2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Neural Network for Saturation Prediction of Solid State Drives

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*— State-of-the-art storage devices that have parallel capability have significantly reduced the performance gap between processor and storage I/O. However, the internal parallelism makes it difficult to measure utilization that can be used as a basis of load balancing, which is a critical feature of performance improvement of parallel systems. When utilization of storage reaches to one hundred percent, the I/O saturation occurs, and then some of I/O loads need to be redirected to the other storage to improve the whole storage performance. There is, to my best knowledge, no studies in I/O saturation prediction of the flash SSDs regarding the internal parallelism that the previous HDD based measure cannot reflect. In this paper, we propose I/O saturation prediction method based on ANN (Artificial Neural Network) using kernel I/O statistics and various performance measures. We extracted I/O statistics and performance measures by conducting I/O workload simulation especially on NVMe (Non-Volatile Memory Express) flash SSD so that we can get as many observations from various I/O characteristics as flash SSD possibly can show. We constructed ANN model and compared with SVM (Support Vector Machine) model. The evaluation shows that ANN performs well compared to the existing utilization measure which assumes that HDD can only perform single instruction at a time.**

Keywords—flash SSD; artificial neural network

I. INTRODUCTION

As the number of cores in single CPU increases even more than dozens of cores in enterprise environments, the performance gap between processor and storage has been widened. Parallel systems, such as big data platforms fully exploit multiple CPU cores by dividing a job into multiple tasks running on each core and execute them simultaneously [1-3]. Generally, data sets are too big to fit into memory in enterprise environments. Thus, I/O processing that each of many tasks reads data from storage easily leads to be a bottleneck in parallel systems.

Emerging storage technologies provide the new possibility to mitigate this performance gap dramatically [4]. In particular, flash memory-based solid state drives (SSDs) are much impressive when it comes to latency which is faster compared to HDDs. In the enterprise, multiple disks including hard disk drives (HDDs) and SSDs are occasionally configured to disk arrays in a single server to improve storage performance. At this point, load balancing that distributes I/O requests among disks is the most significant feature to extract the best performance of the overall disk arrays used in parallel systems [5]. Load balancer finds the saturated disk and then distribute I/O requests from saturated disks to the idle disk based on the utilization which is a general measure of busyness [6]. The I/O saturation means the utilization of storage reaches one hundred percent and cannot process I/O requests more [7]. The reason why we predict the I/O saturation of the flash SSDs is that the existing measure of the utilization is appropriate only for devices serving requests serially such as HDDs. However, for devices serving requests in parallel, such as flash SSDs, this measure does not consider their internal parallelism [8]. The flash SSDs can tolerate multiple I/O requests simultaneously exploiting the internal parallelism whereas HDD can only perform single I/O request at a time. Consequently, the existing way of measuring the utilization cannot correctly detect the I/O saturation of the flash SSDs. Even vendors who produce the flash SSDs provide the theoretical performance limit through experiment under the specific I/O characteristics.

In this paper, we introduce the way of predicting the I/O saturation of the NVMe flash SSD based on ANN using the

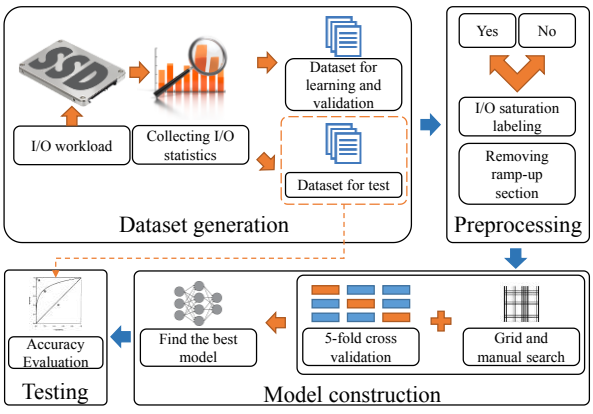


Fig. 1. The overall process of I/O saturation prediction using ANN

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

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).

kernel level I/O statistics and various performance measures [9]. The overall process of our study is represented in figure 1. ANNs have been studied extensively as a useful tool in classification problems. Implementing ANN model, we faced with difficulties. The I/O saturation cannot be defined as a specific formula. Generally, the I/O saturation could be determined manually based on the declining tendency of the bandwidth growth rate as the I/O load increases. Thus, no existing target value can represent whether the I/O saturation occurs or not. To overcome these problems, we define the I/O saturation as a range value which is used for target value for learning and validation. The target value that we defined is based on the change in bandwidth when the I/O load increases, which determines how meaningful some changes are. For example, even though the I/O load is increased higher than the other, if the amount of increased bandwidth is equal to or lower than the other, then it is the most meaningless state which means it is saturated.

To construct model, we generate various I/O workloads on Intel DC 3700 NVMe SSD by doing fine-tune the options of the fio I/O load generator that spawn all possible I/O workloads without any skewness in distribution for any features and then gather data sets of performance measures by monitoring I/O statistics [10]. NVMe SSDs exploit PCIe (Peripheral Component Interconnect Express) interface to deliver more bandwidth performance as compared to SATA-based flash SSDs [11]. NVMe SSDs also have more parallel capabilities that result in the higher error rates when the HDD based measure is applied to it.

The remainder of this paper is organized as follows. Section 2 figures out why the existing measure of the utilization is invalid for the flash SSDs based on experimental results and theoretical basis in terms with the Linux kernel block layer. Additionally, we provide an overview of the internal architecture of the flash SSDs to determine the principles of the parallelism. We explain how we gather and clean data in Section 3. And then, we describe how we define I/O saturation as a formula using analytic results from data and explain the model construction technique in Section 4. In Section 5, the performance of prediction model is described and present conclusions.

II. Background and Motivation

A. I/O Workload

In this paper, storage refers to a technology consisting of a storage device and media it uses. For instance, a hard disk drive (HDD) is the most widely used type of storage in PCs and enterprise environments that stores data on magnetic media. When the operation is transferred between storage and the host, this is called I/O (also known as I/O request), which is an abbreviation for Input/Output. Inputs are the operations and data received by storage and outputs are the operations and data sent from it. I/O operations generated by applications such as a database, big data platforms over a particular period are termed as I/O workload. I/O workload has various characteristics associated with the performance that can be achieved from the storage. These features are described in Table 1.

TABLE I. The meaning of I/O workload characteristics

Characteristics	Description
I/O type	The operation type that each I/O request has (Read or Write)
I/O size	The amount of the data that each I/O request has in bytes
I/O Access Pattern	Whether the I/O access locations on storage are randomly or sequentially distributed
The number of threads	The number of threads that generate I/Os issued to the same storage simultaneously
Read/Write Ratio	The ratio between read and write I/Os in a workload

The performance will vary depending on the state of each I/O workload characteristics. For instance, the write I/O operation is more expensive than the read I/O operation in the case of HDD. The larger the I/O size, the more the bandwidth when the same number of I/Os are issued to storage. Sequential access on HDD shows better performance because it can minimize the seek time. A certain number of the thread may lead to performance gain due to a queue structure in HDD that can optimally merge I/O operations. Read/write mixed I/O operations are slower than read only or write only operations as the mixed I/O will be more likely not to be merged.

B. I/O Saturation Detection Problem

How do we measure the storage performance? It can be measured in two different ways. The first measure is throughput which means the transferred bytes per second usually expressed in Megabytes / Second (MB/s). The second measure is IOPS (I/O Operations Per Second), which means the amount of the read or write operations that could be done in one second. They are closely related to latency which refers to how long it takes for a single I/O request to be performed.

Most storage has performance limits on both throughput and IOPS mainly due to the physical architecture of storage device. Therefore, I/O saturation detection is one of the most important factors for operating and maintaining enterprise parallel systems. Moreover, detecting disk I/O saturation status results in the accurate load balancing that tries to distribute I/O load from saturated disks to the other idle disks to maximize the overall storage performance.

Linux kernel provides many statistical metrics to display the status of storage whether disk saturation occurs or not. These figures, listed in Table 2, are used to calculate measures of disk performance listed in Table 3.

TABLE II. Kernel I/O Statistics

Name	Unit	Description
I/O ticks	Milliseconds	Total time that the storage device has been active
Read/Write ticks	Milliseconds	Total wait time for read/write requests
Read/Write I/Os	The number of requests	Number of completed read/write I/Os
Read/Write sectors	The number of sectors	Number of read/write sectors
Merged read/write sectors	The number of sectors	Number of merged read/write sectors

TABLE III. MEASURE OF DISK PERFORMANCE

Name	Unit	Description
Utilization	Percentage	I/O ticks / Total CPU ticks
Read or Write bandwidth	Bytes per second	Read or Write sectors / Elapsed I/O ticks
Read or Write response time	Milliseconds	CPU ticks / Read or Write I/Os
Read or Write I/Os	The number of I/Os	Number of I/O requests per unit time
Read or Write merged I/Os	The number of I/Os	Number of merged I/O requests per unit time
Average Request Size	The number of sectors	Sum of the size of read and write I/Os / unit time

As described in Table 3, utilization 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. However, this definition is only applicable to an HDD as described before in introduction. This means that even if the utilization goes up to 100%, a flash SSD can perform more requests simultaneously, and thus, utilize more bandwidth. When it comes to NVMe SSD, the gap between value measured as the definition of the utilization and the actual performance limit is more likely to become wider.

We examined the correctness of the HDD based utilization measure through benchmark tests on NVMe SSD. In this test, we configured the block size as 4kbytes and exponentially increased the number of outstanding I/Os. All of the cases were conducted in direct I/O mode to solely evaluate device itself without any effects of OS features such as page buffer. Figure 2 shows that the utilization reaches to 100% before the actual bandwidth reaches to limit. Based on figure 2, the actual maximum bandwidth is more twice higher than where the I/O saturation occurs. It means that although we can exploit more bandwidth, we could not fully utilize storage performance. In the I/O intensive environments, optimizing I/O load balancing directly leads to the performance of the whole system.

C. *Internal Parallelism of the Flash SSDs*

As presented in figure 3, the flash SSDs are comprised of the plural number of flash memory packages. Each package is connected to one of the multiple channels that can deliver data from flash memory to SSD controller and vice versa. SSD controller has a processor dedicated to control the data transfer

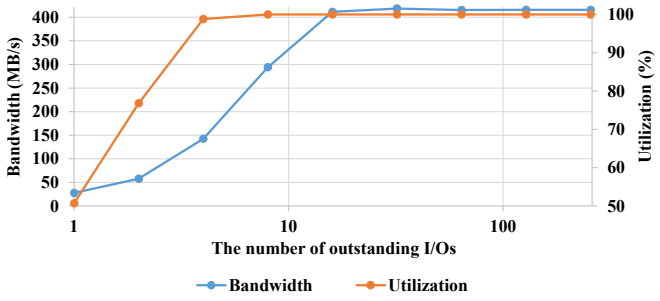


Fig. 2. NVMe SSD read utilization and the actual bandwidth as the number of outstanding I/Os increases

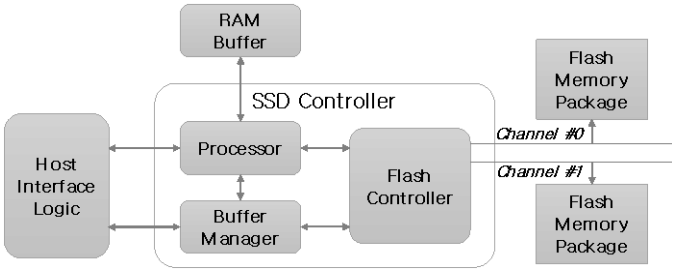


Fig. 3. Flash SSD Internal Architecture

for multiple channels. Internal parallelism is implemented by transferring data to multiple channels simultaneously and controlling these operations using its processor embedded in the flash SSD. This is called channel-level parallelism. On the other hand, there is another parallelism between grouped flash memory packages in a chunk which is called package-level parallelism.

The host interface reconstitutes preferable I/O orders to the channel-level parallelism by swapping the queued I/O requests to make the I/O requests designated to flash memory pages spanning multi-channels. From the knowledge of the channel-level parallelism, we can conclude that the capability dealing with multiple I/O requests make difficult to measure busyness.

III. DATASET GENERATION

Because the utilization cannot be used for the flash SSDs due to the internal parallelism and the flash SSDs themselves do not provide any statistics related to the performance, the other kernel I/O statistics and measures of disk performance except for the I/O ticks and the utilization are only options left listed in both Table 1 and 3. We carefully chose meaningful I/O statistics and measures as indicators that enough to distinguish I/O workload characteristics. These are used as input vectors of the neural network model.

To predict the I/O saturation of the flash SSDs, we generated I/O workloads on NVMe SSD using fio benchmark by fine-tuning its parameters that can reflect the various I/O workload characteristics. All our experiments are conducted using libaio that is Linux native asynchronous I/O library to deliver a heavy load on NVMe SSD without relying on the number of CPU cores [12]. It makes us to correctly measure the performance of the flash SSD with no effects of other hardware devices. Through these figures, we could extract

TABLE IV. FIO BENCHMARK OPTIONS

Name	Settings		Description
Block Size	2^n kbytes, n is an integer given by $2 \le n \le 9$		Block size for I/O units in kbytes
I/O Load	2^n, n is an integer given by $1 \le n \le 11$		Number of I/O units to keep in flight
The type of I/O pattern	***Pattern***	***Type***	I/O access pattern and operation types
	Random	Read Write ReadWrite	
Read/Write Ratio	n : 100 - n, n is an integer given by $10 \le n \le 90$		Percentage of a mixed workload

reasonable observations that the NVMe SSD can show as I/O workload changes.

In Table 4, parameters used in fio benchmark to generate data set are listed. Usually, the default block size of the filesystem is 4k and the maximum request size for the block device is 512k. Thus, we determined the block size between 4k and 512k in powers of two kilobytes increments. We only considered random I/O access pattern because perfectly sequential I/O access pattern rarely exist in the real workload. In the case of mixed I/Os, we attempted to show all possible tendencies by changing the ratio between read and write changes by increments of 10 percent from 10:90 to 90:10. The degree of I/O load is adjusted by the outstanding I/O level that indicates how many I/Os are requested at the same time.

As we described before, there is fluctuation on the performance measure even though the benchmark generates the I/O load in the same settings. Therefore, it is necessary to select the representative value in those fluctuations for each feature. We execute benchmark more than 10 seconds for each setting and collect the I/O statistics and measures in seconds to calculate the mean value as a representative measure. Note that we removed some specific values from observations because there is a ramp-up time to reach to the intended degree of the I/O load while collecting data.

IV. Model Construction

A. The I/O saturation measure

As we mentioned earlier about the definition of the I/O saturation, no existing formula can represent the status of the I/O saturation that can be used for target value in neural network learning. It is only manually determined by monitoring the declining tendency of the bandwidth growth rate while increasing I/O load. Consequently, the specific formula is necessary to label on samples whether the storage is saturated or not. At this point, we propose the formula for detecting a point that indicates the occurrence of the I/O saturation. This label used for one of the input vectors in the neural network model.

Table 5 explains our notation used in the I/O saturation formula. We can calculate the degree of meaningless as the gap between the bandwidth growth and the increased I/O load level. This formula can be represented in (1).

$$M_i = \frac{L_i}{L_{i-1}} - max\left(\frac{B_i}{B_{i-1}}, 1\right),\ i \geq 2 \qquad (1)$$

In the modeling data set generation step (Section 3), we gradually increased the I/O load from step 1 to step 11. For all its steps where i is equal to or greater than 2, we can distinguish which is the most meaningless section through $max(M_i)$, which means there is no necessity that we increase the I/O load.

By the way, if the bandwidth decreases compared to the previous step i-1, meaninglessness M_i can be easily larger than the correct step. There exists fluctuation on the bandwidth as the outstanding I/O level increases even more than M_{max}. Thus we calibrate the effect of the reversal in the bandwidth by the maximum function in (1). Note that the outstanding I/O level cannot be measured using I/O statistics. Thus, it only used for labeling the I/O saturation status of the dataset.

TABLE V. Notations

Notation	Description
i	the number of I/O load increase steps while fixing the other parameters listed in table 4
S_i	i-th sample vector
L_i	outstanding I/O level of the i-th step
B_i	mean bandwidth of the i-th step
M_i	meaninglessness of the growth of the outstanding I/O level when the i-th step compared to the (i-1)th step

Consequently, all the S_j's are labeled as the saturated status if j is equal to or greater than m which is the lowest step among all the steps that have the same M_i as $max(M_i)$.

Figure 4 shows that how well the meaninglessness reflects the tendency of the I/O performance variation as the outstanding I/O level increases. When the outstanding I/O level is 32, it seems that bandwidth reaches about to peak because bandwidth is not improved at the next outstanding I/O level. Thus, it can be considered as meaningless to increase the outstanding I/O level from 32 to 64 and also more than 64.

B. Design of neural network model

The factors that affect the performance limit of the flash SSDs have complex relationships for each feature. ANNs are flexible and nonparametric modeling technique that can well perform the complex function mapping between I/O workload characteristics and storage performance.

We made three different network model rely on the I/O access pattern as listed in Table 6. We chose observations that are independent each other. These careful observations are used as input vectors for input layer, but it is slightly different for each model. Table 6 shows input vectors that are used in each model. Especially, the number of read/write I/Os that were issued to the flash SSD is used to differentiate the read/write ratio for mixed I/O access pattern model. Note that the number of read/write I/Os that were issued to the device

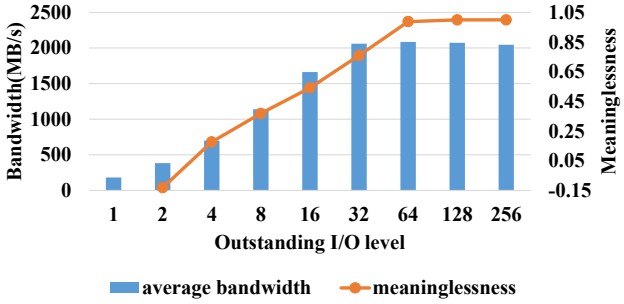


Fig. 4. The variation of meaninglessness and bandwidth as the outstanding I/O level increases for read with 64k block size

TABLE VI. INPUT VECTORS FOR EACH MODEL

Input vectors	Model		
	Read	***Write***	***Mixed***
the number of read I/Os that were issued	√		√
the number of write I/Os that were issued		√	√
the number of read I/Os merged	√		√
the number of write I/Os merged		√	√
The total number of megabytes read	√		√
The total number of megabytes write		√	√
The average queue length of the requests	√	√	√
The average size of the requests	√	√	√
The average time for I/O requests	√	√	√

does not represent outstanding I/O level because it does not count the number of I/Os tried to issue to the flash SSD. Thus, we utilize the other features complexly to predict I/O saturation. The output layer classifies whether the flash SSD is saturated or not.

To optimize hyperparameters for the neural network, we performed the combination of grid and manual search [13]. In our study, we configured a set of configurable variables for the neural network as many as we could sufficiently observe the tendency of quality. Table 7 represents the range of configurable variables that we used for grid search. L1 regularization that can prevent overfitting is used to consider fluctuation on the same I/O workload characteristics. We used ReLU(Rectifier Linear Unit) as an activation function that can mitigate gradient vanishing problem when we applied

TABLE VII. HYPERPARAMETERS FOR GRID SEARCH

Input vectors	Model		
	Read	***Write***	***Mixed***
the number of nodes for input layer	6	6	9
the number of nodes for output layer	2		
the number of hidden layers	1~2		
the number of nodes for hidden layers	1 layer		5 cases {5, 10, 35, 50, 75}
	2 layers		25 cases (cartesian product of two layers)
L1 Regularization	0.0001		
learning rate	0.05		
activation function for hidden layers	ReLU		
activation function for output layer	Softmax		
criteria	LogLoss		
epochs	30,000		
Cross-validation	5 folds		

TABLE VIII. THE EFFECT OF THE NUMBER OF HIDDEN NODES AND LAYERS FOR TRAINING

Model	Hidden nodes	LogLoss	AUC
Read	[50, 75]	0.199881502	0.9590
	[50, 50]	0.200476705	0.9735
	[75]	0.213175548	0.9698
Write	[75, 75]	0.29754443	0.9084
	[50, 35]	0.30833649	0.9151
	[75, 35]	0.318628523	0.9028
Mixed	[10]	0.278938764	0.9721
	[35, 10]	0.282407767	0.9732
	[5, 35]	0.2832191	0.9241

backpropagation technique during learning [14]. The criteria used for quality evaluation of created models are LogLoss.

Table 8 gives the effect of the number of hidden nodes and layers for training step by using the combination of grid and manual search. In the case of read and write only models, when the number of hidden nodes increases, the LogLoss in training sets decreases, whereas the LogLoss of the model for the read/write mixed I/O access pattern is decreased when the number of hidden nodes and layers are lower than the other cases. From this result, we chose the best model that has the lowest LogLoss value and the largest AUC (the Area Under a ROC Curve) value which refers to the accuracy of a model.

C. *Cross-validation*

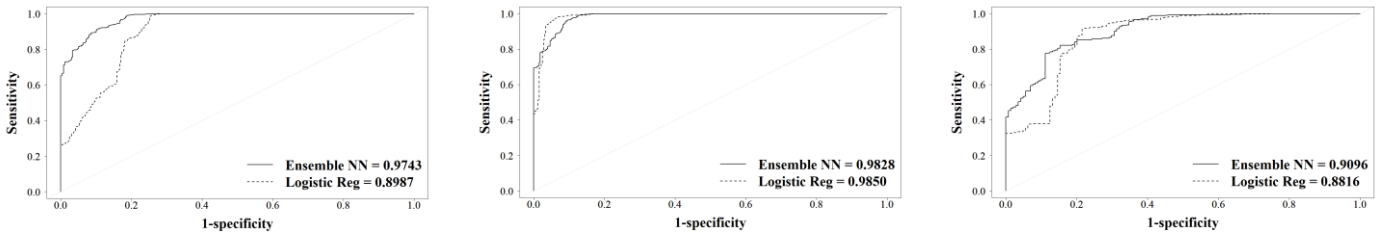
Along with the combination of grid and manual search, 5-folds cross-validation technique is employed to examine the neural network performance in our study regarding sampling variation. In our implementation, the training sample was only used for model construction and/or hyper parameter estimation. The predictive effect of the generated model was evaluated using the test sample. To alleviate to be tailored to fit the training sample, we carried out 5-fold cross-validation. We split the training sample into five equal, and mutually exclusive parts and training was conducted on any four of the five portions. On the remaining part, testing was performed. The results from these steps are described in Table 8.

V. EXPERIMENTS

As described earlier, we generated additional data set only used for testing, which can be used to evaluate whether the derived model is overfitted. To remove any skewness to some

TABLE IX. TEST RESULTS FOR THREE MODEL

Data type	Optimal model	Ensemble model	Logistic model
Read	0.9590	0.9743	0.8987
Write	0.9084	0.9096	0.8816
Read/Write	0.9721	0.9828	0.9850



(a) read (b) write (c) mixed

Fig. 5. RoC curves for the derived models compared with logistic regression

features that can be applied favorably to our model, we chose the block size not used before in testing samples.

The derived model was compared to a traditional classification method in the binomial problem, logistic regression. The logistic regression has 6 or 9 coefficients, corresponding to the number of input variables in each data type. After building the logistic models, we tested them using the independent test set to compare with the optimal neural network model.

An additional experiment was performed to utilize further information using ensemble method with top 3 models. The key element of ensemble method in machine learning is to aggregate weak learners to build a better learner [15]. Since we have five neural networks with different hyper parameters, we average the predictions from top 3 models.

We used receiver operating characteristic (ROC) curve to evaluate the effectiveness of models. Figure 5 shows ROC curves for the data type, and the numbers beside the models demonstrate the area under the curve (AUC). Furthermore, Table 9 highlights the improvement in average models comparing to the optimal models. Unsurprisingly, all three ensemble models outperformed the optimal models based on AUC. The average improvement is slightly less than 1% between optimal and ensemble models.

The ensemble neural network model has considerably better prediction performance than logistic regression in read/write mixed operations. The AUC improved more than 8%, 3% in Read/write mixed model, respectively. However, the baseline logistic model performed better than others in Read/write mixed data. The possible explanation is that the read or write ratio could be calculated and served as linear component, whereas neural network does not consider the ratio, only the absolute number of I/O.

VI. Conclusion

The rapid development of the flash SSDs significantly reduces the performance gap between CPUs and I/O mechanisms by adding parallel capability onto it. However, this capability makes rather difficult to measure how busy it is. This paper focuses on the prediction of the flash SSD whether it is in saturation or not by using classification based on ANN. We used the kernel I/O statistics and measures that can be used to predict the I/O saturation status. We propose the way of measuring meaninglessness from observations, which can determine the status of I/O saturation statistically. By using this, we constructed ANN model and compared with logistic regression. Our experimental result shows better performance than simple logistic regression and also the previous HDD based measure.

References

[1] Chen, CL Philip, and Chun-Yang Zhang. "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data." Information Sciences 275 (2014): 314-347.

[2] Kambatla, Karthik, et al. "Trends in big data analytics." Journal of Parallel and Distributed Computing 74.7 (2014): 2561-2573.

[3] Han, Jing, et al. "Survey on NoSQL database." Pervasive computing and applications (ICPCA), 2011 6th international conference on. IEEE, 2011.

[4] Katz, Randy H., Garth A. Gibson, and David A. Patterson. "Disk system architectures for high performance computing." Proceedings of the IEEE 77.12 (1989): 1842-1858.

[5] Herodotou, Herodotos, et al. "Starfish: A Self-tuning System for Big Data Analytics." Cidr. Vol. 11. No. 2011. 2011.

[6] Iostat. http://man7.org/linux/man-pages/man1/iostat.1.html

[7] Baccelli, François, and Serguei Foss. "On the saturation rule for the stability of queues." Journal of Applied Probability 32.2 (1995): 494-507.

[8] Agrawal, Nitin, et al. "Design Tradeoffs for SSD Performance." USENIX Annual Technical Conference. Vol. 8. 2008.

[9] Proc File System. http://man7.org/linux/man-pages/man5/proc.5.html

[10] fio. https://github.com/axboe/fio

[11] Xu, Qiumin, et al. "Performance analysis of NVMe SSDs and their implication on real world databases." Proceedings of the 8th ACM International Systems and Storage Conference. ACM, 2015.

[12] Kernel Asynchronous I/O (AIO), http://lse.sourceforge.net/io/aio.html

[13] Larochelle, Hugo, et al. "An empirical evaluation of deep architectures on problems with many factors of variation." Proceedings of the 24th international conference on Machine learning. ACM, 2007.

[14] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th international conference on machine learning (ICML-10). 2010.

[15] Rokach, Lior. "Ensemble-based classifiers." Artificial Intelligence Review 33.1 (2010): 1-39.