Chapter 5

Predicted Performance

5.1. Introduction

The self-scaling benchmark of Chapter 4 automatically scales its workload according to the capability of the system being measured. This automatic scaling keeps the benchmark relevant over a wide range of machines. By graphing the workloads around a few focal points, the benchmark also provides information on what workloads perform well for a system. There are, however, several shortcomings of the self-scaling benchmark. First, by self-scaling, the benchmark complicates the task of comparing results from two systems. This complication arises because the self-scaling benchmark will likely choose different workloads by which to measure each system. Next, though the performance graphs apply to a wider range of applications than do today's I/O benchmarks, they stop short of applying to all workloads. In this chapter, I show how *predicted performance* solves these problems by using the results from the self-scaling benchmark to accurately estimate the I/O performance for arbitrary workloads, that is, within 10-15% of actually measuring the workload.

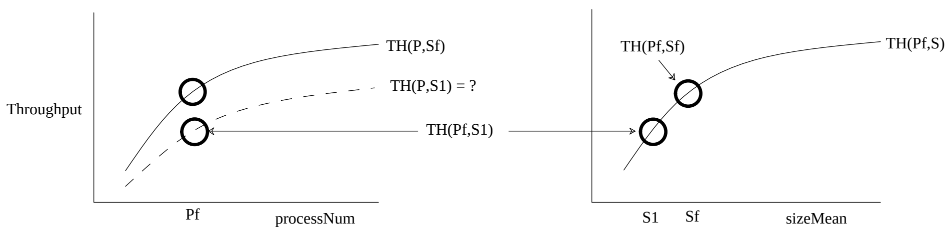
This approach is similar in concept to work done by Saavedra-Barrera who predicts CPU performance by measuring the performance for a small set of primitive FORTRAN operations [Saavedra-Barrera89]. The main difference between the performance of FORTRAN programs and I/O workloads is the way performance depends on operations and arguments. A FORTRAN operation, such as adding two floating-point numbers, takes the same amount of time independent of the values of the arguments. To predict the performance for an arbitrary FORTRAN workload (program), only the performance and frequency for each operation in the program need be known. In Saavedra-Barrera's research, the main difficulty was to choose a complete set of primitive FORTRAN operations.

In contrast to FORTRAN programs, I/O has only two operations, read and write, and performance depends heavily on the arguments (sizeMean, uniqueBytes, seqFrac, and processNum). No short list of performances for each operation is possible because such a list would be for each unique combination of uniqueBytes, sizeMean, readFrac, processNum, and seqFrac. Since it is clearly impractical to measure or list all combinations, I develop a method for using the summary performance graphs of Chapter 4 to estimate performance for any combination of workload parameters.

5.2. Prediction Models

A straightforward approach to estimating performance for all possible workloads takes a fine-grained, orthogonal sampling of workloads. Performance of arbitrary workloads within this orthogonal mesh could then be estimated by interpolating in multiple dimensions between neighboring points. The main disadvantage is the time it takes to gather the orthogonal sample—the number of points in this sample increases exponentially with the number of parameters in the workload. Ten sample points for each of the five workload parameters require 100,000 measurements, or two years on the SPARCstation 1+ of Table 2.1 (page 20)! In addition, Section 5.4 will show how, given an equal number of measured workloads, interpolation using orthogonal sampling gives poorer performance prediction than this chapter's proposed method. Instead of interpolation using orthogonal sampling, I take a different approach by using the performance behavior given in the self-scaling benchmark graphs to estimate performance for unmeasured workloads.

Figure 5.1 shows the general approach to estimating performance for unmeasured workloads applied to a two-parameter workload. Intuitively, the simplification that I investigate in



(a) (b)

Figure 5.1: Predicting performance of unmeasured workloads. This figure shows how to predict performance with a workload of two parameters, processNum and sizeMean. Intuitively, I assume that the *shape* of a performance curve for one parameter is *independent* of the other parameters' values. More specifically, I assume that the *ratio* of performance between two plots against a given parameter (such as sizeMean) is constant. The solid lines represent workloads that have been measured; the dashed line represents workloads that are being predicted. The solid line in Figure 5.1a shows throughput graphed against processNum with sizeMean fixed at $sizeMean_f$. Figure 5.1b shows throughput versus sizeMean with processNum fixed at $processNum_f$. I predict the throughput curve versus processNum in Figure 5.1a with sizeMean fixed at $sizeMean_1$ by assuming that $\frac{Throughput(processNum, sizeMean_f)}{Throughput(processNum, sizeMean_1)}$ is constant (independent of processNum) and fixed at $\frac{Throughput(processNum_f, sizeMean_f)}{Throughput(processNum_f, sizeMean_1)}$. For example, if $Throughput(processNum_f, sizeMean_f)$ is 6 and $Throughput(processNum_f, sizeMean_1)$ is 3, then I estimate $Throughput(processNum, sizeMean_1)$ (the dashed line in Figure 5.1a) as $\frac{3}{6} \times Throughput(processNum, sizeMean_f)$.

this chapter is the *independence* of the *shape* of one parameter's performance curve from the values of the other parameters. In the next section, I demonstrate that this simplification introduces only a small error. In the self-scaling evaluation, I measure workloads with all but one parameter fixed at a focal point. In Figure 5.1, these are shown as the solid-line throughput curves $Throughput(processNum, sizeMean_f)$ and $Throughput(processNum_f, sizeMean)$, where $processNum_f$ is processNum's focal point and $sizeMean_f$ is sizeMean's focal point. Using these measured workloads, I estimate performance for unmeasured workloads $Throughput(processNum, sizeMean_1)$, where $sizeMean_1 \neq sizeMean_f$, shown as the dashed line in Figure 5.1a. I assume a *constant ratio* between $Throughput(processNum, sizeMean_f)$ and $Throughput(processNum, sizeMean_1)$. This ratio is known at $processNum = processNum_f$ to be $\frac{Throughput(processNum_f, sizeMean_f)}{Throughput(processNum_f, sizeMean_1)}$. My assumption that the performance shapes are independent of other parameter values leads to an overall performance equation of

$$Throughput(X, Y, Z...) = f_X(X) \times f_Y(Y) \times f_Z(Z) \cdots$$

where X, Y, Z, ... are the parameters and $f_X(X), f_Y(Y)$, and $f_Z(Z)$ are the graphs for each parameter (Appendix C proves that if the performance shape for each parameter is independent of the other parameter values, the overall performance equation must take this form).

The above section describes how to estimate performance from a family of performance graphs (Appendix B gives a pseudo-code version). Each performance region has its own set of graphs, as in Figures 4.8 and 4.9 on pages 67 and 68. In estimating performance for a workload, I use the graphs from whichever performance region that workload falls in.

5.3. Verification of Prediction Model

In this section, I examine how accurately the equation described above approximates actual performance for the systems measured by the self-scaling benchmark. Ideally, I would like to measure large, I/O-intensive applications and compare their performance to those

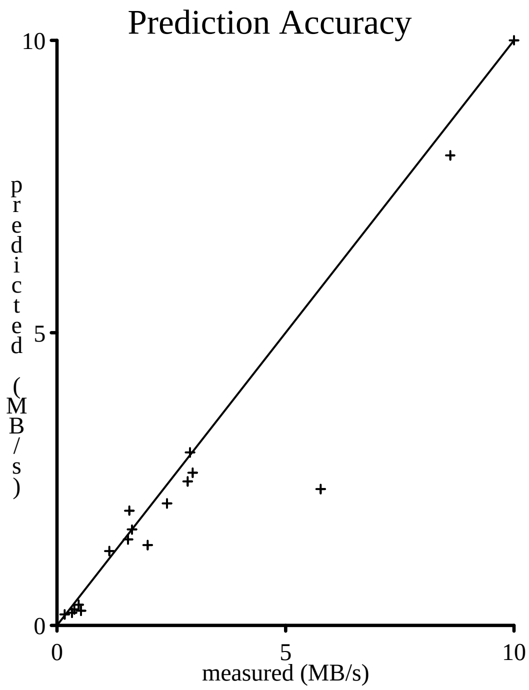


Figure 5.2: Predicting performance of four traditional benchmarks. This figure graphs the predicted performance against the actual (measured) performance for four traditional benchmarks (Andrew, Bonnie, IOStone, and Sdet) for four systems (SPARCstation 1+, DECstation 5000/200, HP 730, and Convex C240).

predicted using the data collected in the self-scaling benchmarks. Unfortunately, no such public domain applications are available because, as Chapter 2 showed, even well known systems benchmarks are CPU-bound. To assess accuracy given the lack of benchmarks I take two tacks:

(1) First, I measure the I/O portion of four I/O benchmarks and compare them to the predicted performance using the self-scaling data for the values of the parameters from the measurement of that I/O benchmark. Figure 5.2 shows a scatter plot of predicted versus measured performance for Andrew, Bonnie, IOStone, and Sdet on four systems: a SPARCstation 1+, a DECstation 5000/200, an HP 730, and a Convex C240. With perfect prediction, predicted performance would exactly equal measured performance, and all points would lie on the 45 degree line. Figure 5.2 shows that this is approximately the case— out of 16 points, all but one are predicted accurately; only Sdet run-

ning on the Convex C240 is substantially in error. These results are typical of the prediction model's accuracy: performance of most workloads is predicted accurately; a few workloads, particularly those far away from the focal point, have significant error. In this experiment, I measured prediction for four workloads designed by independent groups to make sure that prediction is accurate for valued workloads. But there is a danger that these benchmarks do not reflect a wide enough range of current or future workloads.

(2) So, to increase my confidence that the model can predict performance over a wider range of workloads, I measured the performance of 100 random I/O workloads and compare these measurements to their predicted performance. The 100 workloads are randomly selected over the entire space of workloads derived for each system in the self-scaling benchmark. Each workload's predicted performance versus actual, measured performance is then plotted on a scatter plot, as in Figure 5.3. By verifying prediction accuracy over 100 workloads, I became more certain that the average prediction error over these workloads is indicative of the prediction error over the entire workload space. Figure 5.3c shows how the confidence interval from measuring average prediction error tightens with larger numbers of random workloads. By the time I measure 100 workloads, the 90% confidence interval is only a few percent.

5.3.1. SPARCstation 1+

Figure 5.3a shows how accurately the *product of single parameter functions* approach predicts performance for Andrew, Bonnie, IOStone, and Sdet. For each of these benchmarks, the model is able to accurately predict performance. In addition, over the wide range of performances (0.2 MB/s to 3.0 MB/s) resulting from 100 random workloads, the predicted performance values match extremely well to the measured results. Half of all workloads, for example, have a prediction error of 10% or less; three-quarters of all workloads had a prediction error

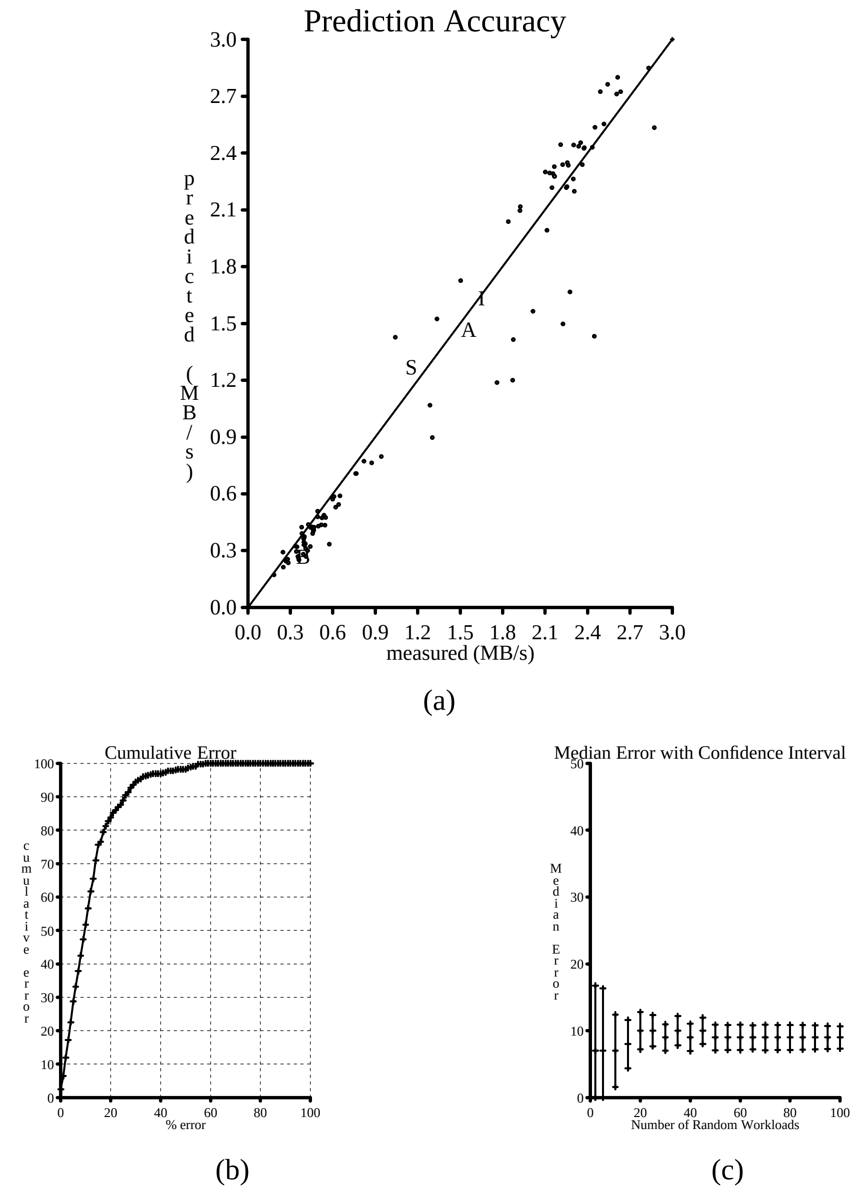


Figure 5.3: Evaluation of prediction accuracy for SPARCstation 1+ with one disk. This figure graphs the predicted performance against the actual (measured) performance for the SPARCstation in Figure 4.7. Points representing traditional benchmarks are shown as the letters ‘‘A’’, ‘‘B’’, ‘‘I’’, and ‘‘S’’, representing Andrew, Bonnie, IOStone, and Sdet, respectively. Graph (a) shows how the prediction model accurately predicts performance for these four traditional benchmarks. In addition to predicting performance for a few benchmarks, the model is able to do so for 100 random workloads. Each point represents a single workload, with each parameter value randomly chosen from its entire range shown in Figure 4.7. The closer the points lie to the solid line, the better the prediction accuracy. Median error for graph (a) is 10%; the complete cumulative error distribution is given in graph (b). Performance for each workload ranges from 0.2 MB/s to 3.0 MB/s. Clearly single-point benchmarks cannot adequately capture the large range of performances generated by diverse workloads. Graph (c) gives the 90% confidence interval of the median error, that is, it displays how certain I am that the median error is really 10%. Using 100 workloads, I am 90% sure that the true median error is between 7.5% and 10.7%.

of 15% or less. Figure 5.3b shows the cumulative error distribution of the prediction. In contrast, any single-point I/O benchmark would lead one to believe that all workloads yielded the same performance. For example, both Andrew's and IOStone's workloads yield performances of 1.25 MB/s, leading to a median prediction error of 50%. Bonnie's sequential block write yields a performance of .32 MB/s, for a median prediction error of 65%. Table A.1 in Appendix A gives the raw data used in Figure 5.3.

It is interesting to note where the points of higher error occur, which gives rise to the question: is there a correlation between certain parameters and regions of high error? Figure 5.4 shows how median error varies with each parameter. Note that error is most closely correlated to the value of uniqueBytes. Prediction is particularly poor near the border between performance regions, that is, between the file cache and disk region. As expected, sharp drops in performance lead to unstable throughput and poor prediction.[1] Other than uniqueBytes, prediction accuracy is fairly independent of parameter values.

5.3.2. DECstation 5000/200, HP 730, Convex C240

Figures 5.5-5.10 show the prediction accuracy for the Sprite DECstation 5000/200, the HP 730, and the Convex C240 (raw data is given in Appendix A in Tables A.2, A.3, and A.4). For all systems, this chapter's method of predicting performance estimates a wide range of workloads accurately, with median errors ranging from 10-15% (Table 5.1). Table 5.1 also lists the inherent measurement error, which I measured by running the same set of random workloads twice and using one run to ''predict'' performance of the other run.

Figure 5.6 and 5.8 show how, similar to the SPARCstation 1+, error is usually most closely correlated to uniqueBytes. For the DECstation 5000/200, performance is unstable and

[1]If a user's workload fell near the border between performance regions, then instead of trying to accurately predict his workload's performance, his effort would be better spent in buying more memory for his system to expand the file cache region and hence to vastly improve his performance.

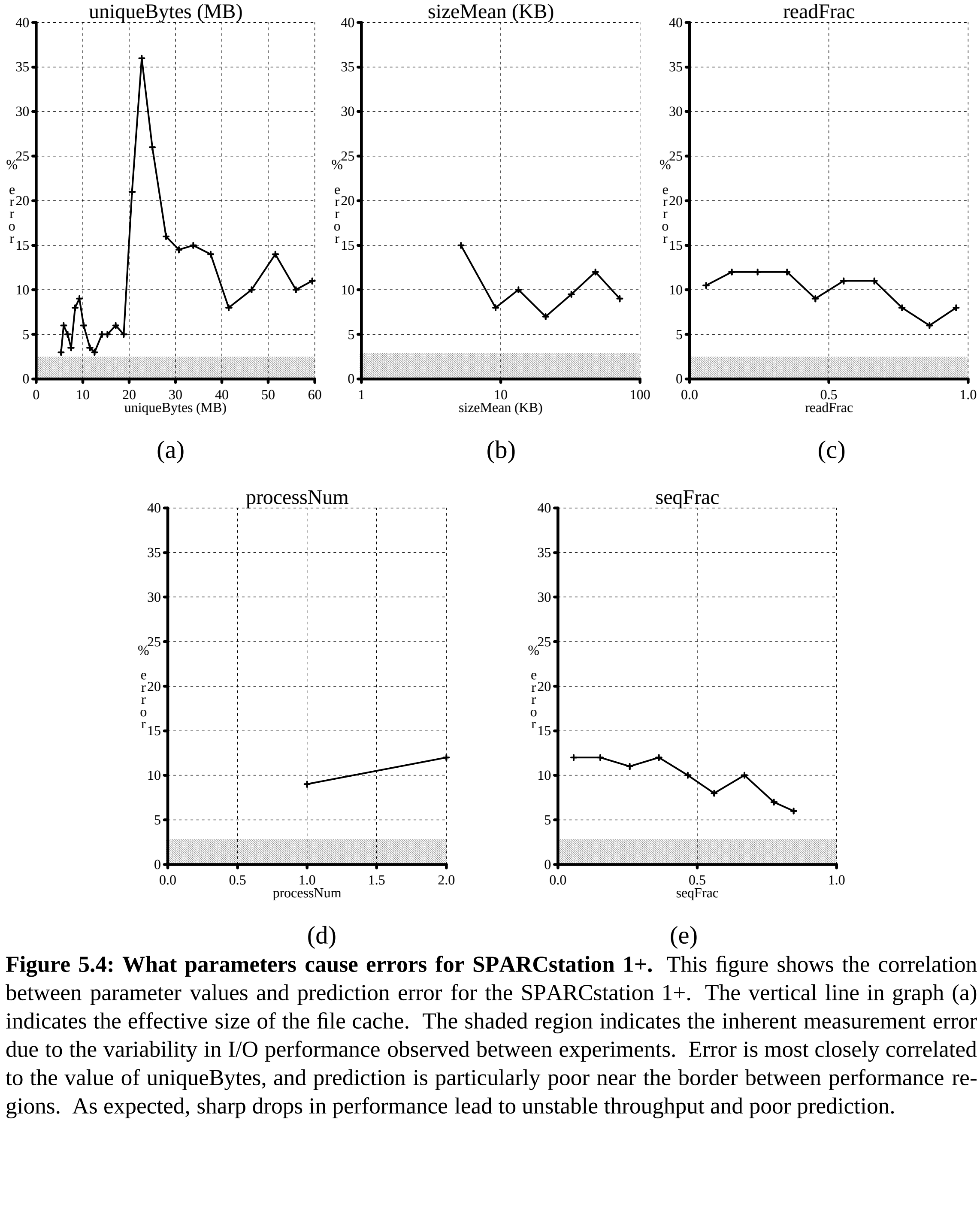
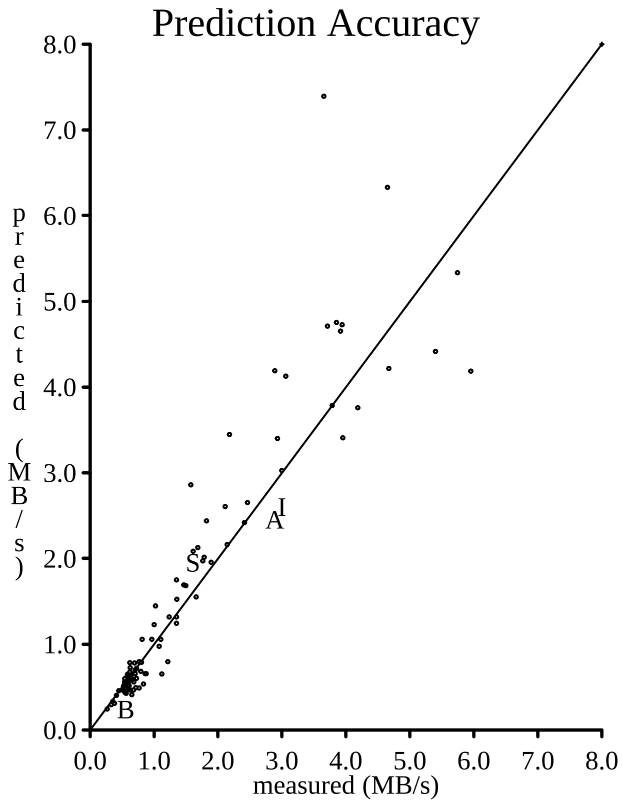
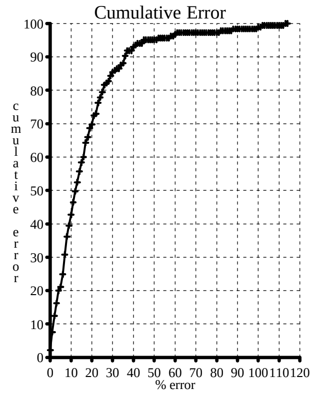


Figure 5.4: What parameters cause errors for SPARCstation 1+. This figure shows the correlation between parameter values and prediction error for the SPARCstation 1+. The vertical line in graph (a) indicates the effective size of the file cache. The shaded region indicates the inherent measurement error due to the variability in I/O performance observed between experiments. Error is most closely correlated to the value of uniqueBytes, and prediction is particularly poor near the border between performance regions. As expected, sharp drops in performance lead to unstable throughput and poor prediction.



(a)



(b)

Figure 5.5: Evaluation of prediction accuracy for Sprite DECstation 5000/200. This figure graphs the predicted performance against the actual (measured) performance for the DECstation 5000/200 in Figure 4.8. Points representing traditional benchmarks are shown as the letters ‘‘A’’, ‘‘B’’, ‘‘I’’, and ‘‘S’’, representing Andrew, Bonnie, IOStone, and Sdet, respectively. Median error is 12%. Performance for each workload ranges from 0.3 MB/s to 6 MB/s.

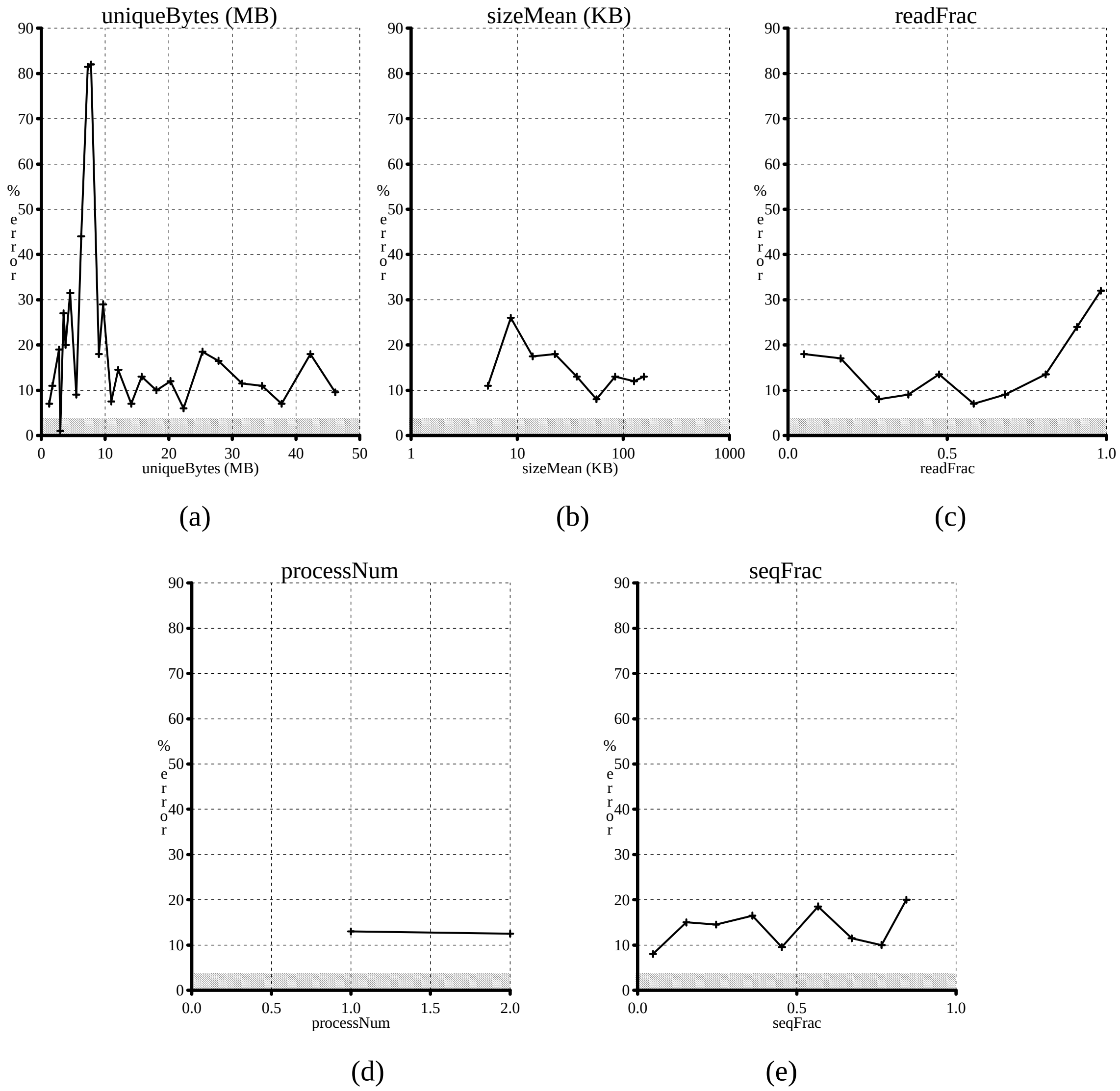
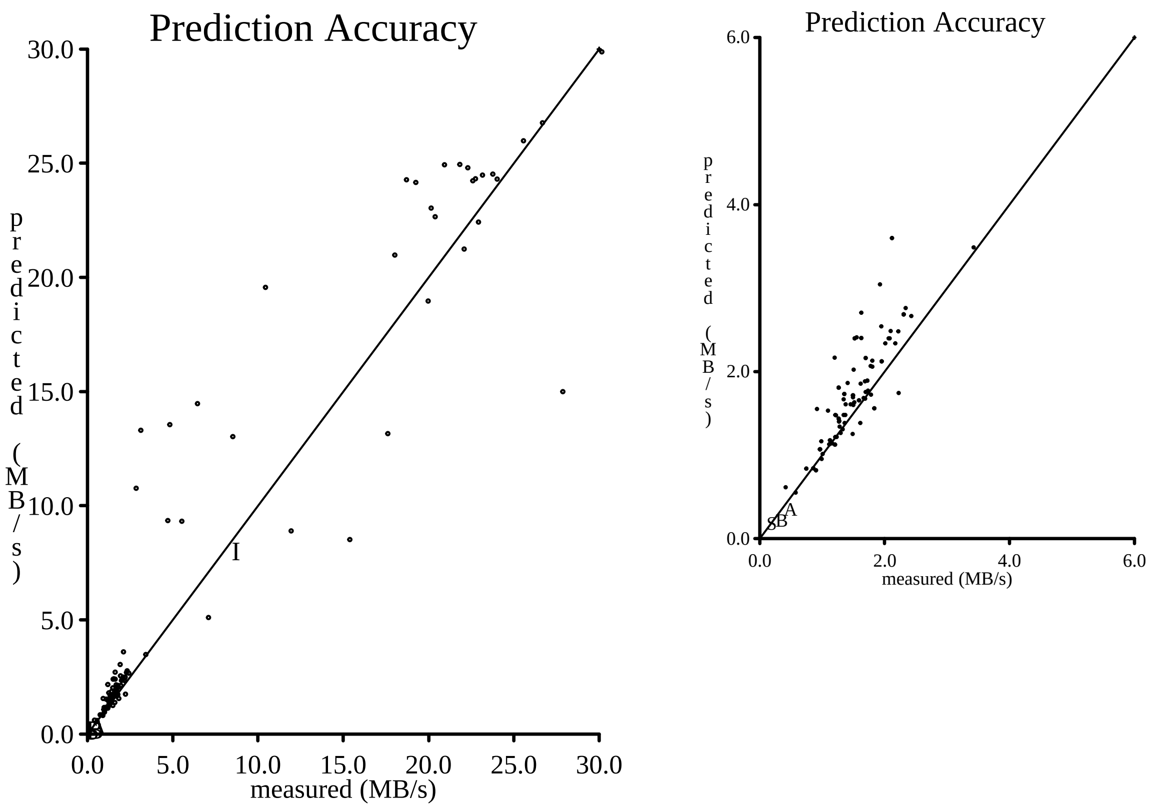
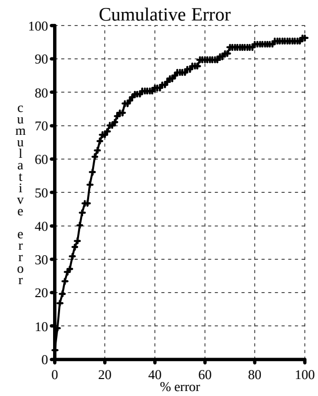


Figure 5.6: What parameters cause errors for DECstation 5000/200. This figure shows the correlation between parameter values and prediction error for the Sprite DECstation 5000/200. The vertical lines in graph (a) indicate the effective size of the file cache, one line for reads, the other for writes. The shaded region indicates the inherent measurement error due to the variability in I/O performance observed between experiments. Again, error is most closely correlated to the value of uniqueBytes; prediction is particularly poor near the border between performance regions. As expected, sharp drops in performance lead to unstable throughput and poor prediction.



(a)



(b)

Figure 5.7: Evaluation of prediction accuracy for HP 730. This figure graphs the predicted performance against the actual (measured) performance for the HP 730 in Figure 4.10 (a closeup of the 0 to 6 MB/s range is shown to the right). Points representing traditional benchmarks are shown as the letters ‘‘A’’, ‘‘B’’, ‘‘I’’, and ‘‘S’’, representing Andrew, Bonnie, IOStone, and Sdet, respectively. Median error is 13%. Performance for each workload ranges from 0.5 MB/s to 31 MB/s.

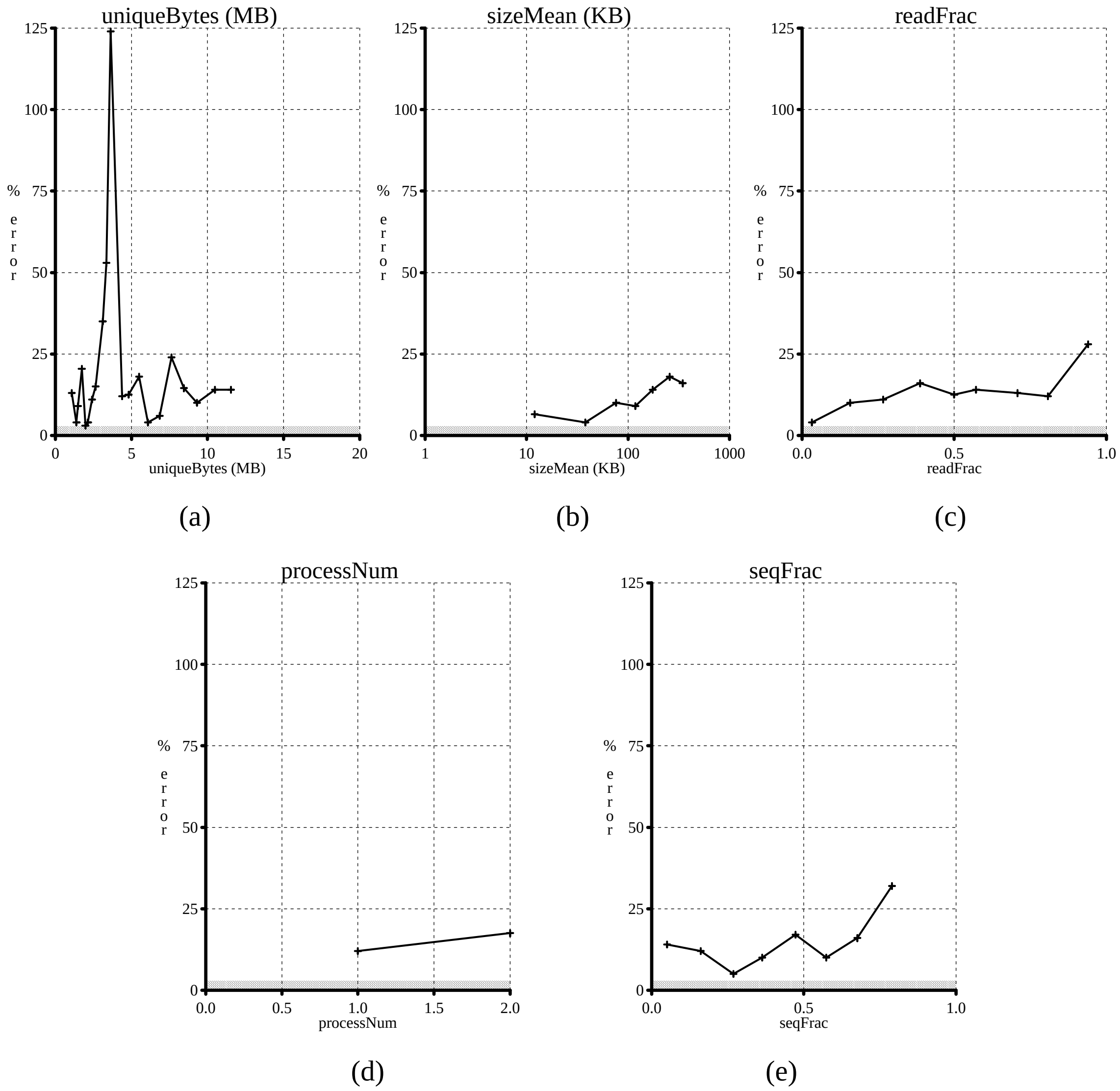
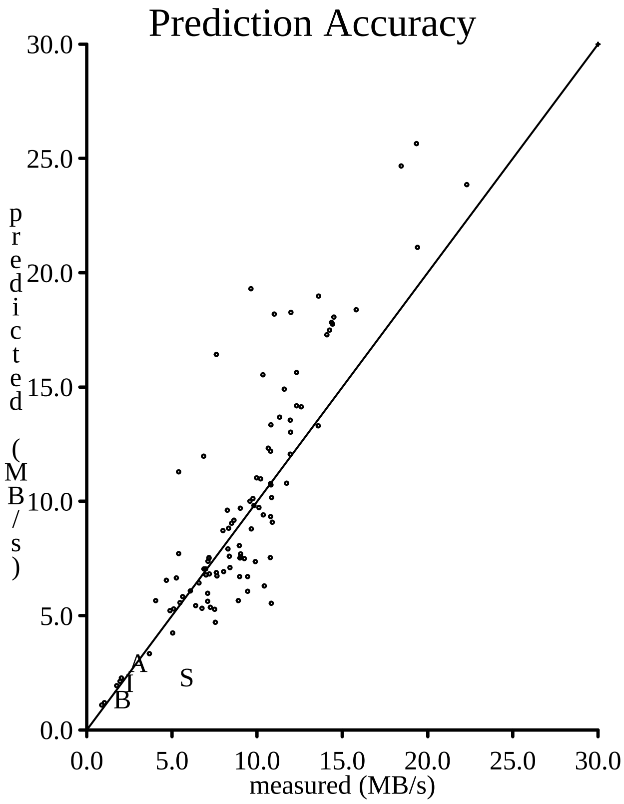
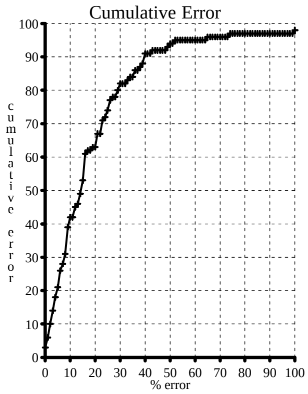


Figure 5.8: What parameters cause errors for HP 730. This figure shows the correlation between parameter values and prediction error for the HP 730. The vertical line in graph (a) indicates the effective size of the file cache. The shaded region indicates the inherent measurement error due to the variability in I/O performance observed between experiments. Again, error is most closely correlated to the value of uniqueBytes; prediction is particularly poor near the border between performance regions. As expected, sharp drops in performance lead to unstable throughput and poor prediction.

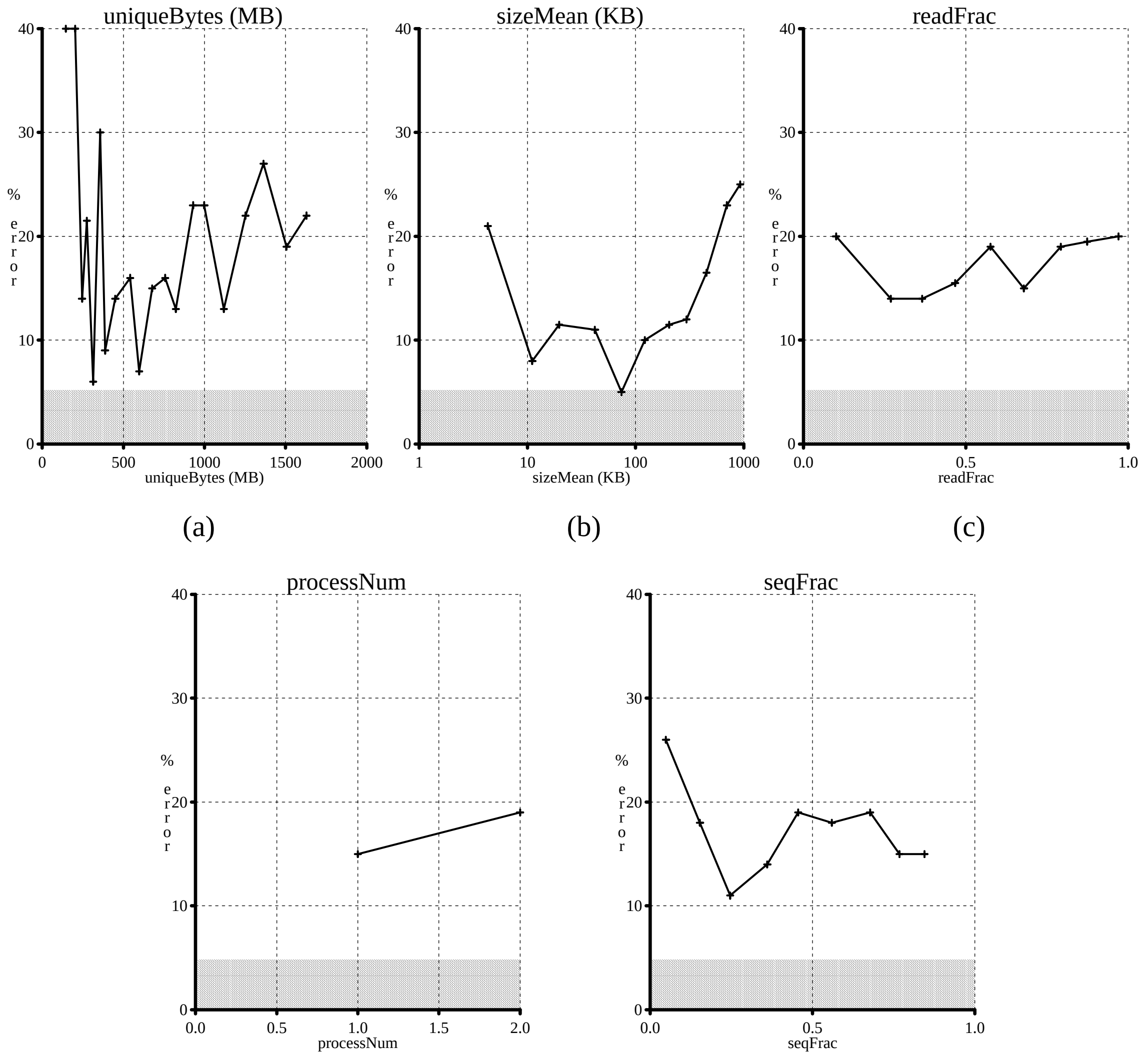


(a)



(b)

Figure 5.9: Evaluation of prediction accuracy for Convex C240. This figure graphs the predicted performance against the actual (measured) performance for the Convex C240 in Figure 4.11. Points representing traditional benchmarks are shown as the letters ‘‘A’’, ‘‘B’’, ‘‘I’’, and ‘‘S’’, representing Andrew, Bonnie, IOStone, and Sdet, respectively. Median error is 15%.



(d) (e)

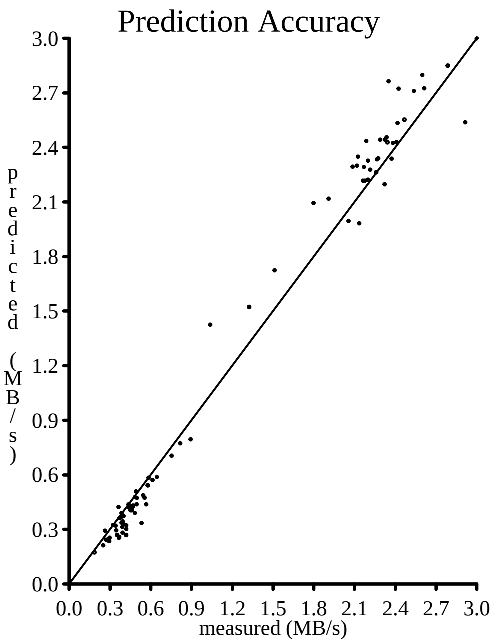
Figure 5.10: What parameters cause errors for Convex C240. This figure shows the correlation between parameter values and prediction error for the Convex C240. The vertical line in graph (a) indicates the effective size of the file cache. The shaded region indicates the inherent measurement error due to the variability in I/O performance observed between experiments. Error is most closely related to sizeMean. Because the Convex supports much larger sizes without degrading performance, it is a more challenging machine for which to predict performance. However, prediction when size is less than 300 KB is quite good (median error of 8%).

System	Median Error	Repeatability Error
SPARCstation 1+	10%	2%
DECstation 5000/200	12%	3%
HP 730	13%	3%
Convex C240	15%	5%

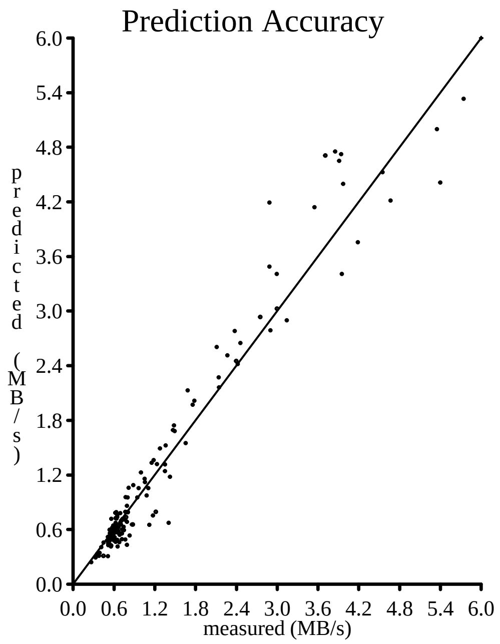
Table 5.1: Summary of median prediction errors. This table summarizes the prediction errors on all systems. The third column in the table lists the inherent measurement error, which was measured by running the same set of random workloads twice and using one run to ‘‘predict’’ performance of the other run.

prediction is poor when uniqueBytes is on the border between performance regions (6-10 MB). For the HP 730, this region of high error occurs between 3 and 4 MB. The HP 730’s exceptionally large drop in performance between the file cache and disk region highlights this high error region. Note in Figure 5.7 how most points of high error have performances between 5-20 MB/s. This level of performance can only occur when the system’s file cache is starting to thrash, which is when uniqueBytes is between 3 and 4 MB. Figure 5.11c shows the enhanced prediction when workloads with uniqueBytes in this range are not plotted. Median error drops from 13% to 8%, and all the points of high error in the 5-20 MB/s region of the scatter plot are no longer present. Figure 5.11a 5.11b show revised scatter plots for the SPARCstation 1+ and the DECstation 5000/200 without their workloads with uniqueBytes in the ranges 20-27 MB and 5-10 MB, respectively.

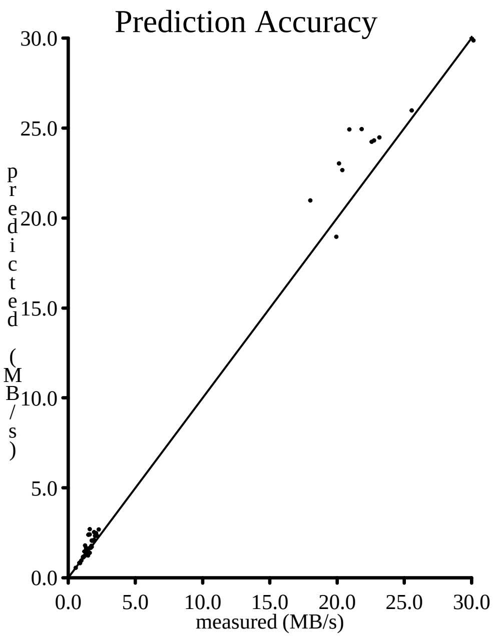
For the Convex C240, the amount of error shows more correlation to sizeMean than to uniqueBytes (Figure 5.10). Because the Convex supports larger sizes before degrading in performance, the self-scaling benchmark chooses a wider range over which to graph sizeMean. When sizes are restricted to a range similar to that of the other systems, prediction accuracy improves to a corresponding level. Figure 5.11d shows a scatter plot without workloads with sizes larger than 300 KB; median error here is only 8%. This indicates that the range of accurate prediction from each focal point may be limited, though not severely. I anticipate that as systems continue to evolve, the range of I/O workloads they support well will widen, forcing any performance prediction to gather and use more points to accurately characterize this



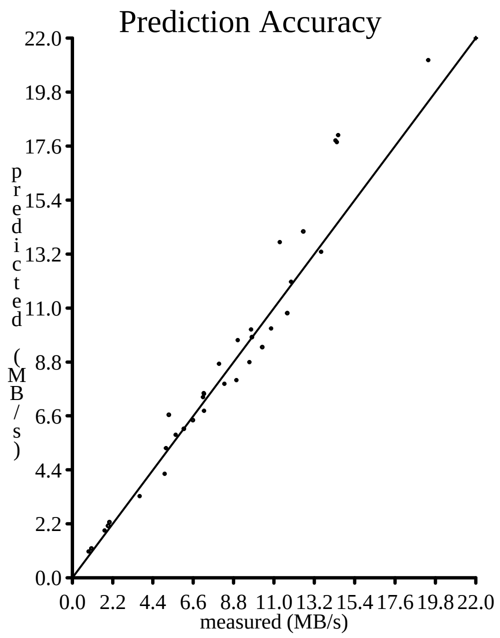
(a) SPARCstation 1+



(b) DECstation 5000/200



(c) HP 730



(d) Convex C240

Figure 5.11: Enhanced prediction accuracy. This figure shows the enhanced prediction accuracy when the points of high error are taken out of Figures 5.3, 5.5, 5.7, and 5.9. For the SPARCstation 1+, workloads with uniqueBytes between 20 and 27 MB are taken out and median error drops from 10% to 7%. For the DECstation 5000/200, workloads with uniqueBytes between 5 and 10 MB are taken out and median error drops from 12% to 10%. For the HP 730, workloads with uniqueBytes between 3 and 4 MB are taken out and median error drops from 13% to 8%. For the Convex C240, workloads with very large mean sizes (greater than 300 KB) are taken out and median error drops from 14% to 8%.

increased range. Perhaps the self-scaling benchmark will need to measure and graph more focal points to cover this wider range; perhaps the self-scaling benchmark will need to get feedback from performance prediction as it chooses focal points.

5.4. Comparison Against Orthogonal Sampling

The section above showed how a simple *product of single parameter functions* model can predict performance much more accurately than can single-point benchmarks. However, the *product of single parameter functions* method of prediction uses many more workloads than do single-workload benchmarks such as Andrew. Perhaps the increased accuracy shown in the previous section is due entirely to using more points and not to the validity of the independent functions model. In this section, I explore the accuracy of predicting performance using more workload points but *not* using the *product of single parameter functions* approach. This prediction is done by using multi-dimensional interpolation on an orthogonal sampling of workloads. Orthogonal sampling means selecting a fixed number of evenly spaced values from each parameter's range and measuring all combinations of workloads with those parameter values. For example, assume that a workload consists of two parameters, readFrac, ranging from 0-1, and uniqueBytes, ranging from 1 MB to 9 MB. Taking three values from each parameter would lead to an orthogonal sampling of nine total workloads of (readFrac, uniqueBytes) pairs: (0, 1MB), (0, 5 MB), (0, 9 MB), (0.5, 1MB), (0.5, 5 MB), (0.5, 9 MB), (1, 1MB), (1, 5 MB), and (1, 9 MB).

For the SPARCstation 1+, my original approach of using a *product of single parameter functions* requires the 84 points, each taking approximately 10 minutes to measure, that are returned by the self-scaling benchmark[2]. As can readily be seen, orthogonal sampling

[2]Six of the nine graphs have 10 points each; one (uniqueBytes) uses 20 points; two (processNum) use only two points each, due to the limited range of processNum.

inherently requires many more points. Sampling three values per dimension requires 162 points[3]. Figure 5.12 shows that even with twice as many points, interpolation using an orthogonal sampling gives much worse prediction accuracy than does the *product of single parameter functions*: the median error using orthogonal sampling is 22% versus 10% using the *product of single parameter functions*, and it takes twice as long to collect the necessary information.

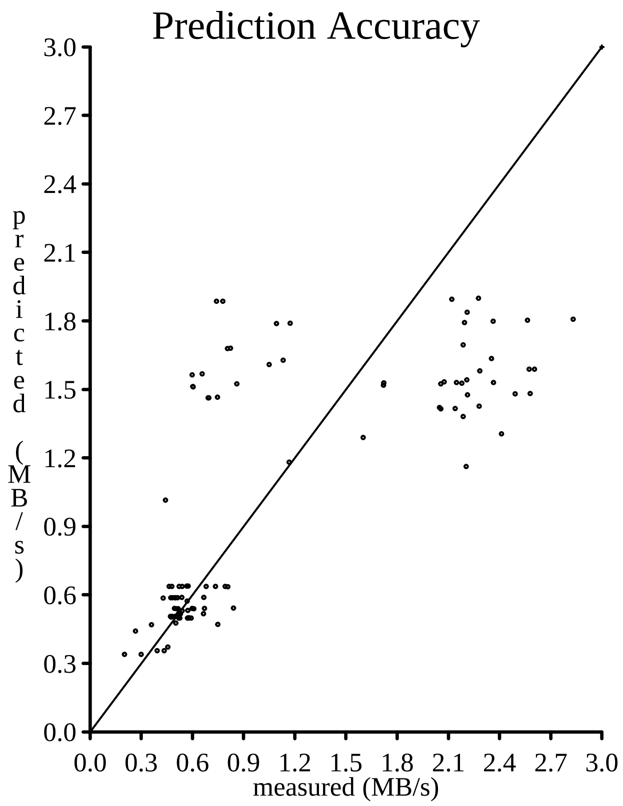


Figure 5.12: Prediction accuracy using interpolation on an orthogonal sample. This figure shows that interpolation using an orthogonal sample gives poorer prediction than does a prediction model based on *products of single parameter functions*. Even when using twice as many workloads to do prediction, median error is twice as high (22%).

[3]Normally this would be 3^5, or 243 points. However, due to the limited range of processNum, only two points are required for this dimension, which brings the total required to 162.

5.5. Application of Predicted Performance—Performance Ratios

Performance evaluators sometimes want to measure more than absolute performance. Their end goal is often to compare two systems head-to-head, generally as the ratio of performances. In this section, I apply the model of performance prediction developed in Section 5.2 to predict the performance ratio between several systems.

To measure and predict performance ratios, I define several workloads to be run on all systems. Table 5.2 describes these workloads and gives example applications for each. Figure 5.13 shows the measured and predicted ratios for these workloads. For the sake of comparison, these figures also show the performance ratio given by Andrew.

Though single-point benchmarks do not predict absolute performance well for a wide range of workloads, they could in theory still accurately estimate the ratio of two machines' performance. For this to be true, the ratio of two systems' performance would need to be constant and independent of the workload. This is not the case, however. Figure 5.13 shows that the performance ratio of two systems can vary greatly. For example, Figure 5.13a shows that for the large utility workload, a DECstation 5000 is five times faster than an HP 730; however, for the workstation and scientific read workloads, an HP 730 is three times faster than a DECstation 5000. Predicting performance using the output of the self-scaling benchmark captures this variability in the ratio of two systems' performance in a way that no single-point benchmark can. In addition, once the results from the self-scaling benchmark are in hand, prediction for any number of workload takes essentially no time. In contrast, measuring just the small set of workloads in Figure 5.13 took 2-3 hours for each system.

5.6. Conclusions

I have shown in this chapter that a simple *product of single parameter functions* approach to performance prediction is highly accurate, an outcome that has two important ramifications.

Title	Example	unique Bytes (MB)	seq Frac	read Frac	sizeMean (KB)	process Num
workstation	compile grep	1	0.8	0.8	4	1
large_utility	sort	10	0.9	0.6	8	1
scientific_write	satellite data	250	0.5	0.2	100	1
scientific_read	image process	250	0.5	0.8	100	1
database	TPC	500	0.1	0.2	4	2

Table 5.2: Workloads to be run on all systems. This table describes the workloads I use when comparing two systems.

First, it makes the graphs from the self-scaling benchmark much more useful, since they can be used to estimate performance for other workloads. Second, it supports the assumption of shape independence, that is, the shape of each graph from the self-scaling benchmarks is approximately independent of the values of the other parameters.

Section 5.5 applies predicted performance to the traditional use of benchmarking, which compares the performance of two systems. Performance prediction can be used to accurately predict the ratio of performance between two systems on identical workloads, even when that workload has not been run on either system. Single-workload benchmarks, in contrast, give poor estimates of performance ratios except for their own workloads.

The enhanced performance prediction over single-point benchmarks comes partly from using more measurements; however, the *product of single parameter functions* approach is equally, if not more, important. This was shown when interpolation could not predict performance accurately, even when using twice as many measured workloads.

The payoff for running the time consuming, self-scaling benchmark comes from predicted performance. The self-scaling benchmark need only be run once per configuration. This means either a manufacturer or an industry-wide performance group, such as SPEC, need only run the data once and then publish it for everyone else to use. Each possible workload can then be

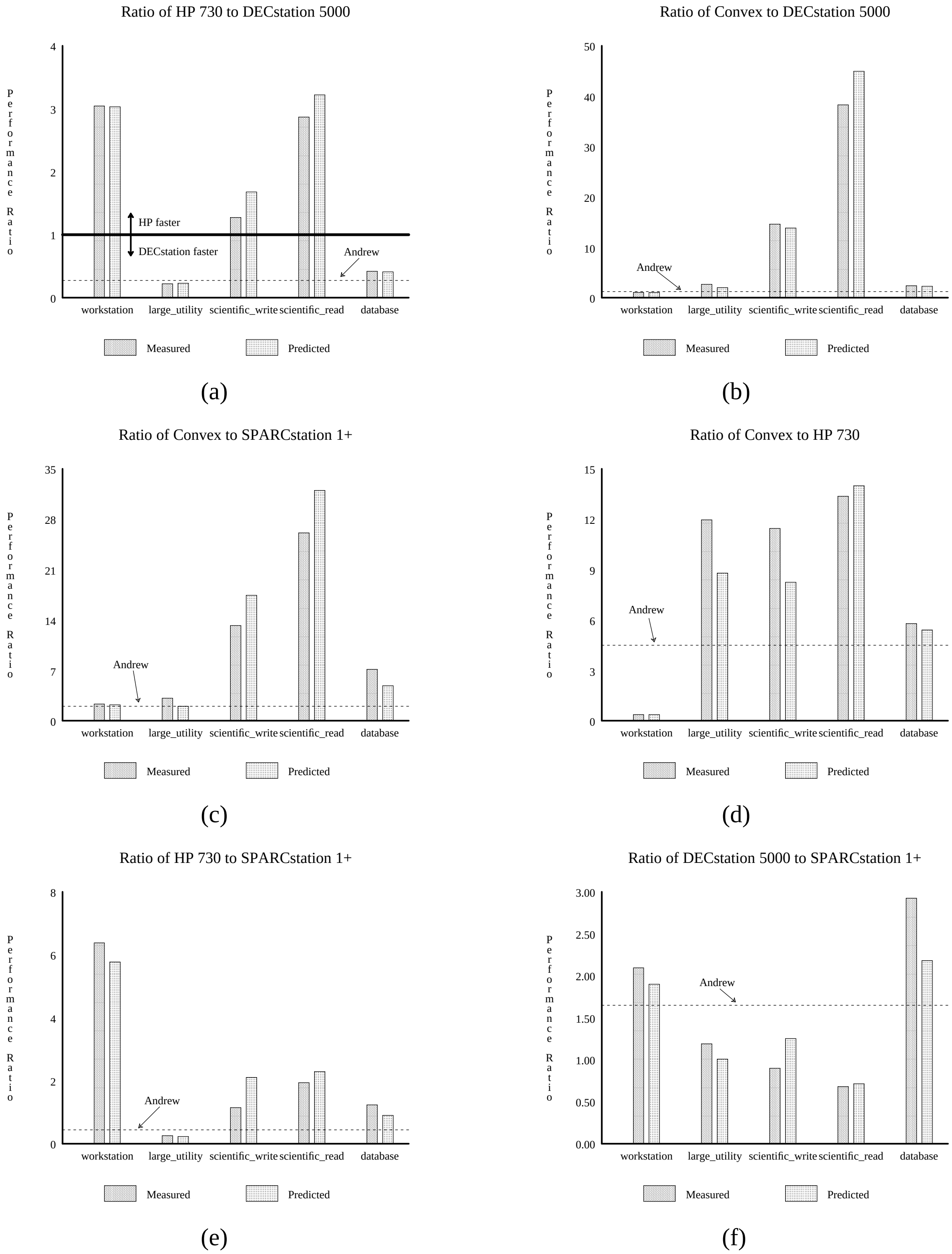


Figure 5.13: Measured versus predicted ratio. This figures shows how accurately systems can be compared using performance prediction. For comparison, the performance ratio given by Andrew is shown as a dashed line. Predicting performance using the output of the self-scaling benchmark captures this variability in the ratio of two systems' performance in a way that no single-point benchmark can.

predicted in milliseconds.

5.7. References

[Saavedra-Barrera89]
R. H. Saavedra-Barrera, A. J. Smith and E. Miya, ‘‘Machine Characterization Based on an Abstract High-Level Language Machine’’, *IEEE Transactions on Computers 38*, 12 (December 1989), 1659-1679.