An Iterative Technique for Distilling a Workload's Important Performance Information

Zachary Kurmas
Georgia Tech
kurmasz@cc.gatech.edu

Kimberly Keeton
HP Labs
kkeeton@hpl.hp.com

Introduction

We are developing a method of automatically finding and extracting information from a workload trace that can be used to synthetically generate a workload with similar performance. We hypothesize that any set of information that can be used to generate a workload with similar performance can also be used to predict performance. Thus, we see this work as a step toward our long-term goal of predicting the performance of an I/O workload using performance-related information extracted from a trace of that workload.

The ability to predict performance has many uses. For example, it has the potential to greatly simplify the process of configuring large, complex disk arrays by allowing us to quickly evaluate the performance benefits of several possible configurations. Unfortunately, predicting performance of disk arrays is very difficult. The disk array cache, cache replacement policy, disk controllers, SCSI buses, RAID configuration, reorder logic, and data layout all interact with the workload and affect performance in ways that are not yet well understood. To help overcome these difficulties, we have defined an intermediate goal of learning how to automatically find and extract information from a workload trace that can be used to synthetically generate a workload with similar performance.

We believe that our biggest contribution will be the reduction of the human effort needed to identify a workload's performance related characteristics. Because every workload and storage system interact differently, the set of workload attributes that most affect performance will differ between workloads. A performance model will be of little practical use if many man-hours of calibration through trial-and-error are required for each new workload or storage system modeled. Our research emphasizes the development of methods of automatically identifying performance-related attributes instead of emphasizing the search for the particular performance-related attributes of a single workload. The application we are developing will identify the performance-related attributes of new workloads easily with little human intervention.

Description of Method

Our basic approach is to iteratively add attributes to a workload characterization until it can be used to generate a synthetic workload with similar performance. We use the term *attribute* to refer to the description of some property or characteristic of a workload that can be measured. For example, the mean request size, the distribution of interarrival times, and the mean run count are all attributes. We use the term *attribute-value* to refer to an attribute paired with its value for a specific workload; for example, a mean request size of 64KB is an attribute-value. We call a collection of attributes a *characterization configuration*; and we call the resulting set of attribute-values a *characterization*. Notice that a characterization configuration distinguishes between the many different possible characterizations of a workload. We classify attributes into *attribute groups* according to the request parameters (request size, location, read/write type, and (inter)arrival time) measured. For example, the mean request size measures only request sizes, while a distribution of locations of read requests measures both location and read/write type.

Given a workload, we are seeking a set of information (i.e. attribute-values) that can be used to generate a synthetic workload with similar performance. More specifically, we are seeking a set of attributes for which any two workloads with similar attribute-values have similar performance (on a given storage system). Our approach is to iteratively select attributes and add them to a characterization configuration of a workload under test until a synthetic workload with similar attribute-values has the similar performance. The challenge is to efficiently select useful attributes from infinitely many possibilities.

To address this challenge we break each iteration into three steps:

Evaluate the characterization configuration: We evaluate a characterization configuration using the method described by Ganger in [1]. We first use the current characterization configuration to characterize the observed workload. We then synthetically generate a workload based on that characterization. Next, we collect the cumulative distribution functions (CDF) of response time for each workload and com-

pare them using the root-mean-square (RMS) metric described in [4]. We consider the root-mean-square of these two CDFs a quantification of the "completeness" of the characterization configuration, with smaller RMS values indicating a more complete configuration.

Choosing an Attribute Group: If the evaluation detects deficiencies in the current characterization configuration, we must improve it, either by adding an attribute to the configuration or by refining an existing attribute.

Our biggest challenge is in choosing which attribute to add or improve during each iteration. It is not possible to examine each possible attribute; therefore, the main contribution of this research is our method of limiting the attributes under consideration during each iteration. We do this by estimating the maximum potential benefit of all attributes in a certain attribute group. We currently have two methods for doing this: We can remove the entire group of correlations (relationships between the I/O request parameters described by the attributes) from the observed workload, or we can add the "perfect" attribute to the current characterization configuration. We call the first approach the "subtractive approach" and we call the second approach the "additive" approach.

For each attribute group, we define a "perfect" attribute. The resulting attribute-value is simply a list of all the values of the parameters measured by this group. For example, the perfect arrival time attribute calls for a list of the arrival times of each I/O request. This list contains all possible relationships between the arrival (and interarrival) times of a workload. Thus, the perfect attribute provides all possible information about a set of request parameters to the workload generator. Of course, using the perfect attribute defeats the purpose of workload characterization; but, we will see how this concept is useful for evaluating an attribute group.

To see how the subtractive approach works, consider a synthetic workload that contains no correlations within arrival time (i.e., removes the "perfect" arrival time attribute), but is otherwise identical to the original, observed workload. If there is no difference in the response time distribution of these two workloads, we infer that our characterization configuration need not contain any arrival time attributes. In other words, we infer that our characterization need not contain any information about correlations between the workload's arrival time values.

To see how the additive approach works, suppose we take our current characterization configuration and add the "perfect" attribute for location. If there is a large change in performance, then we infer that our current attributes describing location are inadequate. On the other hand, if the performance does not change, then we infer that there will be no benefit to improving any of the location attributes because even the best possible attribute-value had little effect. Again, this indicates that our characterization need not contain any additional detailed information about the correlations between a workload's location values.

Implementing the Improvement: When we have determined which attribute group contains the missing information, we must add an appropriate attribute to our characterization configuration. We can automatically iterate through an attribute group's known attributes and choose the best one; however, we are still investigating methods of developing new attributes when necessary. In the interim, our intuition about storage systems and workloads will be necessary to direct the development of new attributes.

Current Status

Currently, we are developing an application that automates our method. This application has two basic tasks: To evaluate a set of attributes and to select the attribute group from which the next attribute will be chosen.

The application's first task is to evaluate a characterization configuration (i.e. a set of attributes). To do this, the application takes as input an observed workload and a characterization configuration. It first analyzes the observed workload to obtain the set of attribute-values corresponding to the attributes in the characterization configuration. Next, the application synthetically generates a workload with the same attribute-values, and issues that synthetic workload to the storage system. After obtaining performance information from the executed synthetic workload, the application measures the RMS difference between the response times of the original and synthetic workloads.

The application's second task is to choose an attribute group for improvement. To do this, the application must first generate several characterization configurations by applying the additive and subtractive methods to different attribute groups. It then evaluates these characterization configurations (as described above) and chooses an attribute group for improvement.

We are still developing the algorithms used to generate and compare the characterization configurations. Developing an algorithm for choosing an attribute group to improve is very difficult because this process is not as straightforward as the brief description above indicates. In practice, we have found that we cannot always choose an attribute group for improvement by simply applying the additive and subtractive approaches and comparing the resulting RMS values. In some cases, it was not possible to apply one of the approaches. In other cases, the results were misleading. Thus far, we have always been

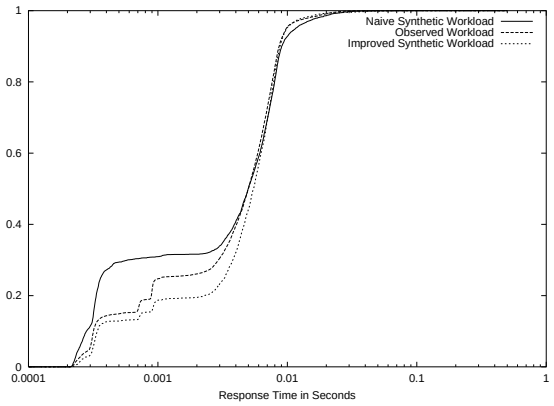

Figure 1: CDFs of read response times

able to work easily around these limitations; however, developing generally applicable techniques for identifying and addressing them has been more challenging.

We designed our application to facilitate the development of such algorithms. To implement a new algorithm, we simply override the *chooseAttributeGroup()* method. Presently, *chooseAttributeGroup()* simply sends the user e-mail requesting a list of characterization configurations. Our plan is to incrementally add functionality to *chooseAttributeGroup()* as we learn how to address the aforementioned limitations. This approach allows us to gradually reduce the amount of human effort needed to develop a performance characterization, instead of having no improvement until our algorithm is fully functional.

Currently, our biggest challenge is the development of new attributes. We learned much from Ganger's work [1]; in addition, we plan to leverage research in arrival and disk access patterns as we search for new attributes [2, 3]. However, we still need to add many more attributes to our collection before we can fully automate the distillation of performance characteristics.

Results

Thus far, we have applied three iterations of our method to a trace of HP's Open Mail e-mail server using an HP FC-60 disk array. Because the FC-60 has a 265MB write-back cache, all write requests are considered to be completed once the data has been placed in the cache; therefore, all writes have approximately the same response time. As a result, we focus on the response times of read requests only.

Figure 1 shows the improvement in our characterization configurations. The line labeled "Observed Workload" shows the cumulative distribution function (CDF) of response times of the read requests in the observed Open Mail workload. The Naive Synthetic Workload was specified by a characterization configuration containing distributions for each of the four request parameters. The workload was generated by simply choosing each request's parameter independently at random from these distributions. The RMS difference between the performance of the Original Workload and the Naive Synthetic Workload is .81.

We then applied three iterations of our method to develop a characterization configuration for the Improved Synthetic Workload. This characterization configuration is similar to that of the Naive Synthetic Workload; however, it contains separate distributions for read and write locations, and an attribute that describes locality. The result is a workload whose RMS difference from the observed workload is only .66.

Clearly, there is much room for improvement; however, the brief results here demonstrate the potential of our method for directing the improvement of characterization configurations.

Conclusion

In summary, we are learning how to identify and extract information from a workload trace that can be used to synthetically generate a workload with similar performance. This will both help direct the development of performance models and also improve our understanding of workload behavior.

Our most important result is not the characterization and synthesis of an individual workload, but the development of a process that can, with limited human intervention, learn how to characterize and synthesize the performance of any workload. The innovation of this process is the way in which we iteratively improve our characterizations and the way in which we direct the search for attributes in each iteration.

References

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

[2] 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.

[3] S. D. Gribble, G. S. Manku, D. Roselli, E. A. Brewer, T. J. Gibson, and E. L. Miller. Self-similarity in file systems. In *Proceedings SIGMETRICS*, pages 141–150, 1998.

[4] C. Ruemmler and J. Wilkes. A trace-driven analysis of disk working set sizes. Technical report, Hewlett-Packard Laboratories, 1993.