**Report on Performance Modeling of Flash Storage**

**1 Methodology**

Flash-based solid-state drives (SSD) generally yield better performance than hard disk drives (HDD). In this project, we use a black-box model[1] to analyze and evaluate the performance of SSD, including latency, bandwidth and throughput.

The black-box model predicts the performance as a function of the inputs, i.e. eight workload characteristics. The eight workload characteristics are as follows:

1) Read and write ratio (*wr\_ratio*), which is defined as the percentage of writes in the request;

2) Queue depth (*q\_dep*), which is defined as the number of out-standing I/O tasks;

3) Write size (*wr\_size*), which is defined as the number of bytes write to the SSD;

4) Read size (*rd\_size*), which is defined as the number of bytes read from the SSD;

5) Write randomness (*wr\_rnd*), which is defined as the percentage of random access in the write request stream;

6) Read randomness (*rd\_rnd*), which is defined as the percentage of random access in the read request stream;

7) Write stride (*wr\_stride*), which is defined as the number of bytes between two consecutive writes in stride access pattern;

8) Read stride (*rd\_stride*), which is defined as the number of bytes between two consecutive reads in stride access pattern;

The three performance-metrics are:

1) Latency (*lat*), which is measured in microsecond;

2) Bandwidth (*bw*), which is measured in MB/s;

3) Throughput (*iops*), which is measured in IO/s;

We construct our black box model in two steps:

1) We benchmark the SSD using Intel-ISCSI open storage toolkit [2] and collect 120,000 training data consisting of the above eight workload characteristics and three performance metrics. The data collected should be representative, i.e. should be able to span the entire input space.

2) For every of the three performance-metrics, we train a regression tree to capture the relationship between the inputs and the output. The regression tree is generated by recursively splitting the inputs into leaf nodes using a binary sequence, and the leaf nodes predict the performance metrics as a constant function of the inputs. And the best split minimizes the mean square error (MSE) of all the training data.

**2 Dataset Analysis**

Our experiments are conducted on a workstation with a solid-state drive. The configuration of the workstation is listed below.

Table 1. The configuration specs for the experiment platform

|  |  |
| --- | --- |
| CPU | Intel® Core® i7-6800K @ 3.4GHz, 6 cores |
| Memory | 16GB, DDR4, 3100MHz |
| Disk | CentOS Linux release 7.2.1511 (Core) |
| OS | Samsung 850 Pro SSD, 512 GB |

We use a synthesis workload generator to generate the training data. Here we use the Intel-ISCSI open storage toolkit (OST) to generate all these data. The expected dataset for the training purpose should be able to span the entire feature space, and should lie evenly in the feature space. So we sample the data as follows:

1) wr\_ratio (write ratio): 0%, 25%, 50%, 75% and 100%

2) qdep (queue depth): 1, 4, 16, 64

3) wr\_size (write size): 1KB, 4KB, 16KB, 64KB and 256KB

4) rd\_size (read size): 1KB, 4KB, 16KB, 64KB and 256KB

5) wr\_rnd (write randomness): 0%, 50% and 100%

6) rd\_rnd (read randomness): 0%, 50% and 100%

7) wr\_stride (write stride): 0KB, 64KB, 128KB and 256KB

8) rd\_stride (read stride): 0KB, 64KB, 128KB and 256KB

In the above, we keep the samples with non-zero randomness feature and the samples with non-zero stride access feature mutual exclusive. In the other words, if the *wr\_rnd* and/or *rd\_rnd* dimension of the sample are not zero, its *wr\_stride* and *rd\_stride* fields are both zero. Conversely, if a sample’s *wr\_stride* and/or *rd\_stride*, we keep its *wr\_rnd* and *rd\_rnd* to zero.

In total, we have generated 12,000 samples for training, in which there are 500 sequential access samples, 4000 random access samples and 7500 stride access samples. For each sample, we warm up the device for 5 seconds and test for 10 seconds. The entire dataset is generated in nearly 3 days. All the data are recorded in the data sheet. The following illustrations plot some of the sample data, which show some basic characteristic of the SSD we test.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a) | (b) | (c) |

Fig 1. The read/write performance against the read/write size

(a) latency (b) bandwidth (c) throughput

From the above illustrations, we can learn that:

1) The latency of read and/or write accesses is nearly proportional to the read/write size.

2) The read/write bandwidth increases rapidly at first as the read/write size becomes larger. After the size goes beyond a critical value, the bandwidth grows slowly.

3) The throughput of the devices decreases rapidly as the read/write size grows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| (a) | | (b) | | (c) |

Fig 2. The read/write performance against the write ratio (under different read/write sizes)

(a) latency (b) bandwidth (c) throughput

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| (a) | | (b) | | (c) |

Fig 3. The read/write performance against the queue depth

(a) latency (b) bandwidth (c) throughput

According the above two sets of images, we can see that:

1) The performance of the disk access in the read/write mixing case is worse than that in the pure read / pure write case.

2) The performance of the disk access goes down when the depth of IO queue becomes larger.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| (a) | | (b) | | (c) |

Fig 4. The write performance against the access size, comparing the sequential and the random

(a) latency (b) bandwidth (c) throughput

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| (a) | | (b) | | (c) |

Fig 5. The write performance against the access size, comparing the sequential and the stride

(a) latency (b) bandwidth (c) throughput

From the above, we can see the impact of the access pattern on the performance. In the stride access cases, the performance is similar to one in the sequential case. But in the random cases, the performance is worse, esp. for the latency.

**3 Model Training and Analysis**

We follow the paper and use a regression tree to train the data from the synthesis workload generator. And we adopt *DecisionTreeRegressor* from *scikit learn library* to train and most tree characterics are properly configured after careful calibration, for example MSE is utilized as the tree splitting criterion.

To train the regression model and test the performance of the model. We first generate the training dataset and test dataset from the data we collect about SSD performance. Our strategy is to shuffle the raw data and split the data into a training set of 90% and the rest 10% as the test set. The most important parameter that will influence the performance of the model is the depth of regression tree. We tune the parameter to compare the MRE and R2 in test dataset. The followings are the relationship between MRE or R2 and tree depth.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Fig 6. The accuracy of the models against the tree depth

We carefully examine the relationship of MAE, MRE and R2-score with tree depth and choose the optimal value for depth. As shown in the figure, a tree with small depth will yield unsatisfactory prediction results. On the other hand, over-deep tree might lead to overfitting, such as in the case of the model for latency. Judging from the result, the optimal tree depth is 7 or 8, which is also consistent with the training results from the paper.

Here we choose 7 as the depth for all three regression tree models for the disk performance. And the final results of our model are listed as below.

Table 2. The accuracy for the resulting 7-depth regression tree model for the disk performance

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE | MRE | R2 |
| Latency | 0.3581 | 0.3615 | 0.9615 |
| Bandwidth | 23.10 | 0.1034 | 0.9494 |
| Throughput | 2862.22 | 0.1204 | 0.9601 |

According to the MRE and R2 values, all three models perform well on the validation data. The bandwidth model and the throughput model have relatively low MRE (the lower the better), and the latency model has the highest R2 (the higher [closer to 1] the better). And these results are also consistent with the results from the paper.

Here we also give an approximate illustration of our regression tree models. The following is a 4-depth regression tree model for bandwidth. The regression tree is essentially a binary tree. Each non-leaf node indicates a condition on the feature. If the condition holds, then go to the left child node; otherwise, go to the right child node if the condition fails. The leaf node represents the predicted value for the samples fallen into that branch.

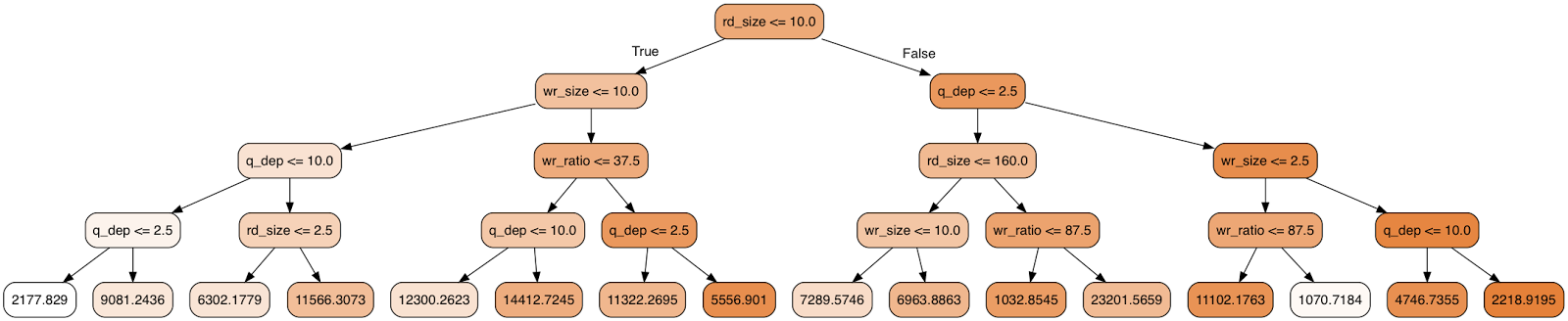


Fig 7. An 4-depth approximation of the regression tree model for I/O access bandwidth

As a 7-depth regression tree model has too much nodes and branches, refer to the spreadsheet for the full illustration of all three models.

**4 Conclusion**

In this project, we try to model the performance of the flash-based solid-state drive (SSD) using regression-tree-based black-box models. We utilize as much as 8 features, like write ratio, queue depth, access size, randomness and stride size, to predict the performance of I/O accesses --- the latency, the bandwidth and the throughput. We use the synthesis workload generator to collect sample data for training and validation. Finally, we train the models and tune the hyper-parameters (such as tree depth and others) for optimal performance. The resulting models yield good predictions on the SSD performance.

*Reference*

[1] Huang, H. Howie, et al. "Performance modeling and analysis of flash-based storage devices." Mass Storage Systems *and Technologies (MSST), 2011 IEEE 27th Symposium on*. IEEE, 2011.

[2] Intel, “Intel-ISCSI open storage toolkit.”