Using the Distiller to Direct the Development of Self-Configuration Software

Zachary Kurmas
College of Computing
Georgia Tech
kurmasz@cc.gatech.edu

Kimberly Keeton
Storage Systems Department
Hewlett-Packard Labs
kkeeton@hpl.hp.com

Abstract

Many storage systems have become so complex that that the system administrator's salary represents almost half of the total cost of ownership. One approach to reducing this cost is to develop storage systems that can configure and manage themselves. Unfortunately, our ability to develop such software has been hindered by a limited understanding of how workloads and storage systems interact.

In [10], we presented the design of the Distiller *— our tool that automates the process of finding a workload's key performance-affecting attributes. In this paper, we distill three production workloads and show that the values of the chosen attributes contain information that will help self-configuring disk array to choose a reasonable prefetch length and RAID stripe unit size. We also discuss how the chosen attributes may help direct the development of algorithms that compute near-optimal prefetch lengths and stripe unit sizes.*

1. Introduction

Management is quickly be coming the dominant cost of many computer systems. Some systems have become so complex that only experts, who tend to be expensive, can efficiently configure and manage them. The storage system is a leading example of this trend. In the 2002 FAST[1] keynote address, David Patterson noted that the system administrator's salary represents almost half of the total cost of ownership for a storage system. The complexity of systems, and the consequent costs, will continue to increase as demands for high-performance I/O increase.

One solution is to develop storage systems that can configure and manage themselves. Such systems must make many decisions including: 1) which hardware to buy and how much, 2) how to organize the individual disks (e.g., RAID groups and RAID levels), and 3) how to set various parameters (e.g., RAID stripe unit size, prefetch length). The optimal configuration depends on the characteristics of the intended workload and the business's reliability and performance requirements.

We see two general approaches to automating these configuration decisions: a "compare" approach and a "compute" approach. A storage system using the compare approach will make configuration decisions based on the known behavior of similar workloads. For example, Hippodrome estimates the utilization of a previously unstudied workload on a given storage system configuration by looking up in a table the utilization of a "similar" workload on the same storage system [1]. In contrast, a storage system using the calculate approach will compute a configuration directly based on an analysis of the workload. For example, [2] gives rules for choosing the RAID level based on the number and size of the workload's write requests.

The key to both the compare and calculate approaches is choosing the appropriate analyses. When using the compare approach, one must choose the metrics by which two workloads will compared. When using the calculate approach, one must choosing the analyses that will provide the information on which the configuration will be based.

Researchers have proposed many metrics for quantifying workload characteristics [4, 6, 7, 8, 13, 14, 16]; however, selecting a set of metrics that captures precisely those aspects of a workload that should be compared and/or analyzed currently requires a trial-and-error process that, until recently, has been impractically tedious.

We have developed the *Distiller* — a tool that automates this trial-and-error process. Specifically, given a trace of a block-level I/O workload and a list of candidate metrics (formally called *attributes*), the Distiller chooses a set of "key" attributes that describe the workload's performance-affecting properties. When using a comparison approach, the chosen attributes will help define a similarity metric that is based on the workload's performance. When using a calculate approach, the chosen attributes will highlight the causes of the workload behavior observed, thereby aiding the development of models that can calculate a good configuration.

[1] FAST: File and Storage Technologies

This paper addresses two questions related to the Distiller's potential contribution to storage systems using the calculate approach:

Do the chosen attributes apply across several different hardware configurations? The Distiller chooses a set of "key" attributes that provide insight into the behavior of the workload on a single storage system configuration; however, in order to choose a configuration based on the values of the attributes, we must be sure the chosen attributes also contain information that can provide insights into the workload's behavior on the other configurations under consideration. In this paper, we show that the chosen attributes contain information that can allow us to understand the effects of modifying a storage system's prefetch length and RAID stripe unit size.

How are the attributes chosen by the Distiller related to the observed trends? Identifying a workload's key attributes is only the first step in using the calculate approach. The bigger challenge is to associate the values of those attributes with the disk array's performance for the different configurations. We will discuss how the attributes chosen by the Distiller fit our current understanding of workload performance and how researchers might potentially use this information.

The remainder of the paper is organized as follows: Section 2 provides a brief overview of the Distiller's design. Section 3 describes the experimental environment and the workloads used. Section 4 presents our results. Finally, section 5 discusses future work; and section 6 concludes.

2. Our approach

The Distiller is our tool for identifying a set of attributes that captures a workload's performance-affecting properties. An *attribute* is the name of a metric used to measure workload characteristics (e.g., mean request size, read percentage, or distribution of location values). An *attribute-value* is an attribute paired with the measurement itself (e.g., a mean request size of 8 KB, or a read percentage of 68%).

The Distiller takes as input a *target* workload trace and a library of attributes. It then automatically searches for a subset of those attributes that effectively describes the target workload's performance-affecting properties. The Distiller evaluates how well a given set of attributes captures the performance-affecting properties by (1) generating a synthetic workload with the same values for the chosen attributes, (2) issuing both workloads to the storage system under test, then (3) comparing the performance of the two workloads. If the two workloads perform differently, then we know that there is some important property that is not described by the chosen attributes. If they have similar performance, then we argue that the attributes have captured the important performance-related information.

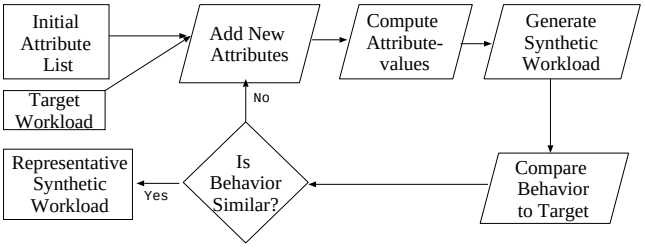

Figure 1. The Distiller's iterative loop.

2.1 Related work

The Distiller is unique because, instead of attempting to preserve a set of attributes chosen *a priori*, it searches through a set of previously developed and studied attributes and generation techniques (e.g., [4, 5, 6, 7, 8, 13, 14, 16]) then chooses those attributes that are most appropriate for the target workload and storage system under test. These existing techniques serve as the Distiller's library.

Researchers have used principal component analysis (PCA) to identify the attribute-values that concisely describe a set of batch computational workloads [3]. Given a large set of workloads and a set of attribute-values that characterize those workloads, PCA computes new variables, called principal components, that best describe the workloads. These principal components are uncorrelated linear combinations of the original attribute-values. Choosing attributes in this manner presents several challenges: First, to use PCA to identify performance-related attributes, we need a set of workloads with similar performance. We do not currently have access to such a set. Second, the resulting principal components are linear combinations of attributes, rather than a subset of the initial list of attributes, and hence may not have an intuitive meaning.

Standard feature selection techniques are used to find a small set of features (in our case, attributes) that can be used to differentiate objects in a set (in our case, workloads). Again, our set of workloads is not large enough to be effectively analyzed using standard feature selection techniques.

2.2 The Distiller

At a high level, the Distiller iteratively builds a list of "key" attributes — attributes that noticeably influence the target workload's performance. During each iteration, the Distiller identifies one additional key attribute, adds it to the list, then tests the quality the resulting synthetic workload. This loop (shown in figure 1) continues until either (1) the difference between the performance of the synthetic and target workloads is below some user-specified threshold, or (2) the Distiller is not able to produce a more accurate synthetic workload by adding attributes in its library.

Evaluating a synthetic workload's performance takes

tens of minutes. Therefore, we limit the number of intermediate workloads the Distiller builds and evaluates by partitioning the library of attributes into groups and investigating only those groups that contain at least one "important" attribute. Each *attribute group* is a set of attributes whose values are calculated using the same set of I/O request parameters. For example, the {location, operation type} group contains only those attributes that consider both location and operation type (e.g., separate histograms of location read requests and write requests).

To determine whether an attribute-group contains any important attributes, we compare the performance of two synthetic workloads: The first preserves (almost) none of the attributes in the group under test (by choosing the values of the parameter(s) under study independently at random from the distribution of values in the target workload). The second preserves every attribute in that group (by using the actual list of values from the target workload). The difference in performance of these two workloads is an estimate of the importance of the attributes in the group. Should there be little or no difference in performance, we know that we need not evaluate individual attributes in that group, because the default distribution produces similar performance.

Once the Distiller has identified an attribute group that contains an important attribute, it searches that group for an important attribute. The Distiller evaluates the importance of an individual attribute by comparing the performance of two synthetic workloads: The first workload preserves the attribute under test; the second does not. The Distiller orders the attributes in the chosen group based on the amount of data necessary to represent the measured attribute-value. It then evaluates the attributes in order of increasing size and chooses the smallest attribute that meets a given accuracy threshold.

In practice, the synthetic workloads compared must be designed carefully so that the only differences between them are related to the attribute group under study. We refer the reader to [10] for details of the Distiller's architecture.

3. Experimental environment

This section discusses the storage system, workloads, and supplementary software used to evaluate the Distiller.

3.1 Storage system

In section 4, we use the Distiller to analyze the workloads described in section 3.2. The disk arrays currently available to us are too small to handle these workloads. Therefore, we use the Pantheon disk array simulator to simulate the execution of our workloads [15].

Pantheon simulates disk arrays comprising several disks connected to one or more controllers by parallel SCSI busses. The controllers have large non-volatile-RAM caches. (This general architecture is similar to the FC-60 used in [10].) Pantheon provides many additional configuration parameters including: number and type of component disks, size of cache, prefetch length (minimum amount of data prefetched upon each read request), high- and low-water marks (points at which the cache begins de-staging dirty data), and size of the RAID stripe unit. Table 2 provides the Pantheon configuration used to study each workload.

Table 1. Summary of Workloads

workload	workload parameters				
	I/Os	t'put (MB/s)	read percent	arrival rate (I/Os / sec)	LUs
OpenMail	1287941	2.33	28%	358	22
OLTP	4257935	1.19	51%	538	37
DSS	332394	32.14	100%	185	4

3.2 Workloads

In this section, we discuss the high-level characteristics of the workloads used to evaluate the Distiller. Table 1 summarizes these characteristics.

OpenMail: This one hour trace of the OpenMail Email server contains 12897941 I/Os for a mean request rate of 358 I/Os per second. This trace comprises 22 LUs. When investigating this workload, we configured Pantheon to simulate a disk array with a 1GB cache and 180 9GB Seagate disks arranged into 45 18GB LUs.

OLTP: This 1994 online transaction processing (OLTP) trace measures HP's Client/Server database application running a TPC-C-like workload at about 1150 transactions per minute on a 100-warehouse database. When investigating this workload, we configured Pantheon to simulate a disk array with 80 1GB Wolverine disks arranged into 40 1GB LUs. The simulated disk array has a 256MB cache. This configuration is nearly identical to that of the disk array on which the trace was collected.

DSS: This decision support system (DSS) trace was collected on an audited TPC-H system running the throughput test (multiple simultaneous queries) on a 300 GB scale factor data set. This workload comprises 8373997 I/Os over 80 LUs. Most LUs are read-only; nearly all I/Os on these LUs are 128 KB in size. Each query generates a series of sequential I/Os. Visual inspection of the read-only LUs shows many independent sequential streams interleaved together.

Repeatedly analyzing, synthesizing, and simulating the entire DSS workload requires considerable computational and storage resources. Although not intractable, we decided that our limited shared resources would be better used analyzing a variety of workloads. Therefore, we investigate the

Table 2. Summary of Pantheon Configurations

workload	num disks	num busses	num raid groups	disk size	disk type	cache size	bus rate
OpenMail	180	4	45	9 GB	Seagate Cheetah 10K rpm	1 GB	40 MB/s
OLTP	80	2	40	1 GB	Wolverine III (HPC2490A)	256 MB	40 MB/s
DSS	16	4	4	9 GB	Seagate Cheetah 10K rpm	256 MB	100 MB/s

four busiest LUs, whose characteristics are summarized in table 1.

3.3 Software

The Distiller utilizes several software packages: Our workload analysis tool, Rubicon, takes a workload trace and an attribute list as input and produces the attribute-values that characterize the workload [12]. Our workload generation tool takes a workload characterization (i.e., a set of attribute-values) as input, and generates a synthetic workload matching that characterization. Pantheon simulates the execution of a workload. As it does, it calculates the performance of the simulated disk array.

To collect workload traces, we use the Measurement Interface Daemon (midaemon) kernel measurement system, part of the standard HP-UX MeasureWare performance evaluation suite [9]. For each I/O request, the midaemon provides the workload parameters. We do not attempt to model the CPU time between I/Os. Consequently, changing the disk array configuration does not affect the synthetic workload's arrival pattern.

3.4 Distiller configuration

Before running the Distiller, the user must select 1) a *demerit figure* to quantify the difference between the performance of two workloads, and 2) a threshold value below which attribute groups are not explored and individual attributes are not considered "important". We discuss three demerit figures:

1. **Mean response time:** The mean response time (MRT) is the simplest, but least accurate demerit figure. A synthetic workload should have a mean response time similar to the workload it models; however, two workloads with very different behaviors can have similar mean response times. We express this demerit figure as the percent difference between the mean response times of the compared workloads.

2. **Root-mean-square:** The root-mean-square (RMS) is the demerit figure used in the related work (e.g., [4], [11]). Specifically, RMS is the root mean square of the horizontal distance between the response time cumulative distribution functions (CDFs) for the synthetic and target workloads. We normalize this demerit figure by presenting it as a percentage of the mean response time of the target workload.

   The RMS demerit figure is useful because it measures similarity based on the performance observed by the user. However, because the RMS metric sums the square of horizontal differences, workloads whose CDFs have horizontal "plateaus" tend to have very large RMS values — especially when those plateaus are near 1 on the y-axis (i.e., "heavy tails").

3. **Log area:** The log area demerit figure is the area between two CDFs plotted with a log scale on the x-axis (as are figures 2 through 4). Using the log scale causes differences at all percentiles to be weighted equally. This demerit figure more accurately reflects the similarity of the workload's overall performance, but deemphasizes differences most noticeable to the user.

When running the Distiller, we use the log area demerit figure with threshold of 7.5%. (These configurations were chosen empirically based on our experience.) The Distiller stops when it either (1) produces a synthetic workloads with a log area demerit figure below 10%, or (2) when it determines that the attributes in the library will not produce a more accurate synthetic workload. (Note that the demerit figure and threshold chosen as terminating conditions need not be the same as the demerit figure and threshold used internally to select attributes and attribute groups.)

4. Experimental results

In this section we analyze the Distiller's ability to identify performance-affecting attributes, evaluate how well the chosen attributes capture information related to a range of prefetch lengths and stripe unit sizes, and discuss how the workload behaviors observed are related to the values of the chosen attributes.

Figures 2 through 4 show the accuracy of the synthetic workloads specified by the Distiller. Table 4 quantifies the accuracy of these synthetic workload using the demerit figures discussed in section 3.4. These demerit figures are necessary to allow the Distiller to run automatically in a reasonable amount of time; however, they are not necessarily the best measure of the overall quality of the attributes chosen by the Distiller.

Table 3. Summary of Attributes Chosen

workload	attribute
OpenMail	separate distributions of read and write locations Markov model of operation type distribution of request size interarrival time clustering [8]
OLTP	separate distributions of read and write locations distribution of operation type run lengths distribution of request size Markov model of interarrival time (with run counts of short interarrival times)
DSS	list of interleaved runs of location distribution of operation type distribution of request size distribution of interarrival time

Table 4. Summary of Initial Synthetic Workloads

workload	MRT	RMS	log area
OpenMail	8.3%	33%	10%
OLTP	26%	176%	30%
DSS	5%	2%	2%

The true quality of a set of attributes should ultimately be based upon whether the attributes contain enough information about the workload's performance to meet the user's needs. In the context of a self-configuring storage system using the "compute" approach, we will show that the chosen attributes contain enough information to predict the effects of modifying the simulated disk array's prefetch length and RAID stripe unit size. We will then discuss how the observed behaviors are related to the values of the chosen attributes. Table 3 shows the attributes chosen for each workload.

4.1 Prefetch length

Workloads that exhibit a high degree of spatial locality or sequentiality may benefit from *prefetching*. Upon receiving a request to read the data in sectors x_{start} through x_{end}, a disk array configured to prefetch i sectors of data will also read the data in sectors $x_{end} + 1$ through $x_{end} + i$ and place it in the cache. Prefetching improves workload performance when the prefetched data is requested before being evicted from the cache; prefetching degrades performance when unrequested prefetch data delays other I/O requests.

To show that the chosen attributes capture enough information related to prefetch length, we issued each production workload to a simulated disk array several times, varying the prefetch length from 0 to 1MB. We then replayed the synthetic workloads using the same prefetch lengths and compared the mean response times. Figures 5 through 7 show the results.

4.1.1 OpenMail and OLTP

As shown in figure 5, the original and synthetic OpenMail workloads produce the same trend: The mean response time increases slowly for prefetch lengths less than 256KB, then increase very rapidly when the prefetch length is greater than 256KB. Similarly, figure 6 shows that the original and synthetic version of the OLTP workload also produce the same trend. Thus, we see that the attributes chosen by the Distiller are useful for helping to predict the effects of modifying the disk array's prefetch length.

The attribute that most influences the OpenMail and OLTP workloads is the separate distributions of read locations and write locations. This is because the disk array's NVRAM cache allows all writes to be completed as soon as they are stored in the cache; whereas, reads that miss in cache are not completed until the data has been retrieved from disk. Consequently, it is essential that the reads and writes (which have very different amounts of temporal locality) be treated separately.

Notice that both the OpenMail and OLTP workloads can be accurately synthesized using only distributions of location. This indicates that there is little or no spatial locality in either workload. This observation is consistent with figures 5 and 6, which show that neither workload benefits from prefetching. Thus, the Distiller's choice of attributes has provided a useful hint about how to configure prefetching for these workloads.

4.1.2 DSS

As with the OpenMail and OLTP workloads, figure 7 shows that the mean response times of the target and synthetic DSS workloads remain close at all prefetch lengths.

The DSS workload is a set of concurrent database queries. Each query produces several long runs of 128KB I/O requests. These interleaved runs represent a high degree of spatial locality. Consequently, we expect that increasing the prefetch length will increase performance. Figure 7 shows that this is, in fact, the case. However, the causes of the increases in mean response time at 64KB and 5512KB are not as obvious.

From the distribution of request size, we know that almost every request is 128KB. When the prefetch length is less than 128KB, future requests are queued behind prefetch requests; however, because the prefetched data does not complete the next request, the next request still requires a physical disk access. As a result, most of the time spent prefetching is wasted. This explains the increase in response time as prefetch length varies from 0 to 64KB.

Mean response time also begins to increase when prefetch length reaches 512KB (4 I/Os). The cause of this

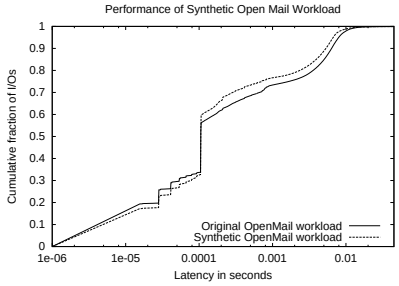

Figure 2. Performance of Synthetic OpenMail Workload

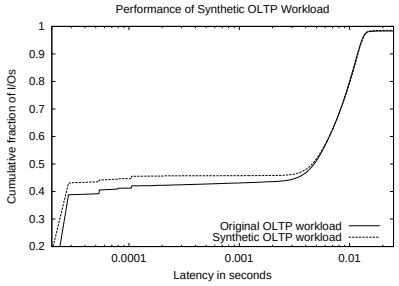

Figure 3. Performance of Synthetic OLTP Workload

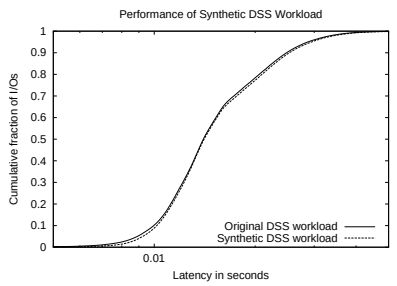

Figure 4. Performance of Synthetic DSS Workload

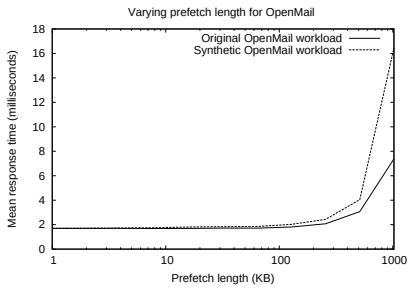

Figure 5. Mean resp. time of OM as prefetch length varies

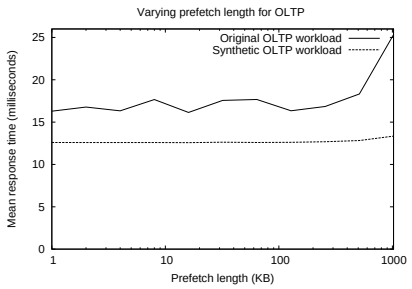

Figure 6. Mean resp. time of OLTP as prefetch length varies

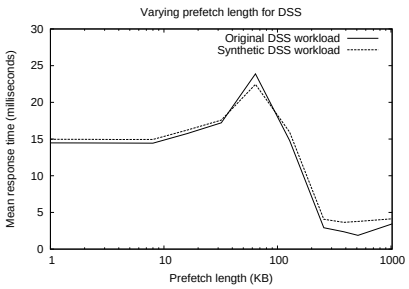

Figure 7. Mean resp. time of DSS as prefetch length varies

trend is not clear because a prefetch length of 512KB will not fill the cache and cause unused prefetched data to be evicted. The disk array does not prioritize I/Os; therefore, we suspect that non-prefetched I/Os are being queued behind prefetch requests. This situation emphasizes the fact that finding key attributes is only one step toward writing self-configuring disk arrays. We must also understand the disk array's hardware and firmware.

4.2 Stripe unit size

The disk arrays simulated by Pantheon use a RAID 1/0 (e.g., striped mirror) configuration. The disk array's disks are partitioned into RAID groups where half of the disks in each group mirror the other half. The stripe unit size defines how much data is stored contiguously on one disk before moving to the next disk in the RAID group. The stripe unit size affects (among other things) at which point several disks will be simultaneously servicing a single I/O. If a request spans several stripe units, then the requested data will be located on several disks. This parallel access can improve performance by reducing the transfer time; however, it can also degrade performance by increasing the number of seeks (i.e., causing two different disks to seek to similar locations, instead of allowing the second disk to seek to the location of a different request).

Figures 8 through 10 demonstrate the ability of the synthetic workloads to predict the effects of varying the stripe unit size from 2KB to 2MB.

4.2.1 OpenMail

The synthetic OpenMail workload predicts the general trend: Mean response time decreases as stripe unit size increases, until the stripe unit size reaches 128KB. At this point, the mean response time remains steady.

This trend of decreasing response time as stripe unit size increases is also consistent with the OpenMail workload's lack of spatial locality. The stripe unit size determines how much data is grouped onto one disk. If a request's size is less than the size of the stripe unit, only one disk serves the request. Otherwise, more than one disk serves the request. Because the OpenMail workload has so little spatial locality, we expect seek time to dominate the cost of serving requests. Using a large stripe unit size will cause all requests to be served by only one disk, leaving the second disk free to serve future requests. If more than one disk serves each request, then future requests will be queued while waiting for all disks to move their read heads.

Notice also that the mean response time decreases rapidly at first, then levels off for larger stripe unit sizes. A large percentage of the workload's requests are 1KB and 8KB. Thus, the benefit of further increases in stripe unit size decreases once the stripe unit size reaches 8KB.

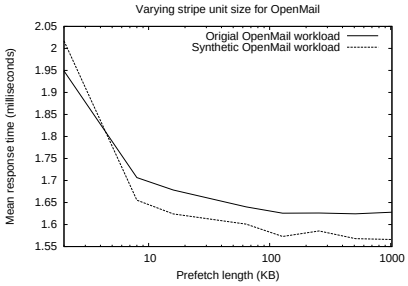

Figure 8. Mean resp. time of OM as stripe unit size varies

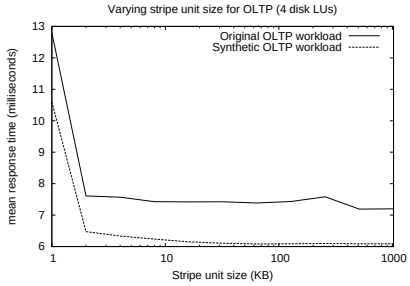

Figure 9. Mean resp. time of OLTP as stripe unit size varies

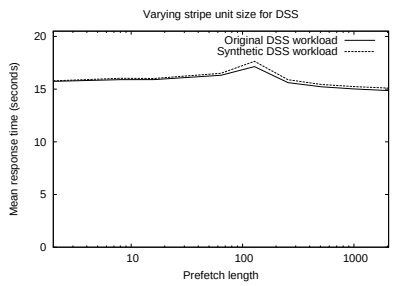

Figure 10. Mean response time of DSS as stripe unit size varies

4.2.2 OLTP

The synthetic OLTP workload also exhibits the same behavior as the original workload: stripe unit size has no effect on performance. In this case, stripe unit size does not affect performance because the LUs in the simulated disk array have only two disks; therefore, stripe unit size has no effect on data layout.

In order to present a more interesting result, we re-ran the experiment simulating a disk array with four disks per LU. Figure 9 shows the result. In this case, we see behavior similar to the OpenMail stripe unit size test. However, in this case, the performance levels off after the size of the stripe unit reaches 2KB. This is consistent with the fact that over 90% of the OLTP workload's requests are 2KB.

4.2.3 DSS

As with prefetch length, we see in figure 10 that the mean response times of the target and synthetic workloads remain close at all stripe unit sizes.

The observed changes in performance as stripe unit size changes make sense given our understanding of the RAID 1/0 data layout algorithm. As with the OpenMail workload, the stripe unit size should be set to minimize seek times. Each RAID group has 4 disks. Read requests are directed round-robin to the primary and mirror disks. When the stripe unit size is less than 128KB, then each I/O is striped across two disks. Suppose an LU contains exactly 4 interleaved, synchronized streams. Then each request will cause two disks to seek. The requests from streams 1 and 3 will go to disks 1 and 2; and the requests from streams 2 and 4 will go to disks 3 and 4. Thus (when prefetching is set to 0), the read heads will "ping-pong" between the locations for two streams.

Setting the stripe unit size to 128KB does not eliminate this problem. Although each I/O is now on only one disk, the striping algorithm will stripe consecutive requests onto alternating disks. Thus, each request still must suffer a seek; however, there is no longer the benefit of having two disks serving the request concurrently. When the stripe unit size increases to 256KB, then two adjacent I/Os are striped onto a single disk. Thus, when there are four streams, it is possible to require only one seek for every two I/Os.

5. Ongoing and future work

This paper demonstrates how the Distiller can contribute to the development of a self-configuring storage system that uses the "compute" approach. However, there are still many open questions.

Given a storage system, is there a single set of attributes that can be used to fully analyze any workload? Table 3 shows that the Distiller chose different {operation type, location} attributes for the OpenMail and DSS workloads. In each case, we were able to use the chosen attributes to explain the workload's behavior as prefetch length and stripe unit size changed. However, in order to develop an algorithm to choose an optimal prefetch length or stripe unit size, we must find a reasonably-sized set of attributes that provides the necessary input. We hypothesize that we can generate such a set by distilling many workloads, then taking the union of all attributes chosen.

Will such a "super-set" of attributes be of manageable size? Ideally, the union of all attributes chosen by the Distiller for all workloads will contain all performance-related information; however, if that union contains hundreds of attributes, its usefulness will be limited. It will be difficult to generate a synthetic workload with which we can verify its correctness. Also, developing an algorithm that considers hundreds of attributes will be quite challenging.

The question of finding one set of attributes per storage system also applies to the "compare" approach. When comparing two workloads using a set of attributes, we must be sure that the resulting attribute-values contain all the relevant performance-related information for both workloads. Furthermore, if the comparison uses a lookup table, it must be small enough to be populated it in a reasonable amount of time. Currently, the attributes chosen by the Distiller are not scalars. They tend to be distributions (represented by histograms with several hundred bins) and Markov models (which include distributions and transition matrices).

The development of autonomic computing algorithms is just one of several uses of the attributes chosen by the Distiller. Our high-level goal is to develop a better understanding of how workloads and storage systems interact by studying how the attributes chosen by the Distiller differ as workloads and storage systems change. We expect that this knowledge will aid the design of storage system hardware, firmware, configuration policies, and analytic models. For example, by learning precisely which attributes have the largest effect on performance, we may learn how to identify precisely those patterns within a workload that firmware should be tuned to handle.

Also, we believe that the synthetic workloads used to evaluate the chosen attributes will help storage systems researchers by providing more evaluation workloads. Currently, there is a very limited supply of traces of production workloads. Many system administrators hesitate to make traces publicly available because they may contain sensitive data. In contrast, synthetic workloads are specified using high-level information (i.e., attribute-values) that is much less sensitive. Consequently, we hope that improving our ability to easily generate accurate synthetic workloads will increase the number of workload traces available to those who study and evaluate storage systems.

Finally, we believe the techniques used by the Distiller generalize to other areas of computer systems design. We plan to investigate how the Distiller can be used to identify the key attributes other workloads and possibly use those attributes a basis for other autonomic systems.

6. Conclusions

We demonstrated that the attributes chosen by the Distiller are of use to those who design autonomic computing algorithms. First, we showed that the attributes contain the information necessary to predict the consequences of modifying a storage system's prefetch length and stripe unit size. This information can, in theory, be used to select a reasonable configuration. Second, we briefly explained how the observed consequences could be explained given the information in the chosen attributes and an understanding of the storage system's configuration and firmware. Although there is much to be done in developing self-managing storage systems, the Distiller provides an important step.

References

[1] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch. Hippodrome: Running circles around storage administration. In *Proceedings of the Conference on File and Storage Technologies*, pages 175–188. IEEE, January 2002.

[2] E. Anderson, R. Swaminathan, A. Veitch, G. A. Alvarez, and J. Wilkes. Selecting RAID levels for disk arrays. In *Proceedings of the Conference on File and Storage Technologies*, January 2002.

[3] M. Calzarossa and G. Serazzi. Construction and use of multiclass workload models. *Performance Evaluation*, 19:341–352, 1994.

[4] G. R. Ganger. Generating representative synthetic workloads: An unsolved problem. In *Proceedings of the Computer Measurement Group Conference*, pages 1263–1269, December 1995.

[5] M. E. Gomez and V. Santonja. A new approach in the analysis and modeling of disk access patterns. In *Performance Analysis of Systems and Software (ISPASS 2000)*, pages 172–177. IEEE, April 2000.

[6] M. E. Gomez and V. Santonja. A new approach in the modeling and generation of synthetic disk workload. In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 199–206. IEEE, 2000.

[7] B. Hong and T. Madhyastha. The relevance of long-range dependence in disk traffi c and implications for trace synthesis. Technical report, University of California at Santa Cruz, 2002.

[8] B. Hong, T. Madhyastha, and B. Zhang. Cluster-based input/output trace synthesis. Technical report, University of California at Santa Cruz, 2002.

[9] HP OpenView Integration Lab. *HP OpenView Data Extraction and Reporting*. Hewlett-Packard Company, Available from http://managementsoftware.hp.com/library/papers/index.asp, version 1.02 edition, February 1999.

[10] Z. Kurmas, K. Keeton, and K. Mackenzie. Iterative distillation of I/O workloads. In *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2003.

[11] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–29, March 1994.

[12] A. Veitch and K. Keeton. The Rubicon workload characterization tool. Technical Report HPL-SSP-2003-13, HP Labs, Storage Systems Department, Available from http://www.hpl.hp.com/SSP/papers/, March 2003.

[13] M. Wang, A. Ailamaki, and C. Faloutsos. Capturing the spatio-temporal behavior of real traffi c data. In *Performance 2002*, 2002.

[14] M. Wang, T. M. Madhyastha, N. H. Chan, S. Papadimitriou, and C. Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffi c. In *Proceedings of the 16th International Conference on Data Engineering (ICDE02)*, 2002.

[15] J. Wilkes. The Pantheon storage-system simulator. Technical Report HPL–SSP–95–14, Storage Systems Program, Hewlett-Packard Laboratories, Palo Alto, CA, December 1995.

[16] J. Zhang, A. Sivasubramaniam, H. Franke, N. Gautam, Y. Zhang, and S. Nagar. Synthesizing representative I/O workloads for TPC-H. In *Proc. of the 10th International Symposium on High Performance Computer Architecture (HPCA10)*, Feb 2004.