Storage Device Performance Prediction with Selective Bagging Classification and Regression Tree

Lei Zhang, Guiquan Liu, Xuechen Zhang, Song Jiang, Enhong Chen

Storage Device Performance Prediction with Selective Bagging Classification and Regression Tree

Lei Zhang[1], Guiquan Liu[1], Xuechen Zhang[2],
Song Jiang[2] and Enhong Chen[1]

[1] University of Science and Technology of China,
[2] Wayne State University
stone@mail.ustc.edu.cn, gqliu@ustc.edu.cn
{xczhang, sjiang}@eng.wayne.edu, cheneh@ustc.edu.cn

Abstract. Storage device performance prediction is a key element of self-managed storage systems and application planning tasks, such as data assignment and configuration. Based on bagging ensemble, we proposed an algorithm named selective bagging classification and regression tree (SBCART) to model storage device performance. In addition, we consider the caching effect as a feature in workload characterization. Experiments indicate that caching effect added in feature vector can substantially improve prediction accuracy and SBCART is more precise and more stable compared to CART.

Keywords: Performance prediction, Storage device modeling, CART, Ensemble learning, Bagging.

1 Introduction

Today's high-end storage systems are highly complex and configurable, and the automation of storage management is a critical research challenge. One key issue in the automation of storage management is the placement of data sets onto various devices in the storage system, or how to map a workload of specific characteristics onto a appropriate storage device for high service quality and system utilization. To this end, the system must be able to predict the performance a device can provide in its service of a particular workload.

Performance prediction for storage systems have long been studied. Among them are three methods which are particularly useful and efficient. They are analytic device modeling, simulation and emulation, and black-box modeling.

It is a cumbersome task to build accurate analytic models for disk drives because of their nonlinear and state dependent behavior. Ruemmler and Wilkes [1] developed analytic disk models that take into account disk head positioning, platter rotation, and data caching and read-ahead. The model is further improved by Worthington et al. [2], resulting in a widely used disk simulator, DiskSim [3], which represents the state of the art in disk simulation. DiskSim simulates almost

all performance-relevant components of a disk, including device drivers, buses, controllers, adapters, and caches. Emulators go one step further than simulators. In addition to modeling performance, they can interoperate with real systems. For example, MEMS devices can interact with existing system [4]. While disk arrays are widely used in the high-end storage systems, a lot of research work focuses on the modeling and simulation of disk arrays [5–7]. Among the work, the Pantheon storage system simulator [7, 8] was built to support the rapid exploration of design choices of storage systems in HP AutoRAID advanced disk array technology [9] and TicherTAIP parallel RAID architecture [10]. Uysal et al. developed a composite analytic model of mid-range disk array and reported its accuracy within 15% of actual measurements [6].

Compared with simulations and emulations, analytic models are much faster. However, they cannot capture as many details as simulators and emulators. Both methods rely heavily on human expertise on the targeted system and thus are called white-box approach. Given sufficient time and expertise, the white-box approach can work well in exploring design space for a particular device. Unfortunately, such time and expertise is not always available for high-end storage systems because the systems are usually complex and opaque. In addition, some information, such as patented use of algorithms and optimizations, is not disclosed. Furthermore, the technical trend towards storage consolidation in large data centers hints that building an accurate model or simulator using white box method cannot be a general solution in serving a variety of very different workloads. In contrast, the so-called black box approach treats the storage system as a black box without knowing the internal components or algorithms and can accommodate workloads of different characteristics. In this approach, the training data sets, which contain quantified description of characteristics of input I/O requests and their corresponding response times from the system, are recorded in a table [12] and fed into a statistic model [13] , or a machine learning model [14, 15].

Wang et al. [15] proposed to use classification and regression tree (CART) method as a black-box model for performance prediction as it is easy to fit for different workloads, has good interpretability, and provides good approximations to highly nonlinear mappings. However, the CART model has its critical drawback – it is not stable, in the sense that a small change of the data set can lead to a drastic change of the result (more details will be described in section 2.1). This paper addresses the issue by using ensemble algorithms to improve and enhance the accuracy and stability of the basic CART model.

In this paper, we propose the selective bagging CART (SBCART) model, in which we modify the bagging algorithm for regression ensemble. In the model built on top of the CART model, we first train N models by bagging and then select n representative models from N models, where $n < N$. Compared with the CART model, SBCART can provide more precise and more stable performance predictions for modeled storage devices. In addition, an important measure missing in the feature vector designed by Wang [15] is about caching effect, which

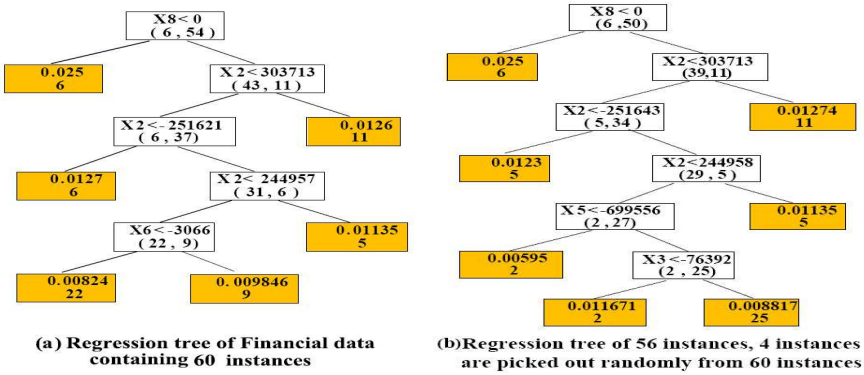

Fig. 1. (a) Regression tree constructed with the Financial I/O trace containing 60 instances; and (b) regression tree of 56 instances, which is produced by removing four randomly selected instances from the tree in (a). The shaded nodes represent leaf nodes containing the predicted values. The N_{min} is set to 25. More details on the Financial trace can be found in Section 3.

makes a substantial difference on prediction accuracy. We include the measure in the vector to make good predictions with the SBCART model.

The remainder of this paper is organized as follows. Section 2 describes the SBCART model. Section 3 presents our experimental results, and Section 4 concludes the paper.

2 The SBCART Models

2.1 The CART Model

CART [16] is a nonparametric model which uses historical data to construct so-called decision trees. Trees are built top-down recursively beginning with a root node. At each step in the recursion, the CART algorithm determines which predictor variable and its value in the training data best split the current node into child nodes. The best split should minimize the difference among the instances in the child nodes. In other words, a good split produces child nodes with instances that contain similar values.

Trees are grown to be excessively large with hundreds of levels if there is no pruning step. Furthermore, a tree of too large size may cause over-fitting, in the sense that it may perform poorly in predicting independent data. There are two pruning algorithms: optimization by minimum number and cross-validation. In the first pruning algorithm, splitting is stopped when the number of instances in the node is fewer than predefined required minimum N_{min}. This approach is efficient, easy to apply and can produce consistent results. However, it requires the calibration of new parameter N_{min}. In the second pruning algorithm, the procedure of cross validation is mainly based on the optimal proportion between

the complexity of the tree and the misprediction error. With the increase of tree size, the misprediction error is decreasing and reaches 0 when the tree grows into maximum tree. Unfortunately, it usually generates complex decision trees that perform poorly on independent data. Therefore, a critical operation is to find the optimal proportion between the tree complexity and misclassification error. Cross-validation does not require adjustment of any parameters. However, it is expensive to apply this pruning algorithm.

Once the tree is built, an instance can travel the pruned tree to make a prediction. At each tree node, either left branch or right branch is taken according to the outcome of comparison of the instance with the split variable and its value of the node. Finally, the instance reaches the leaf node whose value will be the predicted value. However, CART may produce unstable decision trees. Insignificant modification of learning instances, such as elimination of a few instances or changing split variables and values, could lead to radical changes in decision trees. As Figure 1 shows, the decision tree constructed with 60 instances in Figure 1(a) is very different from the tree in Figure 1(b), where only four randomly selected instances are removed.

2.2 Ensemble Learning

The goal of the ensemble learning method is to construct a collection (an ensemble) of individual models to improve the accuracy and performance of a single model. Many researchers have demonstrated significant performance improvements through ensemble methods [17, 18]. Figure 2(a) shows the basic steps involved in the training of a series of models with training data and in the using of voting strategy to predict new data samples.

Two of popular techniques for constructing ensembles are bagging [19] and the adaboost family of algorithms [20]. Both methods invoke a base learning algorithm many times with different training sets. In bagging, a training set is derived by forming a bootstrap replica of the original training set, and each training record has the same weight. Compared with the bagging, the adaboost algorithm maintains a set of weights over the original training set and adjusts these weights after each model is trained with a base learning algorithm. The adjustments increase the weights of examples which are poorly predicted by the base learning algorithm and decrease the weights of examples which are well predicted.

Bagging generates diverse classifiers or models only if the base learning algorithm is unstable, that is, small changes to the training set lead to significant changes of the learned classifier or model. Bagging can be viewed as an approach of improving prediction accuracy by exploiting the instability, because the composite model can efficiently reduce the variation of individual models. Adaboost requires less instability than bagging, because it can make much larger changes in the training set. As CART is instable, we propose to use the ensemble of bagging to improve the prediction accuracy of individual CART models.

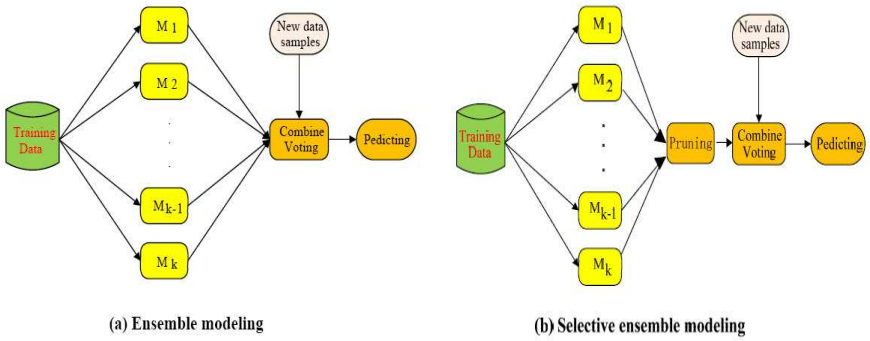

Fig. 2. (a) Using ensemble learning to improve the accuracy of individual basic CART models; and (b) using selective ensemble learning to improve the accuracy of a basic model. In (a), bagging or adaboost can construct a series of models $M_1, M_2, ..., M_k$ and then predict the unknown samples by using voting strategy; In (b), we first construct k models by bagging or adaboost, then select some representative models from k models, and use voting strategy to predict new samples.

2.3 The SBCART Method

Bagging is one of the widely used ensemble learning algorithms. Each training set is constructed by forming a bootstrap replica of the original training set. Thus, some samples in the original training set may appear many times in bootstrap data set while other samples may not appear. Prior research indicates that bagging can substantially improve the effectiveness of the unstable basic learning models [19].

However, as the number of ensemble models increases, the space and time cost will increase linearly. Many methods have been proposed to address the issue by using different classification methods [21–23]. Zhou et al. proposed a method to select a portion of a whole model tree and use genetic algorithms to prune the scale [21]. Bakker et al. proposed to cluster all models and then select representative models in each class to prune whole models [22]. Martinez-munoz et al. proposed a method to prune trees in the ordered bagging ensembles [23]. However, those pruning methods are pretty complicated. In contrast, our SBCART algorithm is proposed to solve the problem on regression and the pruning method is simple. We adopt CART as the basic model. First, we create k models by bagging. Second, we sort the k models by median relative error on the training set. Finally, we select the first 20%-50% of whole models to prune the scale. Figure 2(b) shows the basic steps involved in the training of a series of models on the training data, the selection of a part of whole models, and the using of voting strategy to predict new data samples based on the pruned models. Compared with figure 2(a), figure 2(b) adds a selective (pruning) function to prune the k models. The SBCART algorithm is described as follows using pseudo code.

Input:
D: the dataset containing d samples;
M: CART(the basic model);
k: the number of models;
s: the number of pruned models
Output:
Pruned models M^*;

Training Phase:
(1)for i=1 to k do //Bagging
(2) Sampling with replacement, yield D_i (remove duplicated instances);
(3) Create the model M_i based on D_i;
(4) Compute error(M_i) (error(M_i) = $\frac{1}{d}\sum_{j=1}^{d}\frac{|M_i(X_j)-y_j|}{y_j}$); //median relative error
(5)end for

Pruning Phase:
(1)Order k models by error(M_i) in ascending order;
(2)Get the first 20%-50% of the ordered models

Prediction Phase:
Using the pruned models to predict testing data X
(1)for i=1 to s do
(2) $W_i = log(\frac{(1-error(M_i))}{error(M_i)}+1)$; //assign weight for each model
(3) $V_i = M_i(X)$; //predicted values
(4)end for
(5)Normalize the W_i;
(6)return $\sum_{i=1}^{n} W_i * V_i$

In the algorithm, we remove duplicated instances in D_i while sampling with replacement in training phase, because the duplicated instances in the training set can have a negative effect on choosing the best split variable and lead to significant changes of the structure of the tree. In our experiments, the size of the data set is reduced by half (5000 instances to 2500 instances) and the time of tree construction is reduced from 7 seconds to 1.25 seconds after deleting duplicated instances in D_i. In the algorithm, the weight function in the prediction phase is set to $log(\frac{(1-error(M_i))}{error(M_i)}+1)$, this can guarantee the weight to be positive. Furthermore, in order to get enough representative models, we choose small pruning proportion (20% for example) if the k is large, and choose large proportion (50% for example) if the k is small. Compared with unpruned bagging ensembles of CART model, the SBCART has big advantages in both space cost and computation time because the scale of models is reduced to about 20%-50% of the whole model.

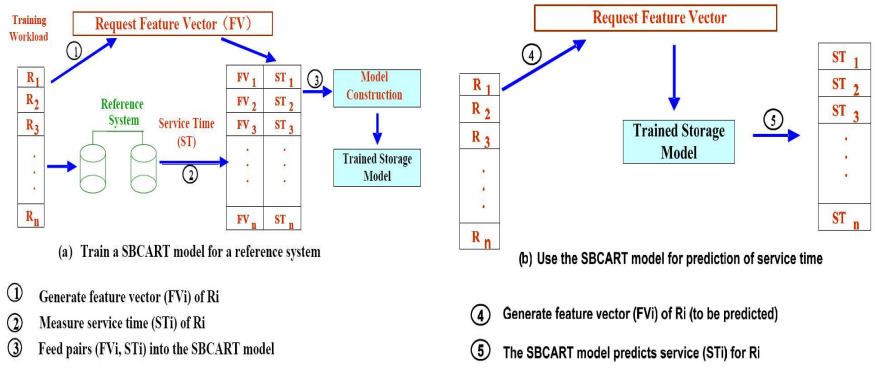

Fig. 3. (a) Training a SBCART model based on observed response times; and (b) using the model to predict response times.

2.4 Predicting Performance with SBCART

Our goal is to build a model for a given storage device to predict device performance as a function of I/O workload. We use the UMass traces [24] which define a workload as a sequence of I/O requests. Each request R_i is characterized with five attributes: application specific unit (ASU), logical block address (LBA), size (SIZE), opcode (OPCODE), and timestamp (TIMESTAMP). The ASU is a positive integer representing the application specific unit; The LBA field is a positive integer that describes the ASU block offset of the requested data; The SIZE field is a positive integer that describes the number of requested bytes, where the size of a block is contained in the description of the trace file; The OPCODE field is a single, case insensitive character that defines the direction of the transfer, R or r indicates a read operation, W or w indicates a write operation; The TIMESTAMP field is a positive real number representing the offset in seconds for this I/O request from the beginning of the trace.

Our approach uses SBCART to approximate the function. We assume that the model construction algorithm can take any workloads on a device for model training. Figure 3 shows the basic steps involved in the training of a model based on the observed response times and using the model to predict system response, which is per-request response time in this study. Model construction does not require any information about the internals of the modeled device. Therefore, the methodology is generally enough to model any device.

We compared our SBCART model with CART model in Table 1. Various aspects are listed, including prediction error, stability, interpretability, robustness to outliers, ability to handle irrelevant input, model construction time, and prediction time. We list these aspects in the order of their importance to the storage performance prediction. Good stability indicates that a small change of the training data set cannot lead to significant change of the prediction results. Interpretability describes a model's ability to infer the importance of input

Table 1. Comparison between SBCART and CART models when they are used to predict per-request response time.

Feature	SBCART	CART
Prediction error(median relative error)	0.1895	0.2319
Stability	Good	Poor
Intepretablility	Good	Good
Robustness to outliers	Good	Good
Ability to handle irrelevant input	Good	Good
Model construction time	25 seconds	7 seconds
Storage requirment	152 KB	32 KB

variables. Robustness describes a model's ability to respond to noisy data sets. Irrelevant input refers to features that have little predictive value. We only compare SBCART with CART, and the comparison of other regression methods can be found in [15]. The two models are constructed using the first 5000 instances of Financial user4 trace and run on another 5000 instances of the same trace for testing (More details on the trace can be found in Section 3). The parameter k in SBCART is set to 20 and N_{min} in CART is set to 10.

As shown in Table 1, the prediction error (median relative error) of SBCART is lower and the stability is better compared to CART, as the composite models can reduce variance of individual models. The construction of SBCART model takes a longer time period and the space overhead of this model is higher, because SBCART needs to build k different models. However, the higher costs are well affordable in the systems for storage device performance prediction. Furthermore, the model construction time for SBCART can be reduced with parallel execution as each bootstrap modeling is independent. Overall, the SBCART method proposed for storage device performance prediction is more stable and more precise than CART.

3 Experiments

3.1 Request Feature Vector

Our request Feature Vector (FV) for R_i contains the following variables: Request Vector $R_i = [TimeDiff_i(1), ..., TimeDiff_i(k),\ LBN_i, LBNDiff_i(1), ..., LBNDiff_i(m), Size_i, RW_i, Seq(i), Hit(i)]$ where $TimeDiff_i(l) = TimeStamp_i$-$TimeStamp_{i-l}$ $(l = 1, 2, ..., k)$, $LBNDiff_i(k)$=LBN_i-LBN_{i-k} $(k = 1, 2, ..., m)$; The first k variables measure the temporal burstiness of the workload when Ri arrives. The next $m + 1$ variables measure the spatial locality in terms of the distance of two continuous requests. $Seq(i)$ indicates whether the request is a sequential access; $Size_i$ and RW_i is related to the data transfer time. $Hit(i)$ indicates whether a request is hit in the cache.

3.2 Devices and Traces

We use DiskSim [3] to simulate a disk (Seagate ST32171W) of 7200RPM. We replay all the traces on the device to obtain the training data set. We use the UMass traces [24] consisting of Financial traces and WebSearch traces. The Financial traces are from OLTP applications at two large financial institutions (relatively more sequential) and the WebSearch traces are from a popular search engine (relatively more random).

There are several fields in the record for a request in UMass trace file. The first field is the ASU, which is related to application. In our experiments, we assume that one user runs one application on the server. Therefore, ASU number can be considered as a user ID. We randomly chose two ASU numbers and filtered out all the requests for each of these ASUs, respectively. Accordingly, we obtained WebSearch-user1 and WebSearch-user2 traces from WebSearch1.spc, Financial-user2 and Financial-user4 traces from Financial1.spc. We built our models based on those traces.

3.3 Evaluation Methods

For evaluation, we use the trained device models to predict response time for a single request. We define the relative prediction error as $\frac{|\hat{Y}-Y|}{Y}$ to show the accuracy of different modeling algorithms. We also show the average, 90th, 80th, and 70th percentile relative errors of response time for different data sets.

Based on the above four users' traces, we trained four models: $Model_{Fin2}$, $Model_{Fin4}$, $Model_{Web1}$, $Model_{Web2}$ respectively. One hundred thousand requests are obtained for each user from the original trace and half of the requests are used for training while half of them are used for testing. In our experiments, k in $TimeDiff_i(k)$ is set to 3 and m in $LBNDiff_i(m)$ is set to 5. The k in SBCART is set to 20, the pruning proportion is set to 50% and the N_{min} of CART is set to 10.

3.4 Experiment Results

Figure 4 compares the median relative errors of the two models (SBCART and CART) in modeling the Seagate ST32171W disk on Financial ($Model_{Fin2}$, $Model_{Fin4}$) and WebSearch ($Model_{web1}, Model_{web2}$) traces, respectively. Overall, the SBCART-based device models provide better prediction accuracy in predicting the average, 90th, 80th and 70th percentile response times than CART. We can make several observations from the experiment results.

First, feature vector must be designed to include all relevant measures. An important measure missing in the feature vector designed by Wang et al. [15] is about caching effect, which makes a substantial difference on prediction accuracy. As hitting in the buffer cache is basically determined by temporal locality of accessed blocks [25], we propose to maintain an approximate LRU stack to efficiently track recency of requested blocks and use it as a measure in the vector. As shown in Figure 4(a), CART-cache can reduce the error from 25.12%, 25.04%,

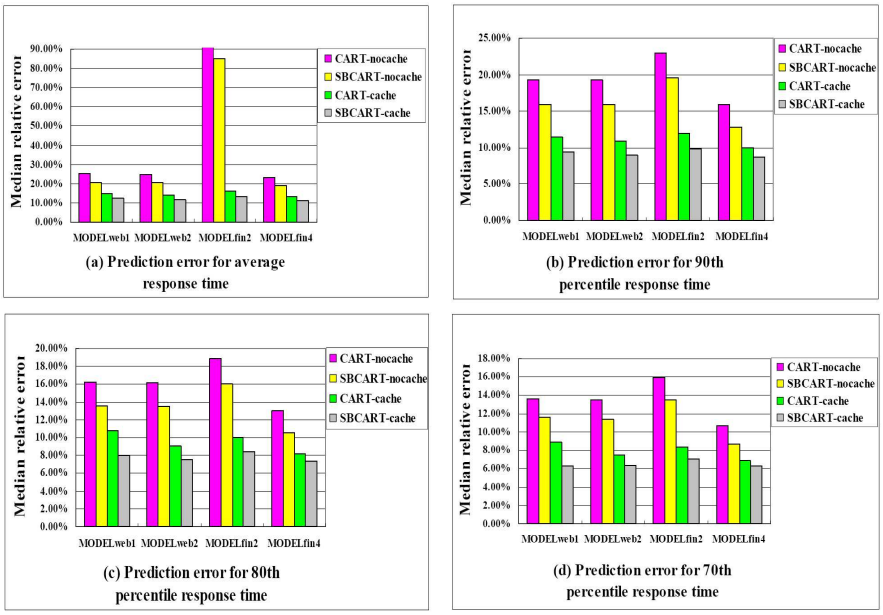

Fig. 4. Comparison of SBCART and CART on four traces: $Model_{Fin2}$, $Model_{Fin4}$, $Model_{Web1}$, $Model_{Web2}$. CART-nocache shows that cache information (Hit) is not considered in feature vector and CART-cache shows that the cache information is used as a measure in feature vector.

91.64%, 23.18% to 15.01%, 14.15%, 15.99%, 13.48% on $Model_{Web1}$, $Model_{Web2}$, $Model_{Fin2}$, $Model_{Fin4}$, respectively. We can see that the median relative error is reduced by about 10% on $Model_{Web1}$, $Model_{Web2}$, $Model_{Fin4}$, and by about 75% on $Model_{Fin2}$. We also observed that the traces of $Model_{Web1}$, $Model_{Web2}$, and $Model_{Fin4}$ are relatively more random and the trace of $Model_{Fin2}$ is relatively more sequential, Therefore, sequential workloads like Financial-user2 are more sensitive to the caching effect, and addition of the cache information can greatly reduce the prediction error.

Second, SBCART can improve the accuracy and stability of CART. As shown in Figure 4(a), the SBCART-nocache can improve prediction accuracy by about 5% compared to CART-nocache, and the SBCART-cache can improve the accuracy by about 3% compared to CART-cache. We can see that by using the measure of Hit in the feature vector and the ensemble method, the prediction accuracy can be increased by about 13% for the relatively random workloads and about 70% for the relatively sequential workloads. As shown in Figure 5, the SBCRAT is more stable than CART when the training data set changes, because selective ensemble models can reduce the variance of individual models.

Finally, it is more difficult to predict response times at high percentiles. As shown in Figures 4(b), (c) and (d), the median relative errors are reduced by

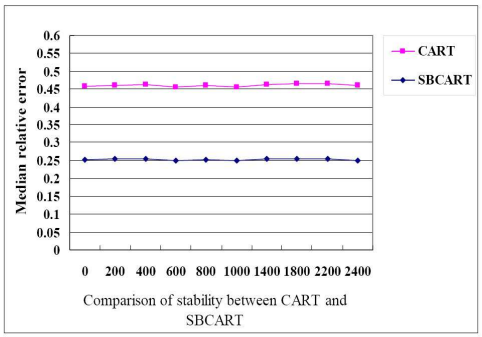

Fig. 5. Comparison of stability between CART and SBCART. X-axis shows the number of training records missing from the first 5000 instances of WebSearch-user1 trace and its testing data from another 5000 instances from the same trace.

about 5%, 8% and 11%, respectively, compared to Figure 4(a). We can observe that SBCART can consistently produce more precise predictions than CART.

In summary, the SBCART model as well as the workload characterization (feature vector) used in the modeling can produce more accurate predictions and is more stable than the CART model.

4 Conclusions

Storage device performance modeling is an important component in self-managed storage systems, especially in high-end storage systems. Our SBCART model takes a workload as input and predicts its performance on the modeled device efficiently and accurately compared to the CART model. Based on bagging algorithms, we proposed a selective bagging classification and regression tree (SBCART) model using the basic model CART model. Our experiment results show that the SBCART model as well as the workload characterization (feature vector) used in the modeling can produce more accurate predictions and is more stable than the CART model.

Acknowledgements. The authors gratefully acknowledge the support of the Fundamental Research Funds for the Central Universities, the National Natural Science Foundation of China (No.60833004 and No. 60775037), the National High Technology Research and Development Program of China (863 Program,No.2009 AA01Z123), and Specialized Research Fund for the Doctoral Program of Higher Education (No.20093402110017). This research was also partially supported by US National Science Foundation under grant CAREER CCF 0845711.

References

1. Ruemmler, C., Wilkes, J.: An introduction to disk drive modeling. IEEE Computer, 27(3),17–18 (1994)
2. Worthington, B., Ganger, G., Patt, Y.: Scheduling algorithms for modern disk drives. In: Proc. of the ACM SIGMETRICS Conference, vol.22, pp.241–251, ACM, New York (1994)
3. The DiskSim Simulation Environment(v3.0), Parallel Data Lab, http://www.pdl.cmu.edu/DiskSim/
4. Griffin, J.L., Schindler, J., Schlosser, S.W., Bucy, J.S., Ganger, G.R.: Timing-accurate storage emulation. In: FAST 2002 on File and Storage Technologies, pp.75–88, USENIX Assoc, Monterey (2002)
5. Barve, R., Shriver, R., Gibbons, P.B., Hillyer, B.K., Matias, B.K., Vitter, J.S.: Modeling and optimizing i/o throughput of multiple disks on a bus. In: ACM SIGMETRICS Conference on Measurement and modeling of computer systems, pp. 83–92, ACM, New York (1999)
6. Uysal, M., Alvarez, M., Merchant, A.: A modular, analytical throughput model for modern disk arrays. In: 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems Conference. pp. 183–92, MASCOTS, Cinncinnati (2001)
7. Wilkes, J.: The Pantheon storage-system simulator. Technical Report HPL-SSP-95-14, Storage Systems Program, Hewlett-Packard Laboratories (1996)
8. Aicheler, U.: A visual user interface for the pantheon storage system simulator. Technical Report HPLSSP961, Storage Systems Program, Hewlett-Packard Laboratories (1996)
9. Wilkes, J., Golding, R., Staelin, C. Sullivan, T.: The HP AutoRAID hierarchical storage system. ACM Transactions on Computer Systems, 14(1), 108–136 (1996)
10. Cao, P., Lim, S.B., Venkataraman, S., Wilkes, J.: The TickerTAIP parallel RAID architecture. ACM Transactions on Computer Systems , 12(3), 236–69 (1994)
11. Schindler, J., Ganger, G.R.: Automated disk drive characterization. CMU SCS Technical Report CMU-CS-99-176 (1999)
12. Andenson, E.: Simple table-based modeling of storage devices. Technical Report HPL-SSP-2001-04, HP Laboratories (2001)
13. Kelly, T., Cohen, I. Goldszmidt, M., Keeton, K.: Inducing models of black-box storage arrays. Technical Report HPL-SSP-2004-108, HP Laboratories (2004)
14. Mesnier, M.P., Wachs, M., Sambasivan, R.R., Zheng, A.X., Ganger, G.R.: Modeling the relative fitness of storage. In: Joint International Conference on Measurement and Modeling of Computer Systems, ACM, New York (2007)
15. Wang, M., Au, K. Ailamaki, A., Brockwell, A., Faloutsos, C., Ganger, G.R.: Storage device performance prediction with cart models. In: 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS, USA (2004)
16. Breiman, L., Friedman, J., Stone, C.J., Olshen. R.A.: Classification and regression trees. Chapman and Hall CRC (1984)
17. Kohavi, R., Kunz, C. Option decision trees with majority votes.: In: 14th International Conference on Machine Learning, San Francisco, CA: Morgan Kaufman (1997)
18. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36(1-2), 105–139 (1999)
19. Breiman, L., Bagging predictors. Machine learning, 24(1), 123–140 (1996)

20. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: 13th International Conference on Machine Learning, Morgan Kaufmann (1996)
21. Zhou, Z.H., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence, 137(1-2), 239–263 (2003)
22. Bakker, B., Heskes, T.: Clustering ensembles of neural networks. Neural Networks , 16(2), 261–269 (2003)
23. Mart nez2mu noz G, Su rez, A.: Pruning in ordered bagging ensembles. In: 23th International Conference on Machine Learning , pp. 1266–73, IEEE, Piscataway (2006)
24. Umass trace repository, http://traces.cs.umass.edu/index.php/Storage/Storage
25. Jiang, S., Zhang, X.: LIRS: an effcient low inter-reference recency set replacement to improve buffer cache performance. In: ACM SIGMETRICS Conference on Measurement and modeling of computer systems, pp. 31–42, ACM, New York (2002)