \text{if } y_i = 1$$
$$-(\mathrm{w} \cdot \Phi(x_i)) \ge 1 - \xi_i, \quad \text{if } y_i = 0$$

where w is a normal vector for hyperplane, n is the number of data points, ξ_i is a slack variable, and C is a unit cost. In Gaussian kernel, γ is needed to avoid overfitting within using



(a) Random Read (b) Random Write (c) Random Read/Write 50:50 Mixed

Figure 2: ROC curves and AUC values of LASSO and SVM

distance between vectors. We chose 10 and 0.1 for unit cost and gamma for three models by grid search with cross validation.

5 EVALUATION

5.1 Experimental Results

We evaluated LASSO GLM and SVM models using 10-folds cross validation for each type of data set separately. We also plotted the receiver operating characteristic (ROC) curves to assess the effectiveness of models. Fig. 2 represents the ROC curves for each type of data, and the numbers beside the models depict the average area under the curve (AUC) scores.

Fig. 2 (a) highlights the accuracy of models in read type data. Even if we built cross validation to make a solid inference, both two models would provide a perfect prediction for independent test sets. The possible explanation for the results is that in the case of read only I/O, no factor can bias performance concerning OS or device itself. The fairness of the read only mode allows building powerful models which classify the saturation using kernel statistics.

On the other hand, garbage collection process consumes I/O processing time when the write I/O occurs, so the additional time makes the input values fluctuate. Thus, Fig. 2 reports 96.35% and 98.07% for AUC scores of LASSO and SVM were achieved in write data type. In Fig. 2 (c), AUC scores for LASSO and SVM are 99.5% and 99.8% in read/write datasets, which is higher than write mode. This implies that the less the write ratio, the more stable the kernel statistics are. As a result, the models from mixed mode made more accurate predictions than those from write mode.

For LASSO GLM, we expected to shrinkage variables so that we can find features which have more influence than others. We used the IOPS, bandwidth, request queue size, average queue size and await time for the input. The coefficient result of LASSO models used all features to create write and mixed model, but not the await time for the read mode. In read type data set, we observed the correlation between the await time and request queue size is 0.7489, which represents that request queue size includes most information about the await time.

To evaluate two different models, LASSO and SVM, we conclude that the performance of SVM is slightly better than LASSO based on AUC scores. However, SVM took longer time than LASSO since the model contains heavy computation to acquire optimum hyperparameters and support vectors. As both performance results from two models show acceptable AUC scores comparing to HDD based measure, we recommend LASSO model to evaluate saturation in the real work load.

5.2 Comparison with HDD based Measure

Table 3: Biased measured values of the HDD based measure

I/O Type	Over-estimation	Under-estimation
Read	3.4%	21.5%
Write	11.8%	38.5%
Mixed	31.8%	37.4%

We also evaluated accuracy compared to the previous HDD based measure. In our experimental results, the accuracy of the HDD based measure is not good while our method shows impressive accuracy. Table 3 shows an error rate of the HDD based measure.

6 CONCLUSIONS

This paper focuses on the determination whether the flash SSD is in saturation or not by using classification based on machine learning techniques. We use the kernel I/O statistics and measures except for the previous HDD based measure that is inappropriate to flash SSDs. We used two representative machine learning methods: SVM and LASSO GLM to build the model. We evaluated them by conducting 10-folds cross-validation from datasets that we generated without any skewness in distribution among measures to show all cases if possible. Our approach performs quite well compared to the existing HDD based utilization measure.

Future work will mainly cover the comprehensive analysis of the effectiveness of the architecture of the Flash SSD, e.g., the size of the overprovisioning area. Additionally, we will apply our approach other Flash SSDs to show the generality.

ACKNOWLEDGMENTS

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the SW Starlab support program(IITP-2017-0-00477) supervised by the IITP(Institute for Information & communications Technology Promotion).

REFERENCES

[1] CHEN, CL Philip; ZHANG, Chun-Yang. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 2014, 275: 314-347.
[2] Katz, R. H., et al. Disk system architectures for high performance computing. University of California, Berkeley, Computer Science Division, 1989.
[3] Baccelli, F. and Foss, S. On the saturation rule for the stability of queues. Journal of Applied Probability(1995), 494-507.
[4] Agrawal, N., et al. Design Tradeoffs for SSD Performance. In USENIX Annual Technical Conference (USENIX '08) (San Diego, CA, June 14–19, 2008). 57-70.
[5] HAAS, X. Y. H. R.; HU, X. The fundamental limit of flash random write performance: Understanding, analysis and performance modelling. IBM Research Report, 2010/3/31, Tech. Rep, 2010.
[6] Iostat. http://man7.org/linux/man-pages/man1/iostat.1.html
[7] V. Vapnik. Statistical Learning Theory. Wiley, NY, 1998.
[8] Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B. 1996;58:267–288.
[9] KOHAVI, Ron, et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai. 1995. p. 1137-1145.
[10] F. Chen, R. Lee, and X. Zhang. Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing. In HPCA, pages 266-277, 2011.
[11] fio. https://github.com/axboe/fio