Modeling the relative fitness of storage devices

Michael Mesnier
Intel Corp., Carnegie Mellon

Matthew Wachs, Gregory Ganger
Carnegie Mellon

CMU-PDL-05-106
August 2005

Parallel Data Laboratory
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Relative fitness modeling is a new approach for predicting the performance and resource utilization of a workload when running on a particular storage device. In contrast with conventional device models, which expect device-independent workload characteristics as input, a relative fitness model makes predictions based on characteristics measured on a specific other device. As such, relative fitness models explicitly account for the workload changes that almost always result from moving a workload across storage devices—for example, higher I/O performance usually leads to faster application execution which results in higher I/O rates. Further, relative fitness models allow service observations (e.g., performance and resource utilizations) from the measured device to be used in making predictions on the modeled device—such observations often provide more predictability than basic workload characteristics. Overall, we find that relative fitness models reduce prediction error by over 60% on average when compared to conventional modeling techniques.

Keywords: storage device, self-managing, CART models, regression trees

Relative fitness (RF): the fitness of a genotype compared with the fitness of another genotype in the same gene system [8].

1 Introduction

Storage administration continues to be overly complex and costly. One challenging aspect of administering storage, particularly in large infrastructures, is deciding which volumes to store on which devices. Among other things, this decision involves balancing loads, matching workloads (i.e., request streams on volumes) to device strengths, and ensuring that performance goals are satisfied. Storage administration currently relies on experts who use rules-of-thumb to make educated, but *ad hoc*, decisions. With a mechanism for predicting the performance and resource utilization for any given workload and storage device, one could automate this decision-making process [1, 3, 6].

Previous research on such prediction and automation focuses on per-device models that take as input device-independent workload characteristics (e.g., request arrival rate and read/write ratio). The output from such a model is a prediction for device performance (e.g., latency). Many envision these device models being constructed automatically in a *black-box* manner. Given pre-deployment measurements on a device, one can train a statistical model [4, 20] to calculate the expected performance of the device as a function of a workload's characteristics. We refer to this as the *conventional* modeling technique.

Though it sounds simple, the above approach has proven quite difficult to realize in practice, for several reasons. First, workload characterization has been an open problem for decades [9, 14, 20]. Describing a complex workload in terms of concise characteristics, without losing information about burstiness or spatio-temporal locality, remains a challenge. Second, and more fundamentally, the conventional modeling technique does not capture the connection between a workload and the storage device on which it executes. Generally speaking, application performance depends on storage performance. If applications progress faster or slower, their I/O rates change in proportion. Such device-dependent feedback is not captured in conventional models.

This paper proposes a new approach based on *relative fitness* models. A relative fitness model predicts how one device will perform based on workload characteristics as measured on a second device. Such models explicitly capture the device-dependency of their input workloads and model the feedback between workloads and devices. Further, since the workload characteristics are measured on a specific device, relative fitness models can just as easily measure and use *service observations* (e.g., response times and resource utilizations) in addition to basic workload characteristics. Often, such observations about how another device handles a workload yield greater information than imperfect workload characterizations—for example, one may not know how to concisely describe access locality, but a workload that experiences a high cache hit rate on one device is likely to experience a similar hit rate on another.

This paper describes the mechanics of constructing relative fitness models and evaluates their effectiveness. Rather than train only one model for each individual device, one constructs two models for each pair of devices—one for translating measurements on the first to predictions on the second and one for going in the other direction. Our best models capture similarities (and differences) between devices by predicting scaling factors (e.g., device $D1$ is X% faster than device $D2$ for random workloads), rather than absolute values. Such scaling factors adapt well to unseen workloads, and can also help with identifying pools of similar storage devices (i.e., identical devices will have a scaling factor of 1.0), thereby reducing the number of models that must be maintained.

Results from experimentation with different storage devices and workloads demonstrate the value of relative fitness models. Workloads are clearly device-dependent, as we observe arrival rates changing as much as 50% when moving from one device to another. Other characteristics, such as average request sizes and read/write ratios, change with device transitions as well. Accounting for these changes is essential.

Most importantly, we find that relative fitness models consistently provide more accurate predictions than conventional models for our sample workloads. We see over a 60% reduction in prediction error, on average, for the micro-benchmarks used in model training and testing. In experiments with the Postmark benchmark, we observe that the relative approach predicts average throughput within 8% whereas the conventional model mispredicts by a factor of two.

The remainder of this paper is organized as follows. Section 2 describes the conventional and relative approaches in greater detail. Section 3 describes our experimental setup. Section 4 makes a case for relative fitness models, demonstrating the device-dependence of workloads and the predictive value of service observations. Section 5 details the mechanics of our relative fitness models. Section 6 evaluates the efficacy and discusses the nuances and uses of relative fitness models.

2 Background and motivation

Storage system configuration is a complex optimization problem. Storage devices must be configured appropriately (e.g., RAID levels and LUN sizes), and workloads must be assigned to them (e.g., file systems, databases, email). Consider the "simple" task of assigning 8 workloads to 8 different storage devices, each of which can be configured in one of two ways, say RAID1 or RAID5. Even with the simplifying constraint of one workload per device, there are still over one million ways of configuring the devices and assigning each a workload (i.e., 2^8 ways of configuring the devices and $8!$ ways of assigning them work).

Such tasks can be automated through system software, but given the large solution space, doing so requires a method for quickly determining how storage will respond to a particular workload. For this determination, device models are needed.

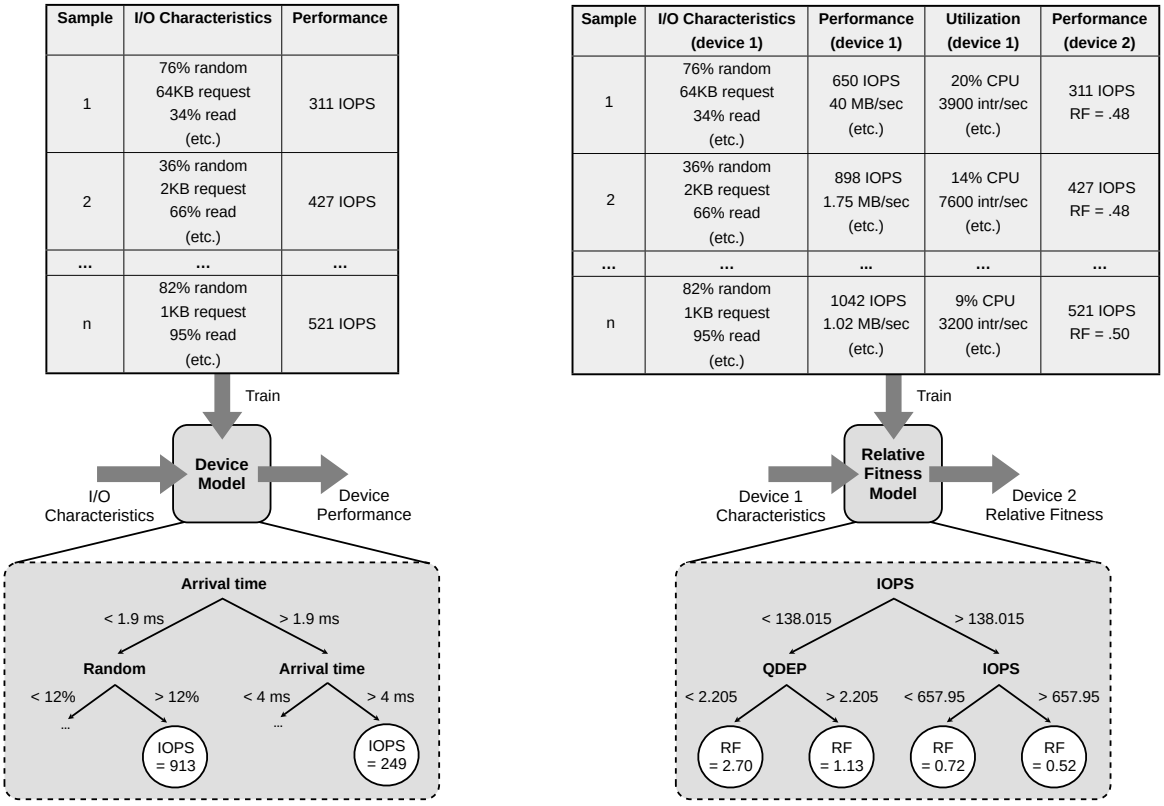
2.1 Conventional device modeling

A device model is a representation of a storage device used to predict a workload's performance. The input into the model is a description of the workload, and the output is an expected performance value (e.g., latency or throughput). For example, a model of a disk drive may predict an average throughput of 166 IO/sec for a workload that is mostly random.

Conventional models can be analytical or statistical. Analytical models are based on queueing theory. Given the internal structure of a storage device, mathematical formulas are constructed to offer a compact representation of system performance, and the model parameters are calibrated by observing a storage device's performance over a wide range of workloads [17, 16, 18]. Statistical models, on the other hand, require no internal knowledge of the storage device but must be trained over a similarly wide range of workloads [20]. Yet another option is a "table-based" approach that memoizes performance values for a particular type of workload and interpolates between these values for unseen workloads [2]. All three modeling approaches, from an input/output perspective, work in a similar manner. As input, they take a concise set of workload characteristics (i.e., discrete or continuous values). As output, they predict a performance metric for workloads matching the description. Figure 1a illustrates the conventional approach in the context of a classification and regression tree, one type of statistical model.

2.1.1 Workload characterization

Workload characterization is the process of describing the I/O accesses of a workload, and lies at the core of model building. I/O has at least three distinguishing components: operation mix (i.e., read vs. writes), spatial locality (e.g., which blocks are accessed), and temporal locality (e.g., when they are accessed).



(a) Conventional (b) Relative

Figure 1: The conventional and relative approaches to device modeling. Given a sufficient number of sample workloads, a model can be constructed to calculate (predict) the performance of workload, given its characteristics. In the conventional modeling approach (a), a workload's I/O characteristics are input into a device model that predicts throughput, latency, or bandwidth. The device from which the workload characteristics are obtained is not captured in the model. In the relative approach (b) a workload's I/O, performance, and utilization characteristics, as measured on a specified device, are input into a model that predicts a scaling factor relative to that device. The model can be used to predict changes in performance and utilization (e.g., CPU utilization), or even workload changes, such as the inter-arrival time.

Some aspects are quite complex and involve correlations, such as the "burstiness" [11] and spatio-temporal correlation [19], making them difficult to obtain and concisely represent.

Note that the characteristics of a workload as seen by a storage device can be radically different from the I/O requests issued by the application. Such is the case when the storage device and the application are separated by entities such as file systems, page caches, or any other middleware that can delay or modify the I/O. When we discuss workload characteristics, we are referring to those as seen at the device. For the purposes of storage system configuration and tuning, the workload as seen by storage is what really matters.

Some common workload characteristics include averages for each of the following:

- The time between I/O requests (inter-arrival delay)
- The percentage of operations that are writes vs. reads
- The request size
- The randomness of the workload (e.g., average inter-request distance in blocks)

In general, workload characterization tries to capture all "important" features of a workload as they relate to storage performance. The trick is determining which characteristics are the best predictors of performance. For example, whether write requests are random is more critical for a disk drive than for a RAID array with a large non-volatile cache; models of the two would likely focus on different workload characteristics for predicting device performance. This adds an additional challenge to workload characterization: one must distill the key performance-affecting attributes for each device in question [13].

2.2 Challenges in conventional modeling

In conventional modeling, workload characteristics are meant to be device-independent, the assumption being that moving a workload across devices will not significantly change its I/O. This is a reasonable assumption in cases where the devices in question are very similar (e.g., slightly different RAID configurations) or when the workload is *open*.

An open workload is one whose I/O accesses are invariant with respect to the performance of the storage device. In an open system, feedback from storage performance (e.g., request latency and throughput) has no effect on application behavior. For example, a video player may issue I/O requests at a constant rate, regardless of the speed of the device storing its media, so the arrival rate can easily be described in terms of an *arrival process* (e.g., periodic) and reasoned about analytically. Open systems are mathematically tractable, and form the foundation for many analytical device models.

In a *closed* system, however, the I/O accesses of a workload may change with storage performance, resulting in I/Os of different sizes, frequencies, or types. For example, a slower device can result in a workload with larger inter-arrival times and write requests when compared to the same workload running on a faster device. This effect should be intuitive. If applications ever block on I/O (e.g., sequentially reading a file), then the faster the storage device completes the I/O, the faster the application will issue the next I/O. In general, any application affected by the speed of its storage (e.g., compilation, backup, file search, etc.) is operating, at least in part, in a closed fashion. Further, an application's temporal access pattern is not the only thing that can change. As we will later show, even characteristics such as the operation mix and average request size can change in a closed system.

In summary, we see two primary challenges with workload characterization and conventional device modeling: concisely representing workload characteristics and capturing performance-workload feedback. Together, these challenges led us down an alternate path.

2.3 Models of relative fitness

A self-configuring storage system should assume neither open systems nor similar storage devices. Indeed, data centers comprise devices from a variety of vendors, the devices can be configured in a variety of ways, and most workloads are at least partially closed in nature [10]. As we will show, the conventional approach to device modeling will not work well in such situations. The largest shortcoming of conventional modeling is a reliance on a set of presumably device-independent workload characteristics, or rather, the fact that device-dependent changes in these characteristics are not anticipated or explicitly modeled.

One could address this concern by explicitly modeling how workloads change when moving across two devices, say, $D1$ and $D2$. For example, it may be the case that the write size, on average, increases by 50%. Given enough sample workloads, this behavior could be learned and the workload characteristics, as observed by $D1$, could be adjusted before they are input into the model of $D2$. Such an adjustment could account for workload change due to close-loop interactions between the application and storage. The alternative, not taking such feedback into consideration, may result in inaccurate predictions.

The above solution unfortunately require two models (and two predictions): one model to predict the changes in a workload, and second (conventional) model to predict performance based on these new characteristics. Rather than explicitly predicting workload change and inputing adjusted workload characteristics into a conventional model, we instead model each device *relative* to the workload characteristics seen by a second device — the device that the workload is moving from.

We construct a model of $D2$ that takes as input the workload characteristics as seen by $D1$. As such, the model of $D2$ explicitly accounts for changes in workload when moving from $D1$ and $D2$, because it was constructed relative to $D1$. So, in contrast with conventional modeling, we construct a model between each pair of devices, or at least the pairs we expect to be moving workloads between. In the conventional approach, only one model is constructed per device.

In addition to coping with changes in workloads, device-dependent models introduce an additional opportunity: *service observations*. We define service observations as the performance and resource utilization of a given workload as measured on a specific device. Performance includes average throughput, bandwidth, and latency of requests. Utilization includes CPU utilization, queue depth, context switch rate, and interrupt rate. So, instead of characterizing a workload by just its workload characteristics, we can now describe a workload in terms of its performance and resource utilization (e.g., that it consumes 50% of the CPU on $D1$ and performs at 200 IOPS). Most importantly, service observations allow us to model devices relative to one another (e.g., that device $D2$ is 30% faster than $D1$ for sequential writes and requires half the CPU). In other words, we can model a device's fitness for a particular workload in relation to other devices. We call this the *relative fitness* (RF) of a device.

Relative fitness models therefore require training data from two devices, *D1* and *D2*. The inputs into the model are the workload characteristics and service observations as seen by *D1*, and the output is a workload characteristic or service observation expected on *D2*. That is, RF models can be trained to predict performance, resource utilization, or workload characteristics. Such models allow for insightful queries during resource provisioning. For example, a self-tuning storage system may ask the following questions when deciding whether or not to place a workload on $D2$: "what is its predicted CPU utilization?, what is its predicted throughput?, will the average queue depth increase or decrease?" Figure 1b illustrates the concept.

Service observations create a strong relationship between a workload and the device on which it was characterized. As we will later show, these additional characteristics are so rich in information that, in some cases, the need for basic workload characteristics is eliminated altogether.

	DISK	SSD	RAID
Server	Dell PowerEdge 650	IBM x345	IBM x345
Processor	P4 2.7 GHz	Dual Xeon 2.7 GHz	Dual Xeon 2.7 GHz
Memory	1GB	1.5GB	1.5G
Network	PRO/1000	PRO/1000	PRO/1000
Storage	Cheetah 10K	N/A	IBM-ESXS 10K
OS	Linux 2.4	Linux 2.4	Linux 2.4

Table 1: Storage platforms.

3 Experimental setup

The previous sections claim that workload characteristics change across storage devices, such change is predictable and can be modeled, and the relative fitness of storage devices can be predicted. In the sections that follow, we conduct a set of experiments to support each claim. This section describes our experimental setup.

3.1 The storage platforms

Our experiments use three storage devices: a disk drive (DISK), a solid-state disk (SSD), and a RAID-0 array (RAID). These three target devices are chosen for their heterogeneity, in order to illustrate the benefits of relative fitness models. Table 1 describes the storage platforms in more detail.

All devices are connected to the host machine using the Internet SCSI (iSCSI) protocol [15] and are implemented using an open source distribution [12]. The target code is a user-level library with APIs for receiving SCSI commands, returning status codes, and transferring any data. On top of this library, we implement each device.

DISK is implemented with a single SCSI disk using the SCSI generic interface in Linux. SSD is implemented entirely in RAM. RAID is is implemented using the software RAID stack in Linux. The DISK and RAID devices are implemented without additional caching, thereby forcing a media access on each I/O. We instrument each storage device to collect the workload characteristics and service observations shown in Table 2.

3.2 The storage host

For this study, we use a single storage host (i.e., client). The hardware platform is the same as the SSD and RAID devices shown in Table 1. It is attached to each storage device through a GbE switch. The host connects to each target through a kernel module (provided by the open source). Each device appears as a SCSI device in /dev (e.g., /dev/sdb).

To capture the interactions of the file system and page cache, we mount an ext2 file system over each SCSI device. We limit the host memory to 100MB so as to reduce the amount of read data that can be served from the page cache — without this the devices see write-mostly traffic for all tests.

Characteristic	Units	Acronym
Inter-arrival delay	msecs	ARV
Write requests	percent	WR
Read requests	percent	RD
Write size	KB	WSZ
Read size	KB	RSZ
Randomness	percent	RND
Observation	**Units**	**Acronym**
Request service time	msec	SRV
Throughput	IO/sec	IOPS
Bandwidth	MB/sec	BW
CPU utilization	0 to 1	CPU
Context switch rate	switches/sec	CTXT
Interrupt rate	interrupts/sec	INT
Queue depth	integer	QDEP

Table 2: Workload characteristics and service observations used in this study. All values are averages over a specified time period. These acronyms are used in tables and figures.

3.3 Sample generation

To train and evaluate the storage models, we need sample workloads. We define a workload *sample* as a synthetically generated sequence of I/O with the following parameters:

- percent of requests that are writes (0 to 100)
- percent of requests that are random (0 to 100)
- queue depth (1 to 16)
- think time (0 to 1000 usec)
- request size (1 to 128KB, stepping powers of 2)

We built a workload generator that takes these parameters as input. From the host system we run the generator in a loop to generate many samples, each time passing in a uniformly random value for each parameter. The queue depth represents the number of active threads (workers) in the generator issuing I/O, and the think time is the time each worker sleeps between I/Os.

Each sample warms for 2 seconds and tests for 8 seconds. These values are large enough for the sample to reach a steady state within each storage device. For each sample, we take the best of 3 runs in order to eliminate any transient perturbations in the network or OS. A total of 400 samples are generated per device.

To capture the effects of client side caching, including write aggregation and pre-fetching, we configure the workload generator to issue all I/O through a file stored in a Linux ext2 file system. The maximum footprint (file size) of each sample is limited to 1GB (the capacity of SSD). To also capture workloads without an intervening page cache, 50 of the 400 samples were obtained with no file system by issuing all I/O through a raw SCSI device. As we will describe in Section 6, these additional (non-cached) samples were necessary to train the models to accurately predict the effects of random I/O, as the page cache does a very good job of re-ordering many of the random writes.

4 A case for relative modeling

As discussed in Section 2, an application's I/O workload can change from device to device, even if the application itself does not change. We measure this change by comparing the distributions of workload characteristics, as measured by each of our three devices, for our 400 sample workloads. Not surprisingly, service observations also change across devices.

The data presented in this section makes two important points. First, over a wide range of synthetic workloads, we see a significant change in the workload characteristics seen by each device. Such change must be anticipated by device models if they are to make accurate predictions. Second, device similarity builds a case for predicting scaled, rather than absolute values. That is, rather than predict absolute values of performance, predicting a scaling factor, or relative fitness, may better leverage device similarity for unseen workloads.

For the 400 samples used in this study, Figure 2 shows the cumulative distribution function (CDF) for the workload characteristics and service observations that vary most across the devices. Table 3 contains a summary of all the workload characteristics used in this study, and Table 4 summarizes their differences.

A note on reading the graphs: each CDF shows the percentage of samples that fall at or below a value on the x-axis. For example, about 60% of the samples on our single disk device (DISK) had an average inter-arrival delay of 3 ms or less.

4.1 Workload characteristics

The most interesting differences occur in the workload characteristics as measured by each storage device. Recall that each device was tested with an identical set of synthetic workloads, so the differences are not in the application generating the I/O but rather from interactions between the application and the file system and between the file system and the storage. Although perhaps complex to reason through, such interactions are not random and can therefore be statistically modeled as we will later show in Section 6.

From the shapes of the CDFs, we can draw the following conclusions for our workload samples. First, the request inter-arrival delay decreases with the speed of the storage device. The device with the smallest average delay is SSD, followed by RAID and then DISK. Given a faster device, I/Os complete faster and can therefore be issued more quickly by the application. Of all of our workload characteristics, the arrival rate always shows the greatest change. This result should not be surprising given that our workload generator is operating in a closed manner.

Many of the other workload characteristics also show interesting change, but in a different way. Their changes are due to interactions with the file system and page cache. In particular, we see the average write request size decreasing when device performance increases. Given that page caches aggregate write requests, a slower device allows for more blocks to be aggregated, resulting in slightly larger requests. The slowest device (DISK) has an average write size of 38KB, the RAID 35KB and the SSD 32KB. Although a 16% swing in request size may not appear large, conventional models that make predictions based on request size could see a difference.

Note that these workload differences only arise when we run our tests over a file system with write-back caching. The same tests when run over a synchronously mounted file system or directly over a raw device showed no such variance.

4.2 Service observations

Naturally, the utilization characteristics (CPU utilization, context switches, interrupts, queue depth) are different for each device and depend on the workload characteristics. In other words, a storage device's "level of exertion" depends on the workload and differs across device types. Despite the differences, there are also

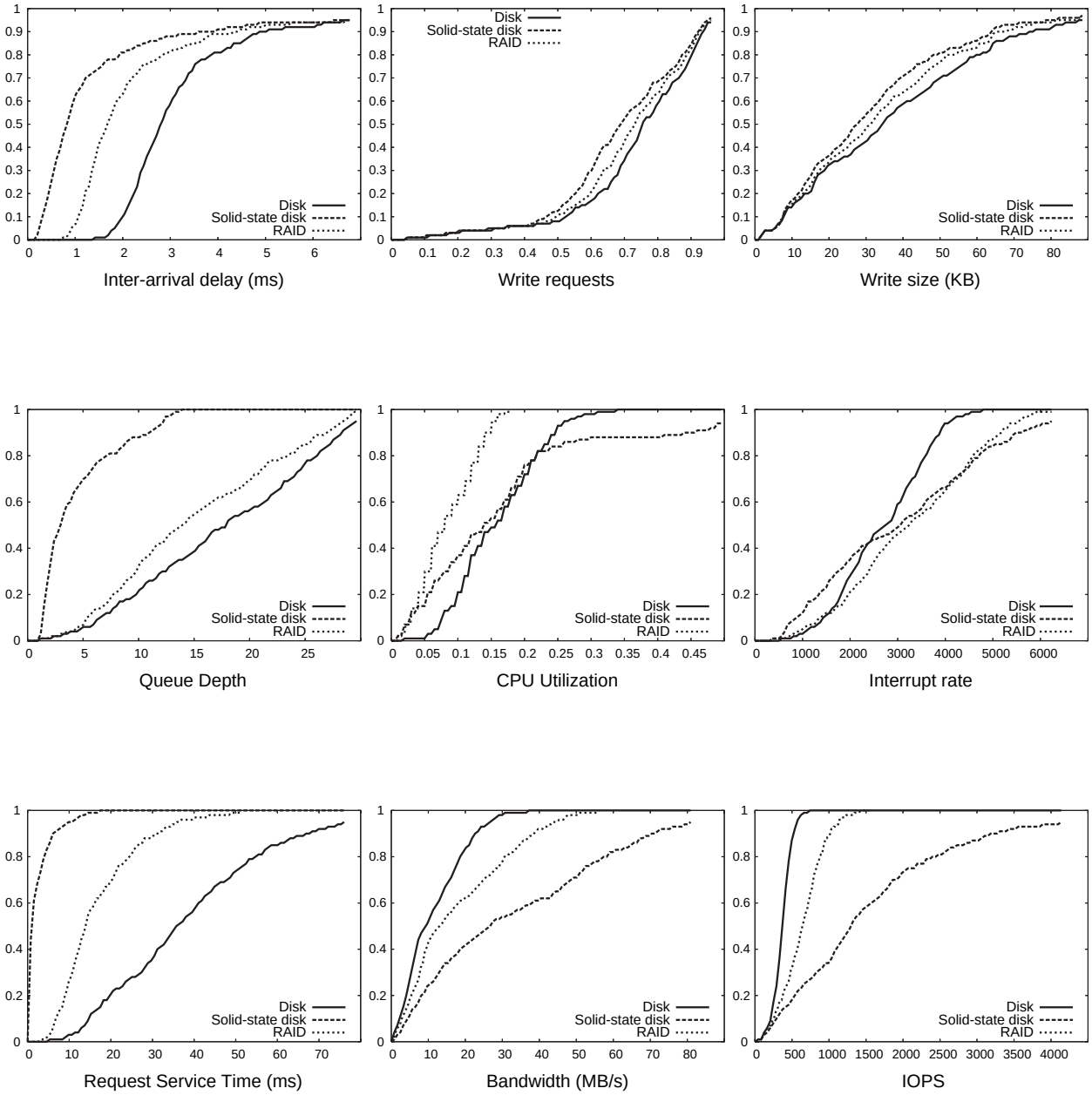


Figure 2: CDFs of selected workload characteristics and service observations.

		COV	Min	Median	90%tile	Max	Mean
DISK	SECS	0.86%	8.08	9.28	10.23	14.51	9.41
	ARV	1.56%	1129.42	2800.75	4878.47	34947.77	3445.08
	SRV	2.08%	2.90	35.85	67.16	90.56	38.62
	CPU	6.68%	0.02	0.16	0.25	0.37	0.16
	BW	1.64%	0.25	9.36	22.67	46.56	11.28
	IOPS	1.58%	31.10	369.78	519.56	923.83	366.82
	WR	0.91%	0.02	0.75	0.94	1.00	0.74
	WSZ	1.50%	1.00	34.26	75.16	246.96	38.99
	RD	3.13%	0.00	0.26	0.49	0.98	0.27
	CTXT	1.85%	525.12	4874.56	6456.30	9591.60	4752.78
	INTR	1.57%	372.55	2745.43	3902.21	5430.66	2683.44
	RND	4.63%	0.00	0.07	0.44	1.00	0.14
	QDEP	1.70%	1.00	17.86	28.17	30.86	17.67
SSD	SECS	0.55%	8.06	8.65	8.89	14.05	8.66
	ARV	2.68%	178.61	824.07	3744.61	33083.40	1702.07
	SRV	4.54%	0.06	0.98	6.15	18.77	2.41
	CPU	5.25%	0.01	0.14	0.44	0.50	0.17
	BW	1.91%	0.46	26.39	70.14	171.93	34.52
	IOPS	2.63%	32.44	1304.69	3215.61	5880.19	1577.24
	WR	0.63%	0.02	0.69	0.92	1.00	0.69
	WSZ	1.42%	1.00	27.41	64.00	244.00	32.90
	RD	2.51%	0.00	0.33	0.54	0.98	0.32
	CTXT	2.10%	511.30	16135.55	39553.81	78807.39	18324.04
	INTR	1.82%	232.73	3049.71	5522.18	7238.46	3101.31
	RND	3.40%	0.00	0.12	0.39	1.01	0.18
	QDEP	2.11%	1.00	2.98	11.10	14.40	4.40
RAID	SECS	0.88%	8.05	8.86	9.31	31.03	8.98
	ARV	3.12%	665.05	1678.51	4431.35	109476.98	2851.32
	SRV	4.46%	1.82	13.87	31.41	52.02	16.85
	CPU	7.04%	0.01	0.08	0.15	0.18	0.09
	BW	2.71%	0.25	12.77	38.44	56.70	17.70
	IOPS	3.09%	32.37	641.64	1011.20	1564.03	638.78
	WR	0.83%	0.02	0.73	0.93	1.00	0.71
	WSZ	1.45%	1.00	30.81	66.67	243.15	35.54
	RD	2.95%	0.00	0.29	0.52	0.98	0.30
	CTXT	2.94%	567.78	7915.73	11958.10	16438.86	7715.50
	INTR	2.60%	316.29	3243.80	5151.26	6927.48	3274.21
	RND	4.87%	0.00	0.10	0.55	1.02	0.18
	QDEP	1.98%	1.00	13.83	26.89	29.97	15.10
SAMPLES: 400							

Table 3: Summary statistics.

Characteristic	COV
Inter-arrival delay	27%
Write requests	7%
Read requests	3%
Write size	7%
Read size	0%
Randomness	12%
Observation	**COV**
Request service time	77%
Throughput	60%
Bandwidth	46%
CPU utilization	27%
Context switch rate	57%
Interrupt rate	8%
Queue depth	46%

Table 4: Average differences in workload characteristics. The coefficient of variation (COV) is the standard deviation divided by the mean.

similarities. For example, DISK and RAID have similar distributions for queue depth, CPU utilization and interrupt rate. Models can learn these similarities.

Also, it is not surprising that the performance characteristics (service request, time, throughput, and bandwidth) differ by device type. As would be expected, SSD has the best performance of the three, followed by RAID and then DISK. As we will show in Section 6, the exact performance differences among these three depend heavily on the workload characteristics of the sample. If it were not for this fact, predicting performance and utilization differences between devices could be accomplished with a single scaling factor between any two pair of devices.

5 Engineering relative fitness models

Relative fitness models are constructed in a similar manner to many conventional black-box models, but there are two fundamental differences: the data on which they train, and the values they train to predict. This section describe the mechanics for building relative fitness models.

5.1 Model selection and training

How workload characteristics and service observations change due to moving between two storage devices can often be determined given enough *training* data. Table 5 shows some actual training data from this study. Each row provides a sample of how, in this case, the throughput (IOPS) changes when a workload moves from RAID to DISK. For example, in the first sample, throughput changes from 122 IO/sec to 115 IO/sec, only a 5% reduction. However, in the second sample we see a 33% reduction.

A trained eye may be able to determine certain correlations by observing such devices over a number of workload samples. For example, the RAID-5 "write-penalty" (i.e., a read-modify-write of parity when writing less than a complete stripe) is well-known and avoided if possible. Such an effect could be seen in training data between, say, a disk and a RAID-5 array. That is, whenever the request size is less than the

RAID workload characteristics						RAID Service Observations					DISK
ARV	WR	WSZ	RD	RSZ	RND	QDEP	CPU	SRV	DBW	IOPS	IOPS
8.2	0.92	5.3	0.07	4.10	0.00	8.43	0.01	7.8	0.62	122	115 (.95)
1.4	0.87	8.2	0.13	4.74	0.02	10.52	0.06	8.9	5.41	718	484 (.67)
3.9	0.91	8.1	0.09	4.26	0.01	9.79	0.02	7.7	1.95	257	247 (.96)

Table 5: The actual training data for a RAID-to-DISK model (CTXT and INT columns omitted due to lace of space). This table shows 3 of the 200 samples used in training an IOPS model for DISK, given the workload characteristics and service observations on RAID.

stripe size of the array, a performance hit is taken relative to the disk. Although many of these effects can be reasoned through by experts, manually keeping track of the numerous correlations between workload and device type, and the service observations they produce, is not realistic. For this, one can use a statistical model. We choose a classification and regression tree (CART) [7] for its ability to effectively model a storage device, and its differences relative to other devices, as a *black-box* (i.e., requiring no knowledge of device internals).

CART models are often used when predicting a continuous *dependent* variable (e.g., IOPS), based on a set of continuous *predictor* variables (e.g., request size and arrival rate). In the case of moving a workload from device $D1$ to $D2$, the predictor variables are the workload characteristics and service observations on $D1$ and the dependent variable is a workload characteristic or service observation on $D2$. That is, for each dependent variable one wishes to predict, one trains a separate CART model. For this study, we want to predict averages for request service time, bandwidth, throughput, and CPU utilization.

The first step in training a CART model is obtaining the training data. For this, we use the same workload samples discussed in Section 3, but only half of them. The other half is the *testing* data we use for evaluating the models in Section 6.

A CART-building algorithm attempts to linearly separate the training samples and, in doing so, builds a regression tree (i.e., the CART model). Each leaf node in the regression tree contains samples with a similar value for the dependent variable, but also with similar values for the predictor variables. The goal in building a regression tree is to determine which of the predictor variables provide the most *information gain* with respect to predicting the dependent variable. In effect, the regression tree is a sequence of if-then-else questions, starting at the root, and terminating at a leaf node, as shown in Figure 1.

Information gain naturally varies across device types. For example, the burstiness of write requests would be more useful when predicting performance changes between two disks, one with write caching and one without, than between two disks with ample caching for absorbing bursts. Intuitively, the characteristics and service observations with the most information gain are those that best explain why two observations are different.

5.2 Absolute relative fitness (ARF)

Figure 3 shows an actual regression tree obtained from the entire set of training samples from which Table 5 was taken. While training, we limited the tree to a depth of 3 so we could show the complete tree and also to illustrate that the inter-arrival delay (ARVL), service time (SRVC), and percentage of reads (RD) have the most information gain for this particular training data. In other words, of all the workload characteristics and service observations shown in Table 5, these are the best three to look at when predicting changes in throughput between these two devices. Unpruned, the tree has a depth of 12 and, oddly enough, found the throughput (IOPS) of RAID to have the least amount of information when prediction the throughput of DISK, and therefore omitted it from the tree.

```
if (ARVL < 2143.33) then
   if (ARVL < 1271.65) then
      if (ARVL < 975.44)
      then IOPS = 554.916
      else IOPS = 459.059
   else
      if (SRVC < 14377.6)
      then IOPS = 408.165
      else IOPS = 347.357
else
   if (ARVL < 4244.6) then
      if (ARVL < 2685.86)
      then IOPS = 301.745
      else IOPS = 238.028
   else
      if (RD < 0.035)
      then IOPS = 63.88
      else IOPS = 157.394
```

Figure 3: A pruned regression tree used to predict the throughput (IOPS) of DISK given the workload characteristics and service observations on RAID. In this example, the read percent (RD), arrival rate (ARVL) and service time (SRVC) on RAID were found to have the most information gain when predicting the throughput of DISK for the same workload.

This modeling technique we refer to as *absolute relative fitness* (ARF). That is, given the workload characteristics and service observations on $D1$, we build a model to predict a service observation on $D2$.

5.3 Scaled relative fitness (SRF)

Rather than predicting an absolute value, however, one can train the model to instead predict a *scaling factor* between $D1$ and $D2$. In doing so, one can better capture the similarity between devices. For example, the best predictor of IOPS on DISK is the IOPS on RAID, multiplied by some constant. This constant is our scaling factor, or relative fitness value. However, the relative fitness depends on workload characteristics and other service observations, so we again build a regression tree. The training data is identical to that for absolute relative fitness. The only difference is that we train using the relative fitness values shown in parenthesis in the last column of Table 5. For example, the first sample shows a relative fitness value of $115.3/122.0 = .95$. Figure 4 shows the actual SRF model pruned to a depth of 2.

In this case, RAID's IOPS and QDEP are found to have the most information when linearly separating the relative fitness values in our training data. Similarly to the ARF model, an unpruned SRF model also grew to a depth of 12 and found useful information in each of the other characteristics and service observations.

5.4 Discussion

Absolute and scaled relative fitness differ from the conventional modeling approach in fundamental ways. First, the workload characteristics are specified relative to a specific device $D1$. Second, we introduce

```
if (IOPS < 138.015) then
   if (QDEP < 2.205)
   then RF = 269.72
   else RF = 113.65
else
   if (IOPS < 657.95)
   then RF = 71.82
   else RF = 51.81
```

Figure 4: A pruned regression tree used to predict the relative fitness (RF) of DISK. In this example, the throughput (IOPS) and queue depth (QDEP) on RAID were found to have the most information gain when predicting the relative fitness of DISK for the same workload. The RF value is a scaling factor for DISK throughput (IOPS) relative to that observed on RAID.

service observations on $D1$ as additional input into the model. Third, in the case of scaled relative fitness, we predict a scaling factor rather than an absolute value. Fourth, relative fitness models may not be symmetric, so we distinguish direction (i.e., moving a workload from $D1$ to $D2$ will use a different model than from $D2$ to $D1$).

6 Evaluation

This section evaluates the efficacy of relative fitness models for predicting three performance characteristics: request service time (SRVC), throughput (IOPS) and bandwidth (BW); and one utilization characteristic: CPU utilization.

Summarizing from previous sections, our hypotheses are as follows:

- **Hypothesis 1**: Models that do not account for changes in workload characteristics when moving a workload between devices will give inaccurate predictions unless the devices have similar performance.
- **Hypothesis 2**: One can capture changes in workload characteristics by training a model using *pairs* of devices. Building such models will more accurately predict the absolute performance or resource utilization of a workload when running on a given device. Moreover, service observations add additional useful information.
- **Hypothesis 3**: Because devices have similar characteristics, assuming similarity by default and predicting a scale factor will be beneficial for unseen workloads.

This section compares conventional modeling to absolute and scaled relative fitness. To test hypothesis 1, we use only conventional modeling. To test hypothesis 2, we add absolute relative fitness. To test hypothesis 3, we further add scaled relative fitness. All three modeling techniques are evaluated on their ability to accurately predict changes in performance and resource utilization across each of our devices, for 200 of the sample workloads discussed in Section 3. The other 200 are used to train the models. As a final test, we evaluate the models using the Postmark benchmark [5].

In total, we evaluate 12 conventional models, one for each prediction (SRVC, BW, IOPS, CPU) on each device (DISK, SSD, RAID). For each of absolute and scaled relative fitness, we evaluate 24 models, one for each prediction and (direction-specific) device pair. For each prediction, the difference between the

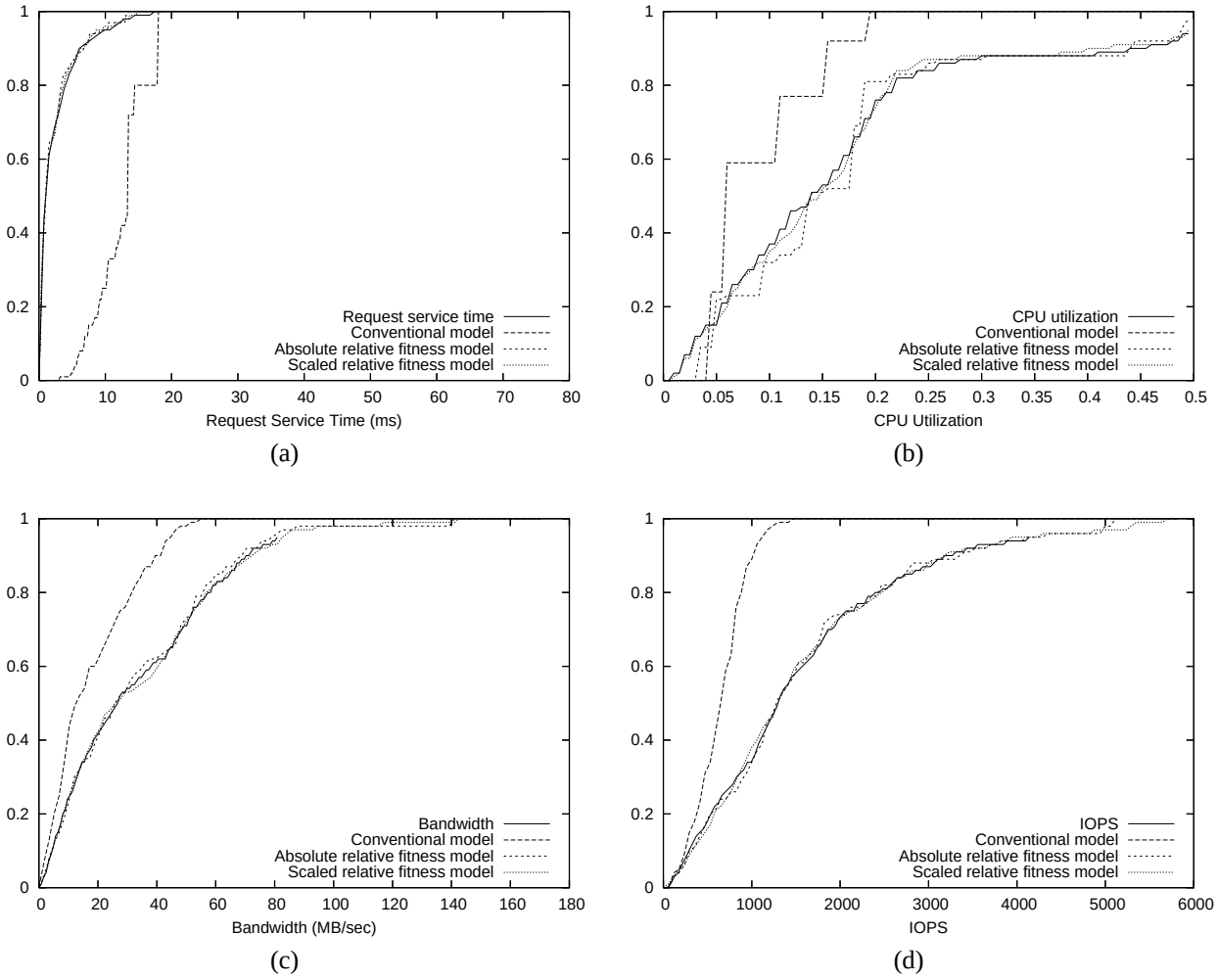


Figure 5: Predicted CDFs of SRVC (a), CPU (b), BW (c), and IOPS (d) when moving a workload from RAID to SSD.

expected and actual value is used to calculate the relative error, $\frac{actual-expected}{actual}$, and we report the median relative error [1] across all samples as a metric of the model's "goodness."

6.1 Results

This subsection defends each hypothesis on the basis of which models yield the lowest median relative error. To illustrate prediction accuracy, Figure 5 shows the predicted and actual CDFs when moving our 200 test workloads between RAID and SSD. Due to lack of space, we cannot show the CDFs for all pairs of devices. Table 6 summarizes the prediction accuracies between all pairs of devices, Table 7 summarizes the prediction for each service observation (SRVC, CPU, BW, and IOPS), and Table 8 provides an overall summary.

[1] The trends are the same when reporting the mean.

CM	DISK	SSD	RAID
DISK	0.12	0.58	0.31
SSD	0.35	0.18	0.30
RAID	0.29	0.43	0.14
ARF	DISK	SSD	RAID
DISK	0.04	0.26	0.12
SSD	0.15	0.05	0.17
RAID	0.10	0.22	0.06
SRF	DISK	SSD	RAID
DISK	0.00	0.24	0.10
SSD	0.14	0.00	0.14
RAID	0.08	0.20	0.00

Table 6: Summary of predictions between pairs of devices

Hypothesis 1 - Conventional Modeling

A quick glance at the predicted CDFs in Figure 5 supports hypothesis 1. The conventional modeling technique provides, at best, a rough approximation of the actual distributions (the darkest line) when workloads are moved from RAID to SSD. For each of the four predicted CDFs, the line furthest from the darkest line is that predicted by the conventional model. Overall, the average error for all predictions from RAID to SSD is 43%, and the average error between all pairs of devices for the conventional model is 38%.

These low prediction accuracies can easily be explained upon closer inspection of the CART models used to make the predictions. The workload characteristics with the most information (i.e., nearest the top of the regression tree) are often the most likely to change between devices. For workloads moving between RAID and SSD, these characteristics are the inter-arrival delay (ARVL), write size (WRSZ), and randomness (RND). Although the workload randomness, on average, does not change between RAID and SSD (i.e., both see 18%), the inter-arrival delay changes by 40%, the write size by 7%, and the percent of write requests by 3%. Consequently, using workload characteristics as measured by RAID as input into the models for SSD often results in incorrect paths being taken in the regression trees.

Yet, when the workload characteristics come from the same device for which the prediction is being made, the conventional model works well with an average error of 15%, as shown by the diagonal for the conventional model (CM) in Table 6 (i.e., DISK to DISK, SSD to SSD, or RAID to RAID). These results are consistent with prediction accuracies seen in other black-box modeling approaches [20].

Hypothesis 2 - Absolute Relative Fitness

The CDFs predicted by the models of absolute relative fitness track and, in many cases, are indistinguishable from, the actual CDFs. Overall, the average error between all pairs of devices is 17%.

These results support hypothesis 2 in three ways. First, modeling a device relative to the characteristics as measured by another device reduces the misprediction due to changing workloads. Second, for our devices, there is enough information in the workload characteristics of $D1$ to build an accurate device model of $D2$. (Note that this does not always necessarily have to be true. In an early experiment we ran, a programming bug in one of the devices resulted in only sector-sized requests being sent to the device. As such, the request size as measured by that device was always 512 bytes (i.e., zero information). Any device model paired up with this buggy device would learn very little from knowing that the request size is 512.)

	Average	WC	SO	WC+SO
CM	0.72	0.33	-	-
ARF	0.72	0.26	0.21	0.19
SRF	0.52	0.22	0.21	0.20

(a) SRVC

	Average	WC	SO	WC+SO
CM	0.44	0.32	-	-
ARF	0.44	0.20	0.21	0.20
SRF	0.45	0.17	0.16	0.15

(b) CPU

	Average	WC	SO	WC+SO
CM	0.58	0.29	-	-
ARF	0.58	0.18	0.16	0.15
SRF	0.31	0.12	0.13	0.11

(c) BW

	Average	WC	SO	WC+SO
CM	0.58	0.29	-	-
ARF	0.58	0.18	0.16	0.15
SRF	0.31	0.12	0.13	0.11

(d) IOPS

Table 7: Summary of predictions for request service time (a), CPU utilization (b), bandwidth (c), and throughput (d). We compare the conventional model (CM) using workload characteristics (WC) with absolute relative fitness (ARF) and scaled relative fitness (SRF). For ARF and SRF, we show results when using only workload characteristics, only service observations (SO), and both together.

	Average	WC	SO	WC+SO
CM	0.51	0.38	-	-
ARF	0.51	0.20	0.19	0.17
SRF	0.41	0.16	0.17	0.15

Table 8: Overall Summary of predictions. We compare the conventional model (CM) using workload characteristics (WC) with absolute relative fitness (ARF) and scaled relative fitness (SRF). For ARF and SRF, we show results when using only workload characteristics, only service observations (SO), and both together.

Third, upon closer inspection of the regression trees, we see that service observations provide slightly more information than the workload characteristics.

To directly test this last point, we construct a model using only service observations and find that the prediction accuracy across all device pairs and predictions is 19%, compared to 20% when only using workload characteristics. With both together, we see the 17%. These differences are captured in Table 8.

Compared to conventional modeling, absolute relative fitness reduces prediction error by 53% and supports our second hypothesis.

Hypothesis 3 - Scaled Relative Fitness

The CDFs predicted by our models of scaled relative fitness are, in all cases, nearly indistinguishable from the actual CDFs. The average error across all device pairs and predictions is 15%. This is a 12% error reduction when compared to absolute relative fitness and a 61% reduction when compared to conventional modeling.

Scaled relative fitness models assume a default similarity between devices, and therefore naturally interpolate for unseen workloads. When the number of training samples is comprehensive, reducing the need for interpolation, we find that absolute and scaled relative fitness perform similarly. Early tests (not presented here) were conducted with fewer training samples. In such cases, the differences between scaled and absolute relative fitness were more pronounced.

The ability to measure device similarity is a direct side-effect of our scaled models. Notice the diagonal of zeros for SRF in Table 6. Given the training data, the SRF model for each pair of like devices was a single node tree with a value of 1.0. In other words, the tree easily captured the fact that any workload running on $D1$ will change by 0% if moved to $D1$, so the relative fitness value is 1.0.

We can therefore use relative fitness to determine which devices are *similar*, and which ones are not, with respect to performance and resource utilization. Looking again at Table 6, we see that SRF prediction errors are much lower between DISK and RAID than they are between DISK and SSD, or between RAID and SSD, suggesting that DISK is more similar to RAID, and this is indeed the case.

With respect to the information for workload characteristic vs. service observations (Table 8), unlike absolute relative fitness, we see a slight increase in prediction error (1%) when building SRF trees that only use service observations. The most effective trees were those that used both workload characteristics and service observations.

6.2 A macro benchmark

Our final test evaluates the trained models on a workload for which there was no explicit training: the Postmark benchmark [5]. Postmark was built to simulate Internet and small file workloads. The first of three phases creates a large pool of files. The second phase performs transactions on these files (create, delete, read, or write). The third phase deletes them all. Postmark tests the small-file performance of a file system, especially its ability to efficiently handle short-lived files and their associated metadata operations (e.g., adding/removing names from a directory, and creating/deleting inodes). When the Postmark working set (i.e., the number of files) exceeds the size of the file system page cache, each Postmark transaction will, on average, result in multiple trips to the storage device. Because the files are being selected at random, the resulting I/O at the device is relatively random. In this case, the latency of the storage device determines the maximum throughput of Postmark. We configure Postmark with 20,000 files and 20,000 transactions. The resulting footprint is approximately 100MB (5KB average file size $\times$ 20,000 files) and exceeds the page cache memory of the host on which Postmark is running.

For this test we select SSD and DISK, our storage devices with the largest disparity in average request latency. On SSD, Postmark completes in 21 seconds with an average device throughput of 2052 IO/sec. On

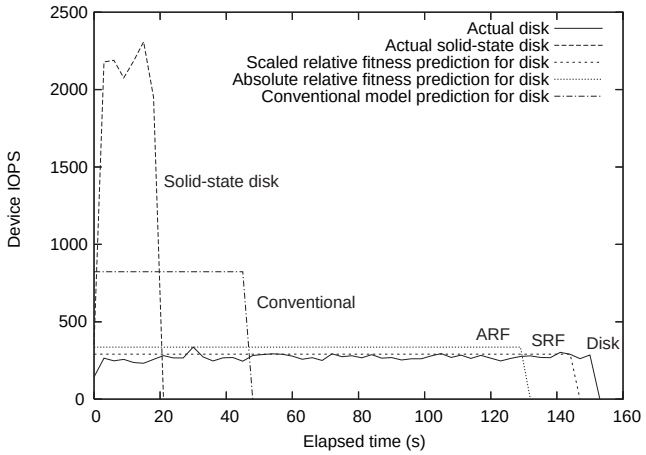


Figure 6: We compare the ability for CM, ARF and SRF to predict the change in device throughput when moving a Postmark workload from SSD to DISK. Both ARF and SRF use service observations and workload characteristics. CM only uses workload characteristics. Due to a change in throughput, SSD completes the Postmark test in 21 seconds, and DISK takes 156 secs. CM predicts this within 205%, ARF within 25%, SRF within 8%.

DISK it takes 156 seconds with an average throughput of 269 IO/sec. The challenge to our models is to make this prediction.

Figure 6 shows our results. Each line plots I/O throughput as a function of time. There is one line for SSD (the tallest line), one for DISK (the shortest line), and one for each prediction for DISK (CM, ARF, and SRF). CM predicts with an average error of 205%, ARF 25%, and SRF 8%.

Just as in the training/testing workloads, the workload characteristics experienced significant change. On SSD the arrival rate was approximately one I/O every 487 usec. On DISK, it was one every 3.7 ms. Similarly, average request service time changed from 600 usec to 24 ms, queue depth changed from 2 to 8, and the randomness, originally 65%, became 80%. Given this, it is no surprise that the conventional model mispredicted as badly as it did. As for absolute and scaled relative fitness, we see a 17% improvement when predicting a scaled rather than absolute value, a result consistent with hypothesis 3.

Interestingly, the first time we ran Postmark, we noticed that the device randomness (65%-80%) was, on average, much higher than the maximum being generated by our micro-benchmarks on the devices (about 30%). Upon closer inspection, we realized that the randomness from the micro-benchmarks, although specified as high as 100% in the workload generator, was being ordered by the OS before going to disk. Recall that the workload generator issues all I/O to the same file, so writes are delayed and optimized by elevator scheduling algorithms within the OS. Postmark, however, did not benefit as much from such out-of-order scheduling due to the large amounts of file system metadata, leading to much more random I/O on the devices. The result was that the models were not trained with a sufficient number of random samples to predict Postmark performance accurately. As such, the models did not distinguish between, say, a 40% random workload and one that was 90% (i.e., both were greater than 30% and mapped into the same leaf of the regression tree), and all three models predicted poorly.

To account for this, we trained each model with an additional 50 workload samples that performed all I/O through a *raw* SCSI device. Raw devices do not use the page cache and are therefore ineligible for elevator scheduling or request coalescing within the OS. Although the additional training data did not benefit CM, ARF and SRF learned from the new samples, resulting in the Figure 6 predictions.

6.3 Summary

The experiments presented support our hypotheses with respect to relative device modeling. First, the effect of a workload changing between two devices $D1$ and $D2$ can be reduced by training a device model of $D1$ using the workload characteristics as measured by $D2$. Second, the service observations on $D2$ can provide just as much information as the workload characteristics when building a model. Third, predicting scale factors best captures the similarity between devices, and can be used to identify like devices in a data center.

7 Conclusion

Relative device modeling is a promising new approach to predicting workload performance and resource utilization. By accounting for the device-dependency of workloads and enabling the use of service observations, relative device modeling significantly increases model accuracy. Experiments show that relative device modeling reduces prediction error by over 60% on average when compared to conventional modeling techniques.

Acknowledgements

We thank the members and companies of the PDL Consortium (including APC, EMC, Equallogic, Hewlett-Packard, Hitachi, IBM, Intel, Microsoft, Network Appliance, Oracle, Panasas, Seagate, and Sun) for their interest, insights, feedback, and support. We also thank Intel, IBM, and Seagate for hardware donations that enabled this work. This material is based on research sponsored in part by the National Science Foundation, via grant #CNS-0326453.

References

[1] Guillermo A. Alvarez, John Wilkes, Elizabeth Borowsky, Susie Go, Theodore H. Romer, Ralph Becker-Szendy, Richard Golding, Arif Merchant, Mirjana Spasojevic, and Alistair Veitch. Minerva: an automated resource provisioning tool for large-scale storage systems. *ACM Transactions on Computer Systems*, **19**(4):483–518. ACM, November 2001.

[2] Eric Anderson. *Simple table-based modeling of storage devices*. SSP Technical Report HPL–SSP–2001–4. HP Laboratories, July 2001.

[3] Eric Anderson, Michael Hobbs, Kimberly Keeton, Susan Spence, Mustafa Uysal, and Alistair Veitch. Hippodrome: running circles around storage administration. *Conference on File and Storage Technologies* (Monterey, CA, 28–30 January 2002), pages 175–188. USENIX Association, 2002.

[4] Eric Anderson, Mahesh Kallahalla, Susan Spence, Ram Swaminathan, and Qian Wang. *Ergastulum: an approach to solving the workload and device configuration problem*. Technical report HPL–SSP–2001–05. HP Labs, 2001.

[5] Network Appliance. PostMark: A New File System Benchmark. http://www.netapp.com.

[6] E. Borowsky, R. Golding, A. Merchant, L. Schreier, E. Shriver, M. Spasojevic, and J. Wilkes. Using attribute-managed storage to achieve QoS. *International Workshop on Quality of Service* (Pittsburgh, PA, 21–23 March 1997). IFIP, 1997.

[7] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*. Chapman and Hall/CRC, 1998.

[8] Douglas J. Futuyma. *Evolutionary Biology. Third edition*. SUNY, Stony Brook. Sinauer. December 1998.

[9] Gregory R. Ganger. Generating representative synthetic workloads: an unsolved problem. *International Conference on Management and Performance Evaluation of Computer Systems* (Nashville, TN), pages 1263–1269, 1995.

[10] Gregory R. Ganger and Yale N. Patt. Using system-level models to evaluate I/O subsystem designs. *IEEE Transactions on Computers*, **47**(6):667–678, June 1998.

[11] Steven D. Gribble, Gurmeet Singh Manku, Drew Roselli, Eric A. Brewer, Timothy J. Gibson, and Ethan L. Miller. Self-similarity in file systems. *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (Madison, WI, 21–26 June 1998). Published as *ACM SIGMETRICS Performance Evaluation Review*, **26**(1):141–150. ACM Press, 1998.

[12] Intel. iSCSI. http://www.sourceforge.net/projects/intel-iscsi.

[13] Zachary Kurmas and Kimberly Keeton. Using the distiller to direct the development of self-configuration software. *International Conference on Autonomic Computing* (New York, NY, 17–18 May 2004), pages 172–179. IEEE, 2004.

[14] Zachary Kurmas, Kimberly Keeton, and Kenneth Mackenzie. Synthesizing Representative I/O Workloads Using Iterative Distillation. *International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems* (Orlando, FL, 12–15 October 2003). IEEE/ACM, 2003.

[15] Julian Satran. iSCSI. http://www.ietf.org/rfc/rfc3720.txt.

[16] Elizabeth Shriver, Arif Merchant, and John Wilkes. An analytic behavior model for disk drives with readahead caches and request reordering. *International Conference on Measurement and Modeling of Computer Systems* (Madison, WI., 22-26 June 1998). Published as *Perform. Eval. Rev.*, **26**(1):182–191. ACM, June 1998.

[17] Mustafa Uysal, Guillermo A. Alvarez, and Arif Merchant. A modular, analytical throughput model for modern disk arrays. *International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems* (Cincinnati, OH, 15–18 August 2001), pages 183–192. IEEE, 2001.

[18] Elizabeth Varki, Arif Merchant, Jianzhang Xu, and Xiaozhou Qiu. Issues and challenges in the performance analysis of real disk arrays. *Transactions on Parallel and Distributed Systems*, **15**(6):559–574. IEEE, June 2004.

[19] Mengzhi Wang, Anastassia Ailamaki, and Christos Faloutsos. Capturing the Spatio-Temporal Behavior of Real Traffic Data. *IFIP WG 7.3 Symposium on Computer Performance* (Rome, Italy, September 2002), 2002.

[20] Mengzhi Wang, Kinman Au,, Anastassia Ailamaki, Anthony Brockwell, Christos Faloutsos, and Gregory R. Ganger. Storage Device Performance Prediction with CART Models. *International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems* (Volendam, The Netherlands, 05–07 October 2004). IEEE/ACM, 2004.