Storage Device Performance Prediction with SBCART (Selective Bagging Classification and Regression Tree) Models

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

[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

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. It can improve the accuracy of a single CART model. In contrast with CART modeling, the black-box SBCART model predicts the performance with the higher accuracy. Experiments indicate that SBCART used in storage device performance prediction is effective and stable. In addition, we use the caching effect as a measure in feature vector to make good predictions.

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

1 Introduction

Today's high-end storage devices are very complex and highly configurable, making the automation of storage management tasks a critical research challenge. One key problem in automation of management tasks is deciding which data sets to store on which devices, or in other words, how to map a workload of specific characteristics to its appropriate storage device for high service quality and system utilization. In order to do this, it requires the ability to predict how well each device will serve each workload.

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

Building accurate analytic models of disk drives is a burdensome task because of their nonlinear, state dependent behavior. Ruemmler and Wilkes [1] developed analytic disk models that take into account head positioning, platter rotation, data caching and read-ahead. The model is further improved by Worthington et al. [2], resulting in the widely used disk simulator, DiskSim [3], which represents the state of the art in disk simulation. DiskSim models 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 usually can not 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 requests for a particular device. Unfortunately, such time and expertise is not available for the high-end storage system because the high-end storage system is complex and opaque. In addition, some information such as patented configurations, use of algorithms and optimizations is not disclosed, which makes automatic approaches [11] ineffective. Furthermore, the trend toward storage consolidation in large data centers means that building an accurate model or simulator using white box method cannot be a general solution for 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 serve a wide variety of very different workloads. In this approach, the training data sets which contain quantified description of characteristics of the input requests and their corresponding responses 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] deployed the black-box method (CART) in performance modeling as it is easy to fit, leading to good interpretability and can provide good approximations to highly nonlinear mappings. However, CART is a kind of decision tree algorithms and is instable, that is, a small change of the data set can lead to a vital change of the result (The details will be described in section 2.1).

This paper attacks this gap by using ensemble algorithms to improve and enhance the accuracy and stability of the basic model(CART). We have modified bagging algorithm for regression ensemble: First, train N models by bagging and then select n representative models from N models, where $n < N$. In our SBCART method, the CART is referred to as the basic model. Compared with CART modeling, the SBCART used in storage device performance prediction is more precise and more stable. In addition, an important measure missing in the feature vector designed by Wang [15] is about caching effect, which makes a substantial difference on prediction accuracy. We used it as a measure in the vector to make good predictions.

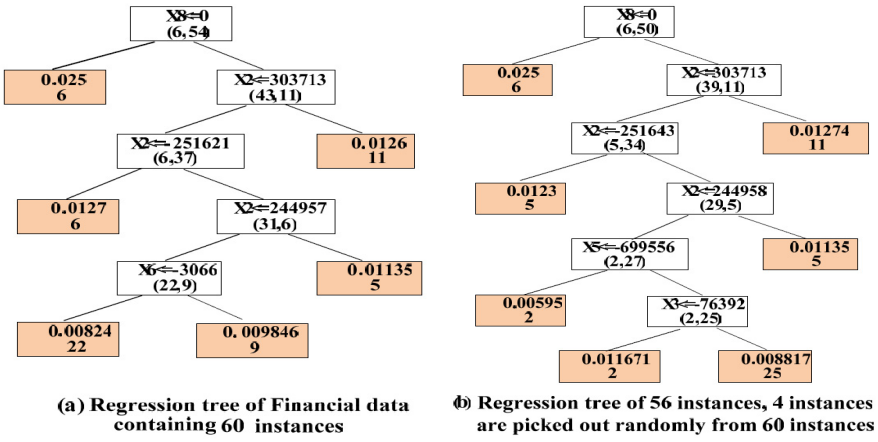

Fig. 1. (a) Regression tree of Financial data containing 60 instances; and (b) regression tree of 56 instances, 4 instances are picked out randomly from 60 instances. The colored node represents leaf node containing the predicted value. The $Nmin$ is set to 25. The information on Financial data may be found in section 3.

The remainder of this paper is organized as follows: section 2 describes SB-CART model, section 3 presents our experimental results and section 4 concludes the paper.

2 The SBCART Models

2.1 CART Methods

CART is a nonparametric algorithm 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 until no further splits are possible. 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 less than predefined required minimum $Nmin$. This approach works very fast, easy to use and has consistent results. However, it requires the calibration of new parameter $Nmin$. 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. As the increase in size of the tree, the misprediction error is decreasing and equal to 0 when the tree grows into maximum tree. Unfortunately, complex decision trees poorly perform on independent

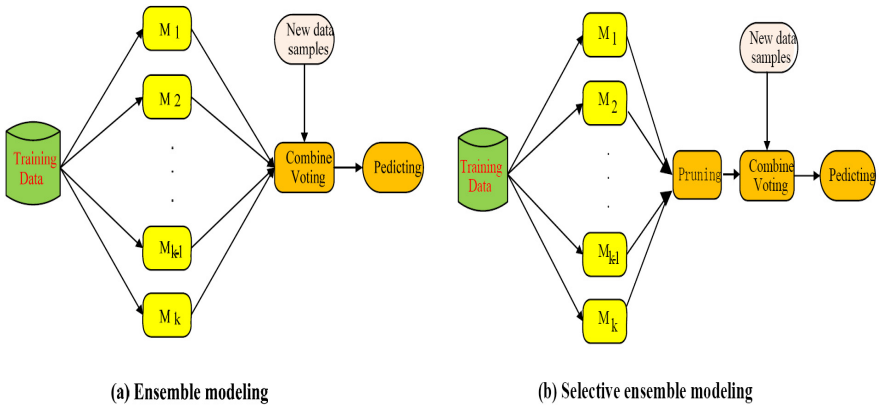

Fig. 2. (a) Using ensemble learning to improve the accuracy of a basic model; and (b) using selective ensemble learning to improve the accuracy of a basic model. In (a), bagging or adaboost can construct a series of classifiers or emulators $M_1, M_2, ..., M_k$ and then predict the unknown sample by using voting strategy; In (b), first constructing k models by bagging or adaboost, then selecting a part of representative models from k models, and finally using voting strategy to predict new samples. In our SBCART model, M_i is set to $CART_i$.

data. Therefore, the primary task is to find the optimal proportion between the tree complexity and misclassification error. Cross-validation does not require adjustment of any parameters, but this process is time consuming.

Finally, as the tree has been built, an instance travels the pruned tree to make a prediction. Compared with the split variable and its value of the visiting node, the left or right branch is taken. Leaf nodes contain the predicted variable and its value will be returned.

CART may have unstable decision trees. Insignificant modification of learning instances, such as eliminate several instances and change split variables and values, could lead to radical changes in decision trees. As Figure 1 shows, the decision tree constructed for 60 instances in Figure 1(a) is very different from the tree in Figure 1(b) where 4 random instances are picked out.

A detailed discussion of CART is available in [16].

2.2 Ensemble Learning

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

Two most 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, each training set is constructed by forming a bootstrap replica of the original training set, and each training record has the same weights. Compared with the bagging, the adaboost algorithm maintains a set of weights over the original training set and adjusts these weights after each emulator is learned by the base learning algorithm. The adjustments increase the weight of examples which are badly predicted by the base learning algorithm and decrease the weight of examples which are well predicted.

Bagging generates diverse classifiers or emulators only if the base learning algorithm is unstable, that is, small changes to the training set lead to large changes in the learned classifier or emulator. Bagging can be viewed as ways of exploiting this instability to improve prediction accuracy. Adaboost requires less instability than bagging, because it can make much larger changes in the training set.

In our SBCART model, we use and adapt the ensemble of bagging to improve the accuracy and performance of a single CART model, because CART is an unstable model.

2.3 The SBCART Method

Bagging is one of the famous ensemble learning algorithms. Each training set is constructed by forming a bootstrap replica of the original training set, thus, some samples in original training set may appear many times in bootstrap data set while other samples may not appear. Research indicated that bagging can boost the performance of the unstable basic method [19].

However, as the number of ensemble models increases, the storage space and computation time will increase linearly. Thus, many methods [21–23] have been proposed to solve this problem on classification. In those methods, Zhou et al. [21] proposed a method to select a part of whole models by using genetic algorithm to prune the scale. Bakker et al. [22] proposed to cluster whole models firstly and then select representative models in each class to prune whole models. Martinez-munoz et al. [23] proposed a new method to prune in ordered bagging ensembles. However, those pruning methods are 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, create k models by bagging; Second, sort the models based on Bagging by median relative error on training set, and finally, select the first 20%-50% of whole models to prune the scale. We can choose the small proportion (20% for example) while the k is large, in contrast, choose the large proportion (50% for example) while the k is small. Compared with unpruned bagging ensembles of CART model, SBCART has big advantages in both storage space and computation time. Figure 2(b) shows the basic steps involved in training a series of models on the training data, selecting a part of whole models, and at last using

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 specific SBCART algorithm is listed below:

Algorithm: SBCART
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) In-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

Predicting 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 $\sum_{i=1}^{n} W_i * V_i$

2.4 Predicting Performance with SBCART

Our goal is to build a model for a given storage device which predicts device performance as a function of I/O workload. We use the Umass traces [24] which define a workload as a sequence of disk requests. Each request R_i is uniquely described by 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 data transferred for this record; The SIZE field is a positive integer that describes the number

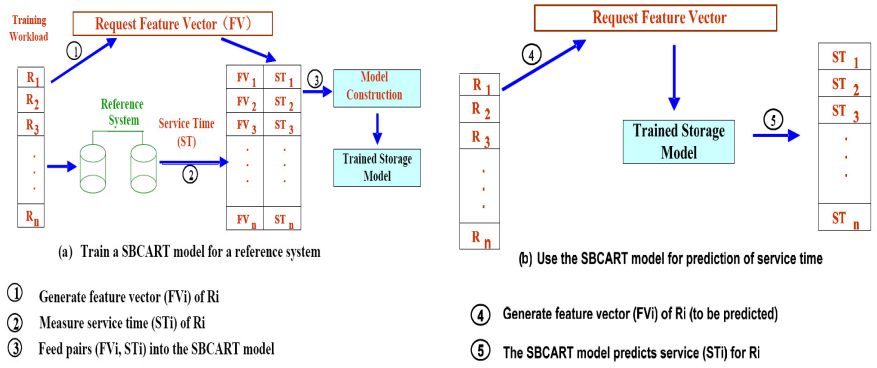

Fig. 3. (a) Training a SBCART model based on observed response times; and (b) using the model to predict service times.

of bytes transferred for this record, 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 start of the trace.

Our approach deploys SBCART to approximate the function. We assume that the model construction algorithm can feed any workloads into a device to observe and profile their characteristics for model training. Figure 3 shows the basic steps involved in training a model based on the observed response times and using the model to predict system response (which is per-request service 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. The two models were constructed on the first 5000 instances of WebSearch user1 trace and run on another 5000 instances of the same trace for testing (The information about the trace we used may be found in section 3). The parameter k in SBCART is set to 20 and $Nmin$ 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 a single model. The construction time of SBCART model is more time-consuming and the storage requirement of this model is higher, because SBCART needs to build k different models, but those are not the most important elements in storage device performance prediction (The most important element is prediction error). Furthermore, the construction time of SBCART can be shortened by using parallel technology as each bootstrap modeling is independent. Overall, the SBCART used in storage device performance prediction is more stable and more precise compared to CART.

Feature	SBCART	CART
Prediction error(median relative error)	0.1205	0.1501
Stability	Good	Poor
Intepretability	Good	Good
Robustness to outliers	Good	Good
Ability to handle irrelevant input	Good	Good
Model construction time	25 seconds	7 seconds
Storage requirement	153 KB	32 KB

Table 1. Comparison between SBCART and CART tools in predicting per-request response time. The comparisons are on prediction error, stability, interpretability, robustness to outliers, ability to handle irrelevant input, model construction time and prediction time. We rank the features in the order of their importance.

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 which supports prediction of the seek time of the request. $Seq(i)$ indicates whether the request is a sequential access; $Size_i$ and RW_i support prediction of the data transfer time. $Hit(i)$ indicates the caching effect, which makes a great difference on prediction accuracy.

3.2 Devices and Traces

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

There are several fields for a single request in UMass trace file. The first field is the ASU which is related to application. In our experiments, we consider one user executing one application on the server. Therefore, ASU number could be considered as a user ID. We randomly chose two ASU numbers and filtered out all the requests for each of these ASUs, respectively. According to this, we obtained WebSearch-user1 and WebSearch-user2 traces from WebSearch1.spc,

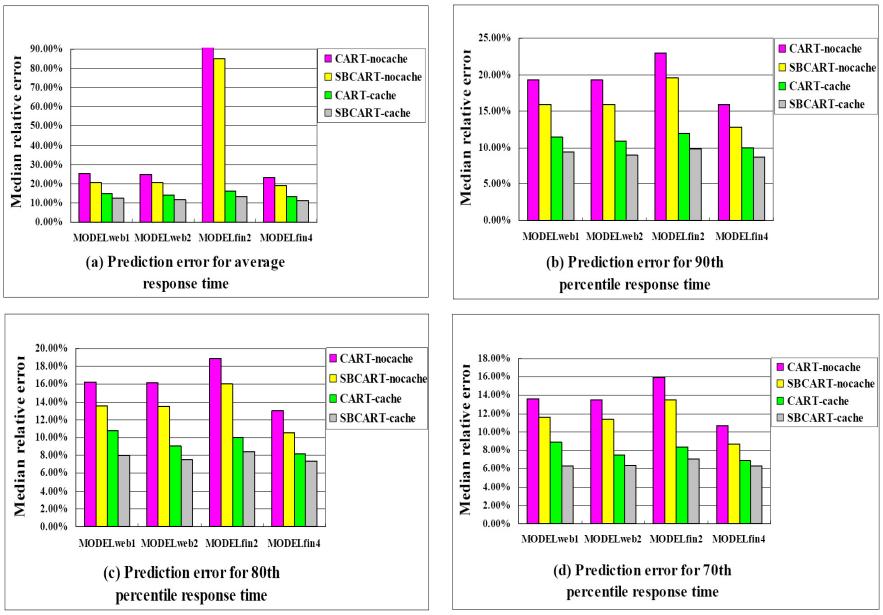

Fig. 4. Comparison of SBCART and CART on four traces: $Model_{Fin2}$, $Model_{Fin4}$, $Model_{Web1}$, $Model_{Web2}$. CART-nocache shows that cache information (Hit) is not considered in feature vector and CART-cache shows that cache information is used as a measure in feature vector.

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 defined 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 error 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 $Nmin$ of CART is set to 10.

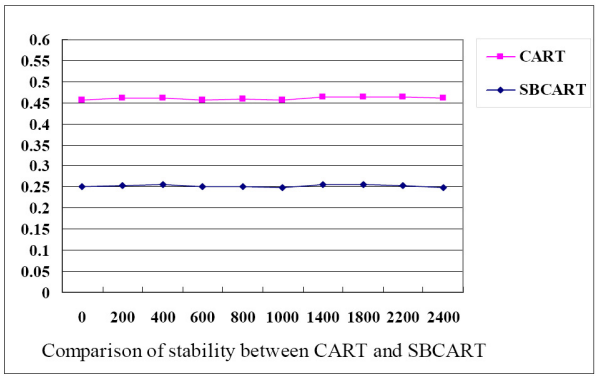

Fig. 5. Comparison of stability between CART and SBCART. X-coordinate describes the number of training data missing from the first 5000 instances of WebSearch-user1 trace and its testing data from another 5000 instances from the same trace.

3.4 Predictors in Comparison

Figure 4 compares the median relative error of the two predictors (SBCART and CART) in modeling a Seagate ST32171W disk on Financial ($Model_{Fin2}$, $Model_{Fin4}$) and WebSearch ($Model_{web1}, Model_{web2}$) traces respectively. Overall, the SBCART-based device models provide good prediction accuracy in predicting the average, 90th, 80th and 70th percentile response times compared to CART predictors. Several more detailed observations can be made.

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%, 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 reduce about 10% on $Model_{Web1}$, $Model_{Web2}$, $Model_{Fin4}$, but about 75% on $Model_{Fin2}$. We also observed that the traces of $Model_{Web1}$, $Model_{Web2}$, $Model_{Fin4}$ are relatively more random and the trace of $Model_{Fin2}$ is relatively more sequential, Therefore, sequential workloads like Financial-user2 are more easily affected by the caching effect so that adding the cache information greatly reduced the prediction error.

Second, SBCART can improve the accuracy and stability of CART. As shown in Figure 4(a), the SBCART-nocache can improve about 5% of accuracy compared to CART-nocache and the SBCART-cache can improve about 3% compared to CART-cache. We can see that using the measure Hit in feature vector and the ensemble method, the prediction accuracy increases by about 13% to

more random workloads and about 70% to more sequential workloads. As shown in Figure 5, the SBCRAT is more stable than CART with the training data changes, because selective ensemble models can reduce the variance of a single model.

Finally, it was more difficult to predict the high quantile response times. As shown in Figure 4(b), (c) and (d), the median relative error is reduced about 5%, 8% and 11% respectively compare to Figure 4(a). We can observe that SBCART are always more precise than CART with different quantile response times.

In summary, the SBCART-based models can give accurate predictions and more stable when the training and testing workloads have the same characteristics. The good accuracy indicates the SBCART and workload descriptions (feature vector) used in storage device performance prediction are more effective.

4 Conclusion

Storage device performance modeling is an important element 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 CART model. Based on bagging algorithms, we proposed our selective bagging classification and regression tree (SBCART) using the basic model CART. The SBCART model makes device models independent of the storage devices being modeled, so it can handle any type of devices. We preprocessed the UMass traces on which we built the SBCART models. In addition, we considered the caching effect and used it as a measure in feature vector to make good predictions.

There are several challenges to allow the black model to make accurate predictions : First, it is not a easy work to select the training data set that should provide a comprehensive and efficient coverage of workload characteristics; Second, workload characterization is still an open problem; Finally, the methodology itself has some deficiency. Thus, in future we will do more research considering above three aspects to make the accurate predictions.

Acknowledgements. The authors gratefully acknowledges the support of the Fundamental Research Funds for the Central Universities.

References

1. C. Ruemmler and J. Wilkes.: An introduction to disk drive modeling In IEEE Computer, 27(3) (1994)
2. B. Worthington, G. Ganger, and Y. Patt, Scheduling algorithms for modern disk drives.: In Proc. of *the ACM SIGMETRICS Conference* (1994)
3. The DiskSim Simulation Environment(v3.0), Parallel Data Lab, *URL: http://www.pdl.cmu.edu/DiskSim/*
4. J. L. Griffin, J. Schindler, S. W. Schlosser, J. S. Bucy, and G. R. Ganger.: Timing-accurate storage emulation. In Proc. of *FAST* , 75-88 (2002)

5. R. Barve, E. Shriver, P. B. Gibbons, B. K. Hillyer, Y. Matias, and J. S. Vitter.: Modeling and optimizing i/o throughput of multiple disks on a bus. In Proc. of *the ACM SIGMETRICS Conference* (1999)
6. M. Uysal, G. A. Alvarez, and A. Merchant.: A modular, analytical throughput model for modern disk arrays. In Proc. of *the 9th MASCOTS Conference* (2001)
7. J. Wilkes. The Pantheon storage-system simulator.: In Technical Report HPL-SSP-95-14, Storage Systems Program, Hewlett-Packard Laboratories (1996)
8. U. Aicheler. A visual user interface for the pantheon storage system simulator.: In Technical Report HPLSSP961, Storage Systems Program, Hewlett-Packard Laboratories (1996)
9. J. Wilkes, R. Golding, C. Staelin, and T. Sullivan.: The HP AutoRAID hierarchical storage system. In ACM Transactions on Computer Systems , 14(1) (1996)
10. P. Cao, S. B. Lim, S. Venkataraman, and J. Wilkes.: The TickerTAIP parallel RAID architecture. In ACM Transactions on Computer Systems , 12(3) (1994)
11. J. Schindler and G.R Ganger. Automated disk drive characterization.: In CMU SCS Technical Report CMU-CS-99-176 (1999)
12. E. Andenson. Simple table-based modeling of storage devices.: In Technical Report HPL-SSP-2001-04, HP Laboratories (2001)
13. T. Kelly, I. Cohen, M. Goldszmidt, and K. Keeton. Inducing models of black-box storage arrays.: In Technical Report HPL-SSP-2004-108, HP Laboratories (2004)
14. M. P. Mesnier, M. Wachs, R. R. Sambasivan, A. X. Zheng, and G. R. Ganger.: Modeling the relativetness of storage. In Proc. of *the ACM SIGMETRICS Conference* (2007)
15. M. Wang, K. Au, A. Ailamaki, A. Brockwell, C. Faloutsos, and G. R. Ganger.: Storage device performance prediction with cart models. In Proc. of *the 12th MASCOTS Conference* (2004)
16. L. Breiman, J. Friedman, C. J. Stone, and R.A. Olshen. Classification and regression trees.: In Chapman and Hall CRC (1984)
17. Kohavi, R. and Kunz, C. Option decision trees with majority votes.: In Proc. of the 14th International Conference on Machine Learning, San Francisco, CA: Morgan Kaufman (1997)
18. Bauer, E. and Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning (1999)
19. Breiman L., Bagging predictors.: Machine learning, 24(1): 123-140 (1996)
20. Freund, Y. and Schapire, R. E. Experiments with a new boosting algorithm.: In Proc. of the 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 and 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 Proc. of the 23th International Conference on Machine Learning , Pittsburgh , PA , 609 - 616 (2006)
24. Umass trace repository. *URL: http://traces.cs.umass.edu/index.php/Storage/Storage*
25. S. Jiang and X. Zhang.: LIRS: an ecient low inter-reference recency set replacement to improve buffer cache performance. In Proc. of the ACM SIGMETRICS Conference, (2002)