Carnegie Mellon University
Research Showcase @ CMU

Parallel Data Laboratory | Research Centers and Institutes

4-2009

Relative Fitness Modeling

Michael P. Mesnier
Intel Corporation

Matthew Wachs
Carnegie Mellon University

Raja R. Sambasivan
Carnegie Mellon University

Alice X. Zheng
Microsoft Research

Gregory R. Ganger
Carnegie Mellon University

Follow this and additional works at: http://repository.cmu.edu/pdl

This Article is brought to you for free and open access by the Research Centers and Institutes at Research Showcase @ CMU. It has been accepted for inclusion in Parallel Data Laboratory by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

DOI:10.1145/1498765.1498789

Relative Fitness Modeling

By Michael P. Mesnier, Matthew Wachs, Raja R. Sambasivan, Alice X. Zheng, and Gregory R. Ganger

Abstract

Relative fitness is a new approach to modeling the performance of storage devices (e.g., disks and RAID arrays). In contrast to a conventional model, which predicts the performance of an application's I/O on a given device, a relative fitness model predicts performance *differences* between devices. The result is significantly more accurate predictions.

1. INTRODUCTION

Relative fitness: the fitness of a genotype compared with another in the same gene system.

Managing storage within a data center can be surprisingly complex and costly. Large data centers have numerous storage devices of varying capability, and one must decide which application data sets (e.g., database tables, web server content) to store on which devices. Sadly, the state-of-the-art in Information Technology (IT) requires much of this to be done manually. At best, this results in an overworked system administrator. However, it can also lead to suboptimal performance and wasted resources.

Many researchers believe that automated storage management[2, 5] is one way to offer some relief to administrators. In particular, application workloads can be automatically assigned to storage devices. Doing so requires accurate predictions as to how a workload will perform on a given device, and a *model* of a storage device can be used to make these predictions. Specifically, one trains a model to predict the performance of a device as a function of the I/O characteristics of a given workload.[1, 7, 11, 13] Common I/O characteristics include an application's read/write ratio, I/O pattern (random or sequential), and I/O request size.

Though it sounds simple, such modeling has not been realized in practice, primarily because of the difficulty of obtaining workload characteristics that are good predictors of performance, yet also suitable for use in a model. For example, the I/O request size of an application is often approximated with an average, as opposed to the actual distribution (e.g., bimodal). Although such approximations reduce modeling complexity, they can lead to inaccurate predictions.

This article describes a new modeling approach called *relative fitness* modeling.[9, 10] A relative fitness model uses observations (performance and resource utilization) from one storage device to predict the performance of another, thereby reducing the dependence on workload characteristics. Figure 1 illustrates relative fitness modeling for two hypothetical devices A and B.

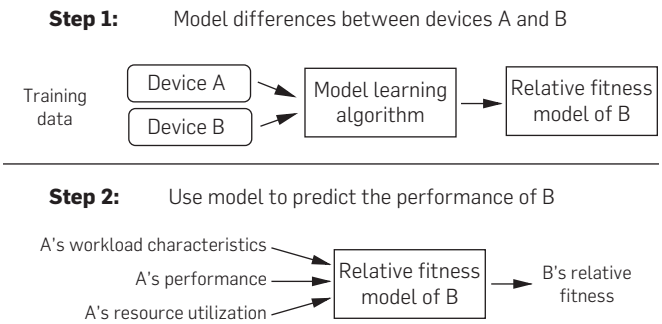
The insight behind relative fitness modeling is best obtained through analogy. When predicting your grade in a college course (a useful prediction during enrollment), it is helpful to know the grade received by a peer (his performance) and the number of hours he worked each week to achieve that grade (his resource utilization). Naturally, our own performance for a certain task is a complex function of the characteristics of the task and our ability. However, we have learned to make predictions relative to the experiences of others with similar abilities, because it is easier.

Applying the analogy, two storage devices may behave similarly enough to be reasonable predictors for each other. For example, they may have similar RAID levels, caching algorithms, or hardware platforms. As such, their performance may be related. Even dissimilar devices may be related in some ways (e.g., for a given workload type, one usually performs well and the other poorly). The objective of relative fitness modeling is to learn such relationships.

2. BACKGROUND

Storage performance modeling is a heavily researched area, including analytical models,[11] statistical or probabilistic models,[1, 7] and machine learning models.[10, 13] Models are either *white-box* or *black-box*. White-box models use knowledge of the internals of a storage device (e.g., drives, controllers, and caches), and black-box models do not. Given the complexity of modern-day storage devices,[12] black-box approaches are becoming increasingly attractive.

Figure 1: Using sample workloads, a model learns to predict how the performance of a workload changes between two devices (A and B). To predict the performance of a new workload on B, the workload characteristics, performance, and resource utilization (as measure on device A) are input into the model of B. The prediction is a performance scaling factor, which we refer to as B's "relative fitness."



A previous version of this research paper was published in the *Proceedings of the International Conference on Measurement and Modeling of Computer Systems* (San Diego, CA, June 2007), ACM, NY.

Perhaps the simplest of all black-box models is a numeric average. For example, the fuel efficiency of a car (average miles per gallon) and a soccer player's performance (average goals per game) are both black-box models. Of course, such models can be easily extended with workload characteristics (e.g., highway or city, home game or away), and an average can be maintained for each type of workload.

Table 1 shows a simple black-box model of a storage device (a table of performance averages), and Figure 2 shows the same information in a regression tree.[3] Both models are indexed using one workload characteristic (the average request size of the I/O that is issued to the storage device by the application), and both models must be *trained* with sample workloads in order to learn performance averages for various request sizes. Some form of interpolation is required when an exact match is not found in the model. For example, to predict the performance of a workload with 3KB requests, using Table 1, one might average the 2 and 4KB performance and predict 37MB/s. Of course, storage researchers have explored a number of workload characteristics in addition to request size, including the read/write ratio, multiprogramming level (queue depth), I/O inter-arrival delay, and spatial locality. More complex characteristics (e.g., I/O burstiness, spatio-temporal correlation) have also been investigated.

More formally, a model of a storage device i (white-box or black-box) can be expressed as a function F_i. During training, the inputs are the workload characteristics $\mathbf{WC}_i$ of an application running on device i and the output is a performance metric P_i (bandwidth, throughput, or latency):

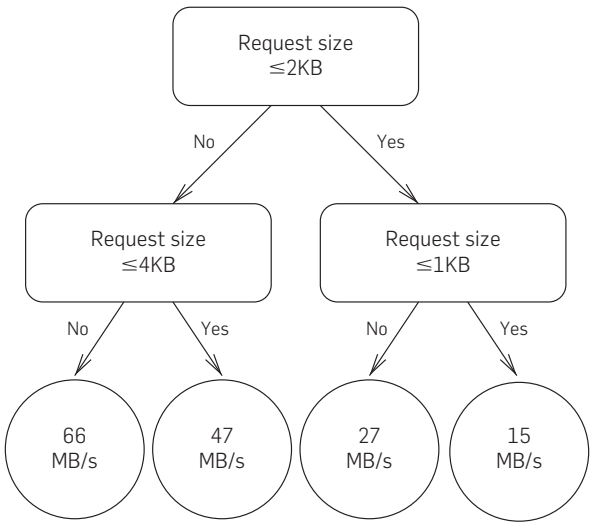
$$\mathrm{P}_i = F_i(\mathbf{WC}_i). \quad (1)$$

We refer to Equation 1 as an *absolute model*, to signify that the inputs $\mathbf{WC}_i$ are absolute, and not relative to some other device. However, in practice, one does not possess $\mathbf{WC}_i$, as this would require running the workload on device i in order to obtain them. Because running the workload to obtain $\mathbf{WC}_i$ obviates the need for predicting the performance of device i, one instead uses the characteristics $\mathbf{WC}_j$ obtained from some other storage device j. That is, the model assumes that the characteristics of a workload are static and will not differ across storage devices. More precisely, the model assumes that $\mathbf{WC}_i$ and $\mathbf{WC}_j$ are equivalent. However, this is not always a safe assumption.

Table 1: A table-based model that records the performance of a disk drive for sequentially-read data.

Request Size	Bandwidth
1KB	15MB/s
2KB	27MB/s
4KB	47MB/s
8KB	66MB/s

Figure 2: A regression tree that learns the performance of a disk drive for sequentially read data.



2.1. The challenges with absolute models

The primary challenges with absolute models relate to workload characterization, which has been an open problem for decades.[4] First, one must discover the performance-affecting characteristics of a workload. This can be challenging given the heterogeneity of storage devices.[8] For example, a storage array with a large cache may be less sensitive to the spatial access pattern than an array with little cache, so models of the devices would likely focus on different workload characteristics when predicting performance.

Second, one must manage the trade-off between expressiveness and conciseness. Most models expect numbers as input, and it can be challenging to describe complex workloads with just a few numbers. In effect, workload characterization compresses the I/O stream to just a few distinguishing features. The challenge is to compress the stream without losing too much information.

Third, and more fundamentally, an absolute model does not capture the connection between a workload and the storage device on which it executes. While the assumption of static workload characteristics (i.e., $\mathbf{WC}_i = \mathbf{WC}_j$) is safe for *open* workloads, where the workload characteristics are independent of the I/O service time, it is not safe for *closed* workloads. The most obvious change for a closed workload is the I/O arrival rate: if a storage device completes the I/O faster, then an application is likely to issue I/O faster. And other characteristics can change, such as the average request size, access pattern, read/write ratio, and queue depth. Such effects occur when file systems, page caches, and other OS middleware reside between an application and the storage device. Although the application may issue the same I/O, the characteristics of the I/O as seen by the storage device could change due to write reordering, aggregation and coalescing, caching, prefetching, and other interactions between an operating system and a storage device. For example, a slower device can result in a workload with larger inter-arrival times and larger write requests (due to request coalescing) when compared to a faster device.

Collectively, these challenges motivate the work presented in this article. Rather than attempt to solve the difficult problem of identifying workload characteristics that are expressive, yet concise and static across devices, we choose to use performance and resource utilization. That is, we use

the performance and utilization of device j to predict the performance of a different device i. Of course, such *relative* models must be built between each pair of devices, as performance and resource utilization are device-specific.

3. RELATIVE FITNESS MODELING

Relative fitness begins with an absolute model (Equation 1). Recall that a workload is running on device j, $\mathbf{WC}_j$ can be measured on device j, and we want to predict the performance of moving the workload to a different device i. The first objective of relative fitness is to capture the changes in workload characteristics from device j to i, that is, to predict $\mathbf{WC}_i$ given $\mathbf{WC}_j$. Such change is dependent on the devices, so we define a function $G_{j \to i}$ that predicts the workload characteristics of device i given j:

$$\mathbf{WC}_i = G_{j \to i}(\mathbf{WC}_j).$$

We can now apply G in the context of an absolute model F_i:

$$\mathrm{P}_i = F_i(G_{j \to i}(\mathbf{WC}_j)).$$

However, rather than learn two functions, the composition of F and G can be expressed as a single composite function $RM_{j \to i}$ which we call a *relative model*:

$$\mathrm{P}_i = RM_{j \to i}(\mathbf{WC}_j). \quad (2)$$

With each model now involving an origin j and target i, we can use the performance of device j ($\mathbf{Perf}_j$) and its resource utilization ($\mathbf{Util}_j$) to help predict the performance of device i. $\mathbf{Perf}_j$ is a vector of performance metrics such as bandwidth, throughput, and latency. $\mathbf{Util}_j$ is a vector of values such as the device's cache utilization, the hit/miss ratio, its network bandwidth, and its CPU utilization:

$$\mathrm{P}_i = RM_{j \to i}(\mathbf{WC}_j, \mathbf{Perf}_j, \mathbf{Util}_j). \quad (3)$$

In other words, one can now describe a workload relative to some other device. Recalling the analogy, if you want to predict your grade in a course that a colleague has already taken, you could simply have the colleague tell you his grade and the number of hours he worked each week. Other details of the course (workload characteristics) could be useful, but this information may not be as critical.

Next, rather than predict performance P_i, one can predict the performance ratio $\frac{\mathrm{P}_i}{\mathrm{P}_j}$, which may be a simpler function to model (e.g., perhaps device i is twice as fast as device j). We call such a model a *relative fitness model*:

$$\frac{\mathrm{P}_i}{\mathrm{P}_j} = RF_{j \to i}(\mathbf{WC}_j, \mathbf{Perf}_j, \mathbf{Util}_j). \quad (4)$$

To use the relative fitness model, one solves for P_i:

$$\mathrm{P}_i = RF_{j \to i}(\mathbf{WC}_j, \mathbf{Perf}_j, \mathbf{Util}_j) \times \mathrm{P}_j.$$

3.1. Model training

Training a relative fitness model requires workload samples from two devices i and j. Each workload sample can be described with three vectors: workload characteristics (**WC**), performance (**Perf**), and resource utilization (**Util**). During training, the goal is to learn relationships between the predictor variables ($\mathbf{WC}_j$, $\mathbf{Perf}_j$, and $\mathbf{Util}_j$) and the predicted relative fitness value $\frac{\mathrm{P}_i}{\mathrm{P}_j}$ (for some P in **Perf**).

Table 2(c) shows the format of the training data for a relative fitness model. For comparison, Table 2(b) shows that of a relative model which trains to predict performance (not a ratio), and Table 2(a) shows that of an absolute model which only requires samples from one storage device.

Given sufficient training data, one can construct a relative fitness model using a variety of learning algorithms. The problem falls under the general scope of *supervised learning*, where one has access to a set of predictor variables (**WC**, **Perf**, and **Util**), as well as the desired response (the relative fitness value). It is as though an oracle (or supervisor) gives the true output value for each sample, and the algorithms need only learn the mapping between input and output.

The domain of supervised learning problems can be further subdivided into classification (discrete-valued predictions) and regression (continuous-valued predictions). Relative fitness values are continuous, and there are many regression models in statistical literature. We choose to use classification and regression tree (CART) models, for their simplicity, flexibility, and interpretability.[3]

3.2. Summary and modeling cost

Whereas conventional absolute modeling constructs one model per device and assumes that the workload characteristics are static across devices, relative fitness modeling constructs two models for each pair of devices ($i \to j$ and $j \to i$) and implicitly models the changing workload characteristics. In addition, the relative approaches use performance and resource utilization when making predictions, thereby relaxing the dependency on expressive workload characteristics.

Of course, the cost of the relative approach is the additional model construction: $O(n^2)$ versus $O(n)$, where n is the number of storage devices. However, in our evaluation, model construction takes at most a few seconds. Moreover, models can be built and maintained by each storage device. That is, each device can

Table 2: Training data formats for the various models. The last column in each table is the variable that we train a model to predict. All other columns are predictor variables.

Sample	Predictor Variables	Predicted Variables
(a) Absolute model		
1	$\mathbf{WC}_{i,1}$	$\mathrm{P}_{i,1}$
2	$\mathbf{WC}_{i,2}$	$\mathrm{P}_{i,2}$
N	$\mathbf{WC}_{i,n}$	$\mathrm{P}_{i,n}$
(b) Relative model		
1	$\mathbf{WC}_{j,1}$ $\mathbf{Perf}_{j,1}$ $\mathbf{Util}_{j,1}$	$\mathrm{P}_{i,1}$
2	$\mathbf{WC}_{j,2}$ $\mathbf{Perf}_{j,2}$ $\mathbf{Util}_{j,2}$	$\mathrm{P}_{i,2}$
N	$\mathbf{WC}_{j,n}$ $\mathbf{Perf}_{j,n}$ $\mathbf{Util}_{j,n}$	$\mathrm{P}_{i,n}$
(c) Relative fitness model		
1	$\mathbf{WC}_{j,1}$ $\mathbf{Perf}_{j,1}$ $\mathbf{Util}_{j,1}$	$\mathrm{P}_{i,1}/\mathrm{P}_{j,1}$
2	$\mathbf{WC}_{j,2}$ $\mathbf{Perf}_{j,2}$ $\mathbf{Util}_{j,2}$	$\mathrm{P}_{i,2}/\mathrm{P}_{j,2}$
N	$\mathbf{WC}_{j,n}$ $\mathbf{Perf}_{j,n}$ $\mathbf{Util}_{j,n}$	$\mathrm{P}_{i,n}/\mathrm{P}_{j,n}$

construct $O(n)$ models that predict its fitness relative to all other devices. As such, the computational resources for maintaining the models can be made to scale with the number of devices. Also, in large-scale environments, certain collections of devices will be identical and can share models.

4. EVALUATION

The motivation and advantages of relative fitness modeling can be stated as four hypotheses:

- **Hypothesis 1.** Workload characteristics can change across storage devices ($\mathbf{WC}_i \neq \mathbf{WC}_j$) and reduce the accuracy of an absolute model.
- **Hypothesis 2.** A relative model (Equation 2) can reduce the inaccuracies that result from changing characteristics.
- **Hypothesis 3.** Performance and resource utilization can improve prediction accuracy (Equation 3).
- **Hypothesis 4.** Performance ratios (Equation 4) can provide better accuracy than raw performance values (Equation 3).

To test these hypotheses, the accuracy of various CART models can be compared: absolute models (Equation 1), relative models (Equation 2), relative models with performance (Equation 3), and relative fitness models (Equation 4).

4.1. Setup

Experiments are run on an IBM x345 server (dual 2.66GHz Xeon, 1.5GB RAM, GbE, Linux 2.6.12) attached to three iSCSI storage arrays. The arrays have different hardware platforms, software stacks, and are configured with different RAID levels.[a] More specifically,

- **Vendor A** is a 14-disk RAID-50 array with 1GB of cache (400GB 7200 RPM Hitachi Deskstar SATA)
- **Vendor B** is a 6-disk RAID-0 array with 512MB of cache (250GB 7200 RPM Seagate Barracuda SATA)
- **Vendor C** is an 8-disk RAID-10 array with 512MB of cache (250GB 7200 RPM Seagate Barracuda SATA)

The server attaches to each array using an iSCSI device driver[6] that contains counters (below the file system and page cache) for characterizing workloads and measuring their performance. A synthetic workload generator[6] is used to generate numerous workload samples, which we refer to as a *fitness test*. These samples are used to train and test the CART models. Similar results from other workloads (e.g., Postmark, TPC-C), as well as details on the CART algorithm (e.g., tree construction and pruning), can be found in our conference paper.[10]

4.2. Fitness test results

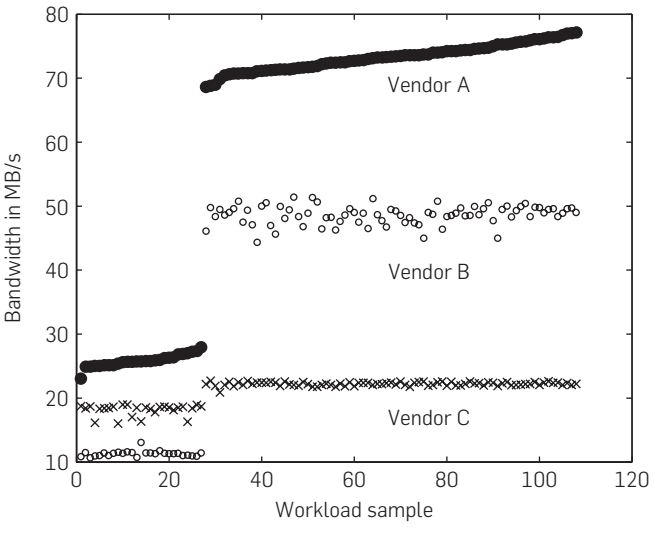
The fitness test compares the performance of the storage arrays across a wide range of workloads (various runs of the workload generator). The measured workload characteristics (**WC**) of each sample include the write percent, the write and read request sizes, the write and read randomness (average seek distance, in blocks, per I/O), and the queue depth (average number of outstanding I/Os). The performance (**Perf**) of each sample run is the average bandwidth (MB/s), throughput (IO/s), and latency (ms). Resource utilization (**Util**) is not used in this evaluation, as this requires modifying storage device software to which we did not have access. A total of 3000 samples are generated.

Over all 3000 samples, Vendor A is the fastest array with an average bandwidth of 25 MB/s, an average throughput of 624 IO/s and an average latency of 37ms. Vendor B is the second fastest (17 MB/s, 349 IO/s, and 45ms). Vendor C is the third (14 MB/s, 341 IO/s, and 84ms). Although Vendor A is the fastest, on average, it is not necessarily the fastest for all sample workloads in the fitness test. There are samples where Vendors B and C do better than A (relative fitness values greater than 1) and cases where they do worse (values lesser than 1). In short, the relative fitness of a device can vary with the workload characteristics.

As an example of how devices can behave similarly, Figure 3 illustrates how the sequential write bandwidth for each array varies for different request sizes and queue depths. From the 3000 samples, we show only the sequential write workloads. There are 120 such samples, sorted by the performance of Vendor A. The graph illustrates the similar performance of the arrays. In particular, the prominent discontinuity in the graph is shared by all arrays (a drop in performance when there are only one or two outstanding requests). Also note how Vendor B is faster than Vendor C to the left of the discontinuity, but slower to the right. Such piecewise functions are ideally suited for CART models.

In support of Hypothesis 1, Table 3 contains averages for the workload characteristics of each sample. Note the variance across devices ($\mathbf{WC}_i \neq \mathbf{WC}_j$), most notably the average spatial randomness of writes, which varies by as much as 38%. In particular, Vendor A experiences the most randomness (an average seek distance of 321MB per write), Vendor B the second most

Figure 3: Device similarity. The performance of each array changes similarly, indicating that the performance of one array is a good predictor of another.



[a] RAID level 0 is striping, 1 is mirroring, 5 is striping with parity, 10 is striping over mirrored pairs (4 in this case), and 50 is striping over RAID-5 parity arrays (2 in this case).

(250MB), and Vendor C the third (233MB). Although masked by the averages in Table 3, the request sizes and queue depths also vary across storage devices for some of the sample workloads.

4.3. Interpreting the models

Of the fitness test samples, 75% are used to build the CART models and 25% are reserved for testing. Figure 4 illustrates four of the bandwidth models, one of each modeling type. For readability, each tree is pruned to a depth of 4, resulting in at most 8 leaves (prediction rules).

The models in Figure 4 predict the performance of Vendor C given observations from Vendor A. CART builds trees top-down, so nodes near the top of the tree have the most information. In particular, note how the relative and relative fitness models learn that the bandwidth of Vendor A is the best predictor of the bandwidth of Vendor C.

As an example of how to use the trees to make a prediction, suppose a workload is running on Vendor A and we want to predict its performance on Vendor C. Also suppose that the workload, as measured by Vendor A, has an average read seek of 2048 blocks, a request size of 64KB, a write percentage <0.5%, a bandwidth of 83 MB/s, and a throughput of 1328 IO/s. The absolute model will predict 75.0 MB/s (see highlighted path in Figure 4a), the relative model (Figure 4b) predicts 75 MB/s, the relative model trained with performance (Figure 4c) predicts 65.0 MB/s, and the relative fitness model (Figure 4d) predicts that Vendor C is 63% of Vendor A or 51 MB/s.

4.4. Modeling accuracy

Recall that 25% of the fitness test samples are reserved for testing, so the performance of each sample is known and can be used to determine the relative error of each prediction.

Table 3: Fitness test workload characteristics.

WC	Vendor A	Vendor B	Vendor C	Maximum Difference (%)
Write percent	40	39	38	5.2
Write size (KB)	61	61	61	0
Read size (KB)	40	41	41	2.5
Write seek (MB)	321	250	233	38
Read seek (MB)	710	711	711	0
Queue depth	23	22	21	9.5

For example, if the performance of Vendor C (for a given test sample) is 45MB/s and the prediction is 51MB/s, the relative error is $\frac{|45-51|}{45} \times 100$, or 13.3%. To quantify the average error of each model (over all test samples), we report the average relative error of the predictions.

As a baseline, Table 4 contains the average relative error of the bandwidth, throughput, and latency predictions for the absolute model. The table is organized pairwise. Workload characteristics ($\mathbf{WC}_j$) are obtained from one array and predictions (P_i) are made for another. For example, the average relative error of the bandwidth predictions when characterizing on Vendor A and predicting for Vendor C is 22% (the top right cell in Table 4).

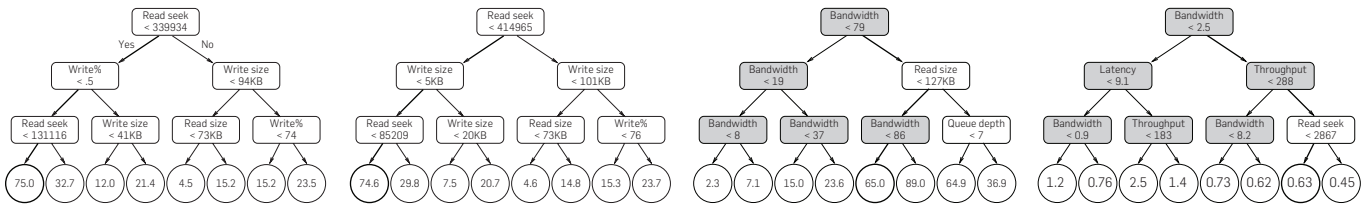
The first observation is that the most accurate predictions occur when the workload is characterized on the same device for which the prediction is being made ($\mathbf{WC}_j = \mathbf{WC}_i$), as indicated by the diagonals in bold. However, if one runs a workload on device *i* to obtain $\mathbf{WC}_i$, there is no need to make a prediction. These predictions are only included to illustrate how changing workload characteristics ($\mathbf{WC}_j \neq \mathbf{WC}_i$) can affect prediction accuracy. For example, the bandwidth prediction error for Vendor A increases from 23% (when characterized on Vendor A) to 29% (when characterized on Vendor B) and 30% (when characterized on Vendor C). Therefore, Table 4 supports Hypothesis 1: changing workload characteristics can affect prediction accuracy.

Figure 5, in contrast, shows the prediction errors for the relative model (Equation 3) and relative fitness model

Table 4: Prediction error for the absolute model. Workload characteristics ($\mathbf{WC}_j$) are obtained from array *j* and predictions (P_i) are made for array *i*.

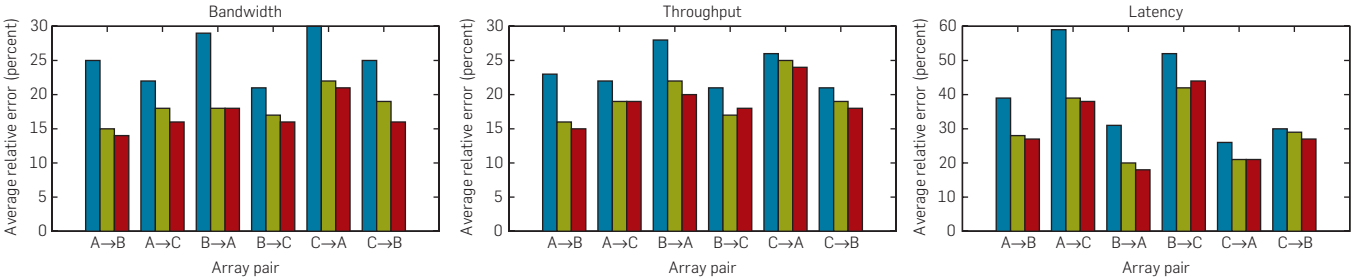
	$\mathbf{Bandwidth}_A$ (%)	$\mathbf{Bandwidth}_B$ (%)	$\mathbf{Bandwidth}_C$ (%)
WC_A	**23**	25	22
WC_B	29	**19**	21
WC_C	30	25	**17**
	$\mathbf{Throughput}_A$ (%)	$\mathbf{Throughput}_B$ (%)	$\mathbf{Throughput}_C$ (%)
WC_A	**20**	23	22
WC_B	28	**15**	21
WC_C	26	21	**14**
	$\mathbf{Latency}_A$ (%)	$\mathbf{Latency}_B$ (%)	$\mathbf{Latency}_C$ (%)
WC_A	**20**	39	59
WC_B	31	**21**	52
WC_C	26	30	**21**

Figure 4: CART models trained to predict the bandwidth of Vendor C. The leaf nodes in the absolute and relative models represent bandwidth predictions; the leaves in the relative fitness model are relative fitness predictions. The absolute model (a) and relative model (b) only use workload characteristics from Vendor A; the relative model (c) with Perf. and relative fitness model (d) also use Vendor A's performance (shaded). All but the absolute model account for changes in the workload characteristics between Vendors A and C.



(a) Absolute (b) Relative (c) Relative with Perf. (d) Relative fitness

Figure 5: Errors of the absolute (first bar in each graph), relative (second), and relative fitness models (third), where *i*→*j* indicates that workload characteristics from array *j* are used to predict the performance of array *i*.



(Equation 4), both of which use performance information to make a prediction; the absolute model prediction errors from Table 4 are shown for comparison. Overall, the relative fitness models reduce the average bandwidth prediction error from 25% to 17%, throughput from 24% to 19%, and latency from 40% to 29%. Moreover, in most cases, the relative fitness model is slightly more accurate than the relative model. These results confirm that models trained with performance can be more accurate (Hypotheses 2 and 3) and that predicting ratios can further improve accuracy (Hypothesis 4).

In summary, workload characteristics can change across devices and impact the accuracy of an absolute model (Hypothesis 1), a relative model can reduce the inaccuracy due to changing workloads (Hypothesis 2), the performance of one device can be used to predict the performance of another (Hypothesis 3), and performance ratios can be better predictors than raw performance values (Hypothesis 4).

5. CONCLUSION

By modeling storage devices relative to one another, relative fitness models can use the observed performance and resource utilization of a workload on one device when making predictions for another. Such modeling addresses many of the challenges associated with workload characterization and, therefore, brings automated storage management a step closer to becoming a practical solution for the data center.

In addition, relative fitness models may find a broader applicability outside of storage management. In the same manner that storage models can exploit performance and resource utilization, so too can models of other data center resources (e.g., application servers).

Acknowledgments

We thank Christos Faloutsos, Arif Merchant, Mic Bowman, and the PDL Consortium (APC, Cisco, Data Domain, EMC, Facebook, Google, Hewlett-Packard, Hitachi, IBM, Intel, LSI, Microsoft, NetApp, Oracle, Panasas, Seagate, Symantec, and VMware). We thank EqualLogic, Intel, IBM, LeftHand Networks, NetApp, Open-E, Seagate, and Sun for equipment donations. This research is sponsored in part by the NSF (CNS-0326453, CCF-0621499, and 0431008) and the Army Research Office (DAAD19-02-1-0389). Matthew Wachs is supported in part by an NDSEG Fellowship.

References

1. Anderson, E. *Simple Table-Based Modeling of Storage Devices*. Technical Report HPL-SSP-2001-4, Hewlett-Packard Laboratories, July 2001.
2. Anderson, E., Hobbs, M., Keeton, K., Spence, S., Uysal, M., Veitch, A. Hippodrome: running circles around storage administration. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies* (*FAST 02*) (Monterey, CA, January 2002), The USENIX Association.
3. Breiman, L., Friedman, J., Olshen, R.A., Stone, C.J. *Classification and Regression Trees*. Chapman and Hall, New York, NY, 1984.
4. Ganger, G.R. Generating representative synthetic workloads: an unsolved problem. In *Proceedings of the 21st International Computer Measurement Group Conference* (*CMG*) (Nashville, TN, December 1996), Computer Measurement Group (CMG).
5. Ganger, G.R., Strunk, J.D., Klosterman, A.J. *Self-* Storage: Brick-Based Storage with Automated Administration*. Technical Report CMU-CS-03-178, Carnegie Mellon University, August 2003.
6. Intel Corporation. Open Storage Toolkit. http://www.sourceforge.net/projects/intel-iscsi.
7. Kelly, T., Cohen, I., Goldszmidt, M., Keeton, K. *Inducing Models of Black-Box Storage Arrays*. Technical Report HPL-2004-108, Hewlett-Packard Laboratories, June 2004.
8. Kurmas, Z., Keeton, K. Using the distiller to direct the development of self-configuration software. In *Proceedings of the 1st International Conference on Autonomic Computing* (*ICAC-04*) (New York, NY, May 2004), IEEE Computer Society.
9. Mesnier, M. *On Modeling the Relative Fitness of Storage*. Ph.D. dissertation. Carnegie Mellon University, Pittsburgh, PA, December 2007.
10. Mesnier, M., Wachs, M., Sambasivan, R.R., Zheng, A., Ganger, G.R. Modeling the relative fitness of storage. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems* (*SIGMETRICS 2007*) (San Diego, CA, June 2007), ACM.
11. Uysal, M., Alvarez, G.A., Merchant, A. A modular, analytical throughput model for modern disk arrays. In *Proceedings of the 9th International Symposium on Modeling Analysis and Simulation of Computer and Telecommunications Systems* (*MASCOTS-2001*) (Cincinnati, OH, August 2001), IEEE/ACM.
12. Varki, E., Merchant, A., Xu, J., Qiu, X. Issues and challenges in the performance analysis of real disk arrays. *IEEE Transactions on Parallel and Distributed Systems* (*TPDS*) *15*, 6 (June 2004), 559–574.
13. Wang, M., Au, K., Ailamaki, A., Brockwell, A., Faloutsos, C., Ganger, G. R. Storage device performance prediction with CART models. In *Proceedings of the 12th International Symposium on Modeling Analysis and Simulation of Computer and Telecommunications Systems* (*MASCOTS-2004*) (Volendam, The Netherlands, October 2004), IEEE.

Michael P. Mesnier
Intel Corporation
Hillsboro, OR.

Alice X. Zheng
Microsoft Research
Redmond, WA.

Matthew Wachs
Carnegie Mellon University
Pittsburgh, PA.

Gregory R. Ganger
Carnegie Mellon University
Pittsburgh, PA.

Raja R. Sambasivan
Carnegie Mellon University
Pittsburgh, PA.

© ACM 0001-0782/09/0400 $5.00