Automated Clustering-Based Workload Characterization

Odysseas I. Pentakalos
Code 930.5
NASA GSFC
Greenbelt MD 20771
odysseas@cesdis.gsfc.nasa.gov
301-286-4403

Daniel A. Menascé
Dept. of CS
George Mason University
Fairfax VA 22030
menasce@cs.gmu.edu

Yelena Yesha
Dept. of EE and CS
Univ. of Maryland
Baltimore County
Baltimore MD 21228
yeyesha@cs.umbc.edu

1. Introduction

The demands placed on the mass storage systems at various federal agencies and national laboratories are continuously increasing in intensity. This forces system managers to constantly monitor the system, evaluate the demand placed on it, and tune it appropriately using either heuristics based on experience or analytic models. Performance models require an accurate workload characterization. This can be a laborious and time consuming process. In previous studies [1,2], the authors used *k*-means clustering algorithms to characterize the workload imposed on a mass storage system. The result of the analysis was used as input to a performance prediction tool developed by the authors to carry out capacity planning studies of hierarchical mass storage systems [3]. It became evident from our experience that a tool is necessary to automate the workload characterization process.

This paper presents the design and discusses the implementation of a tool for workload characterization of mass storage systems. The main features of the tool discussed here are:

- *Automatic support for peak-period determination*: histograms of system activity are generated and presented to the user for peak-period determination.
- *Automatic clustering analysis*: the data collected from the mass storage system logs is clustered using clustering algorithms and tightness measures to limit the number of generated clusters.
- *Reporting of varied file statistics*: the tool computes several statistics on file sizes such as average, standard deviation, minimum, maximum, frequency, as well as average transfer time. These statistics are given on a per cluster basis.
- *Portability:* the tool can easily be used to characterize the workload in mass storage systems of different vendors. The user needs to specify through a simple log description language how the a specific log should be interpreted.

The rest of this paper is organized as follows. Section two presents basic concepts in workload characterization as they apply to mass storage systems. Section three describes clustering algorithms and tightness measures. The following section presents the

architecture of the tool. Section five presents some results of workload characterization using the tool. Finally, section six presents some concluding remarks.

2. Workload Characterization

One of the important steps in any capacity planning and performance modeling study is workload characterization. The purpose of this step is to understand the characteristics of the workload submitted to a system and determine a synthetic description—called workload model—of the global workload. To make these concepts more specific, let us turn our attention to a mass storage system subject to two types of requests: ftp gets and ftp puts. Imagine that the system is observed during a few hours of operation the following information about each request is gathered:

- type of request (get or put),
- request arrival time,
- size of the file involved in the request, and
- time at which the file transfer completed.

If the system is sufficiently busy during the observation period you may collect thousands of such tuples. The question is what to do with this information? To use this information in a predictive performance model, one needs a more compact representation than a list with thousands of entries, one per request. If one looks at all requests, we may find that one can aggregate them into a reasonably small number of groups of "similar" requests. The notion of similarity is formalized in the next section. In this section we consider an intuitive meaning to the term. Within each group, each request is characterized by a pair (Z, S) where Z is the time since the last arrival of a request of the same type (get or put) and S is the file size. Suppose that one draws a scatter plot, such as the one in figure 1, where the x axis represents values of Z and the y axis represents values of S. As one can see, there is a natural grouping or clustering of points that have similar values of Z and S. Each *cluster* can then be represented by the coordinates of its center, called the *centroid*.

In the case of the example of figure 1, the centroids are: (Z1= 2.48 sec, S1= 4.26 MB), (Z2= 4.95 sec, S2= 107 MB), and (Z3= 13.76 sec, S2= 45.2 MB). The other information we obtain from the clustering exercise shown in figure 1 is that 41.2% of requests fall into cluster 1, 26.8% fall into cluster 2, and 41.2% fall into cluster 3. One can now drop all measurements and work with the more compact representation of the workload provided by the three clusters.

Another important aspect in workload characterization is the determination of the interval during which measurements are obtained. For capacity planning studies and system sizing, one usually looks for the periods of time when the system is more heavily utilized, or the *peak period*. This is usually obtained by looking at histograms of system activity, for example number of requests submitted or number of bytes transferred, during each hour of the day for many days. The peak period is the time interval or sets of time intervals during which the load on the system is high compared to other intervals. Consider figure 2 that shows a histogram of number of get requests submitted to a mass storage system during a period of one day. As one can see, the peak period is between 9AM and 6PM. Thus, this is the period during which measurements should be collected.

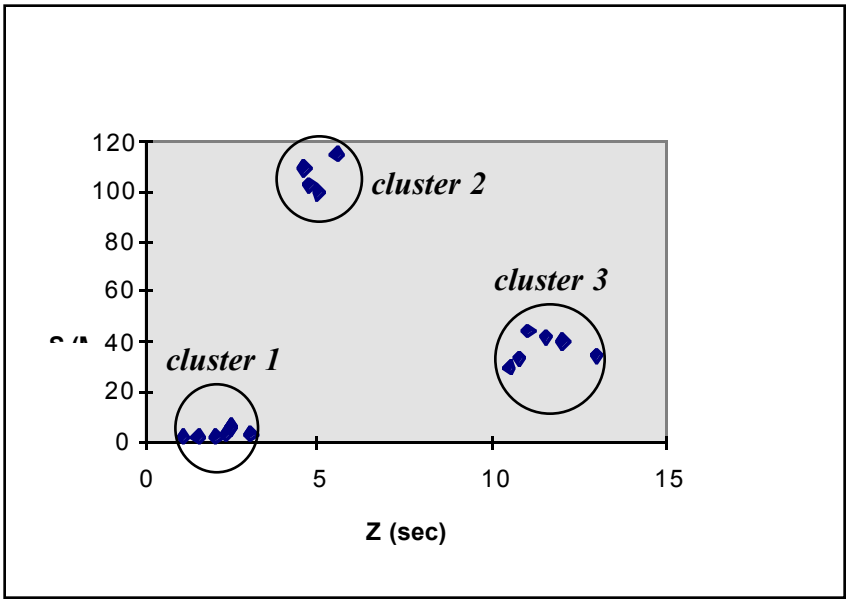

Figure 1 - S versus Z scatter plot.

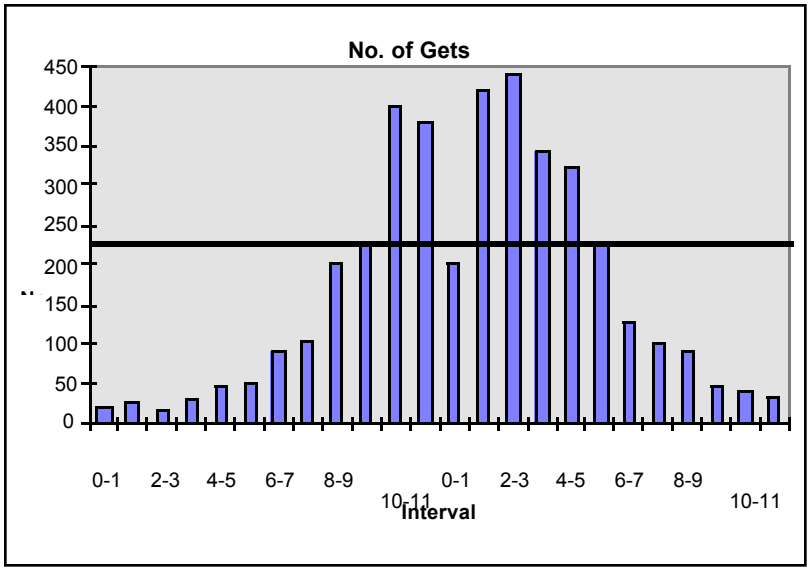

Figure 2 - Histogram for determination of peak period.

So, in summary, workload characterization is composed of the following steps:

1. Determine the basic type of requests (e.g., ftp gets and ftp puts).
2. Collect measurements on system activity for each type of request over a period of several days and plot a histogram to determine peak periods for each type of request.
3. Collect the measurements needed to characterize the workload during the peak period (e.g., measure file sizes, inter-arrival times for requests).
4. Cluster the measurements obtained into a small number of groups or clusters using a clustering algorithm (see next section).

The process described above can be quite laborious and time consuming. The purpose of the tool described in this paper is to automate the whole process for mass storage systems. The next section discusses in detail the algorithms used to perform clustering analysis and the criteria used to determine the number of clusters to use in workload characterization.

3. Clustering Algorithms

During the process of workload characterization of a mass storage system, logs of the requests which arrive at the system to be processed are analyzed. The objective of clustering is to classify the individual requests into a relatively small number of classes which impose on the average a load on the system similar to that of the actual workload. This classification is made based on a measure of similarity or proximity between the requests. A log with *n* data points consisting of *d* components each can be described as a set of vectors $\vec{x}_i = (x_{i,1}, \ldots, x_{i,d})$ for *i*=1,...,*n*, where $x_{i,k}$ is the *k*-th feature of the *i*-th data point in the log. The proximity between data points is described most commonly in terms of an *n x n* matrix, called the *proximity matrix* where entry $d_{i,j} = d(i,j)$ is a distance metric (or dissimilarity measure) between the *i*-th and *j*-th data point. The three most commonly used distance metrics are: the Euclidean distance $d(i,j) = \sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2$, the Manhattan distance $d(i,j) = \sum_{k=1}^{d} \left| x_{i,k} - x_{j,k} \right|$, and the sup distance $d(i,j) = \max_{i \le k \le d} \left| x_{i,k} - x_{j,k} \right|$.

A number of approaches have been proposed in the literature for clustering data and those clustering approaches are themselves classified using various characteristics. A clustering algorithm is *exclusive* or *nonexclusive* based on whether the data points are allowed to belong to only one or more than one cluster, respectively. An exclusive clustering algorithm is *intrinsic* or *extrinsic* based on whether the classification is done based on only the proximity matrix or based on the proximity matrix and category labels assigned to the data points. In extrinsic clustering the objective is to determine the discriminant surface which separates the points based on their categories. An exclusive, intrinsic clustering algorithm is *hierarchical* or *partitional* based on whether the resulting classification is a nested sequence of partitions or a single partitioning. Finally, an exclusive, intrinsic algorithm is *agglomerative* or *divisive* based on whether the algorithm proceeds by gradually merging clusters into larger and larger classes or by subdividing larger clusters into smaller ones.

The specific algorithm used in our tool can now be described using the above terminology as an exclusive, intrinsic, partitional, agglomerative algorithm. The problem of exclusive, intrinsic, partitional clustering can be stated as given *n* points and a desired fixed number of clusters *K*, select the partition of those points into clusters such that points in one cluster are more similar to points in their cluster than to points in the other clusters. The number *S (n,K)* of ways to partition *n* points into *K* clusters is given by:

$$S(n,k) = \frac{1}{K!} \sum_{i=1}^{K} (-1)^{K-i} \binom{K}{i} i^n$$

Therefore, an exhaustive evaluation of all possible partitions is not feasible. The centroid of cluster C_k is given by:

$$\vec{m}_k = \frac{1}{|C_k|} \sum_{\vec{x} \in C_k} \vec{x}$$

The within-cluster variation e_k for cluster C_k is the average distance of all points in the cluster to its centroid. Thus,

$$e_k = \frac{1}{|C_k|} \sum_{\vec{x} \in C_k} d(\vec{x}, \vec{m}_k)$$

and the tightness E_K of a particular clustering is defined as the average of the within-cluster variation normalized by the maximum value of all within-cluster variations. Hence,

$$E_K = \frac{1}{K \max_{j=1}^{K} \{e_j\}} \sum_{k=1}^{K} e_k$$

Note that both e_k and E_K are numbers between 0 and 1. The closer to zero the value of the tightness, the better the clustering quality is. Typically, an iterative algorithm is used to minimize the global metric E_K. One disadvantage of iterative algorithms is that occasionally they terminate at a local minimum rather than at the desired global minimum. In practice, one can get more confidence at the quality of the solution by repeating the algorithm with various different starting partitions and ensuring that the algorithm terminates at the same solution. The *K*-means algorithm is one such agglomerative, iterative algorithm. It is defined as follows:

1. Start with an initial assignment of points to *K* clusters.
2. Compute the centroid $\vec{m}_k$ of each cluster C_k for *k*= 1, ...,*K*.
3. For each point $\vec{x}$ in the collection do:
   $j = \min_{1 \le k \le K} \{d(\vec{x}, \vec{m}_k)\}$ /* find the cluster closest to point $\vec{x}$ */
   let i be such that $\vec{x} \in C_i$
   if $i \neq j$ then
      $C_j = C_j \cup \{\vec{x}\}$. /* add $\vec{x}$ to the *j*-the cluster if not already there */
      $C_i = C_i - \{\vec{x}\}$ /* remove $\vec{x}$ from the *i*-th cluster */
   end if
4. Recompute the centroid $\vec{m}_j$ of each cluster C_j for $j = 1,..,K$.
5. Repeat steps 2 through 4 until no point changes its cluster assignment during a complete pass or a maximum number of passes is performed.

A number of variations exist for the *K*-means algorithm based on how the initial cluster centroids are selected and on whether the cluster is recomputed after each re-assignment of a point or after an entire pass through the points has completed as described above [4,5,6,7]. The version of the algorithm used in our tool is the one proposed by Forgy[8].

4. Architecture of the Tool

Figure 3 shows the architecture of the workload characterization tool. Modules are shown in ovals and internal databases are shown in parallel solid lines. The figure also shows the Hierarchical Mass Storage System (HMSS) log and a file containing a description of the log format as the two input files to the tool. Users interact with the tool through a Graphical User Interface (GUI). Through this interface they can request the tool to open a specific log, generate histograms for peak period analysis, do workload characterization through clustering, and view file access statistics and the results of the workload characterization process.

The *Filter* process reads the log descriptor and reads the HMSS log and stores the information in a Measurements DB. This feature of the tool allows it to be used to characterize workloads for virtually any HMSS provided one can specify the log format using the log descriptor. Once the information is entered into the DB it can be used as input to the *Histogram Generator* and *Clustering Engine* modules. The *Histogram Generator* plots a histogram of system activity for a selected day and time and a selected type of request (get or put). The *Clustering Engine* reads the data points from the Measurements DB, runs a *K*-means clustering algorithm, and saves the data into a Workload DB. This module also generates tightness measures to illustrate the quality of the clustering generated. The *Report Generator* module reads the workload model from the Workload DB and generates a workload description in the format expected by Pythia—a tool for performance prediction of mass storage systems developed by the authors [3]. The *Report Generator* also stores that information in a format needed by the *Results Manager* for presentation of file access statistics and workload model characteristics to the users of the workload characterization tool.

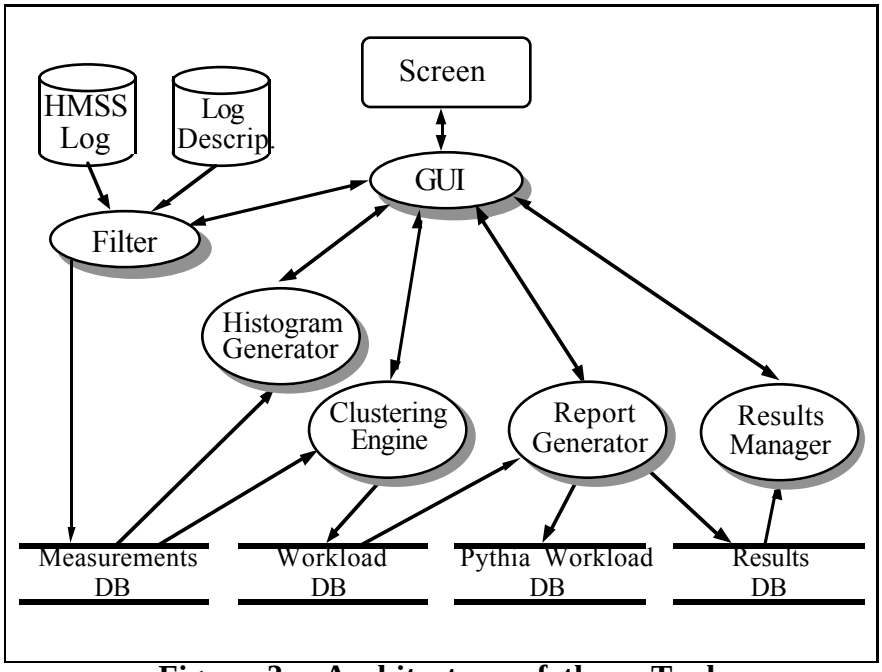

Figure 3 - Architecture of the e Tool.

5. Using the Tool

This section describes the operation of the tool by going through a sample session of workload characterization. Figure 4 shows the main screen of the tool. The menu bar has three options. The File menu provides options for opening a log of data, saving the results of the workload characterization, and quitting from the tool. The View menu provides options for generating a histogram, viewing the tightness measure as a function of the number of clusters, and generating a workload characterization. Finally, the Help menu provides help on the use of the tool. The Data Range window describes the range of the data values contained in the currently selected log file. Initially, the range will be empty, but as soon as a log file is opened from the File menu option, the entries will be filled automatically to describe the range. The user may change the range under consideration at any time.

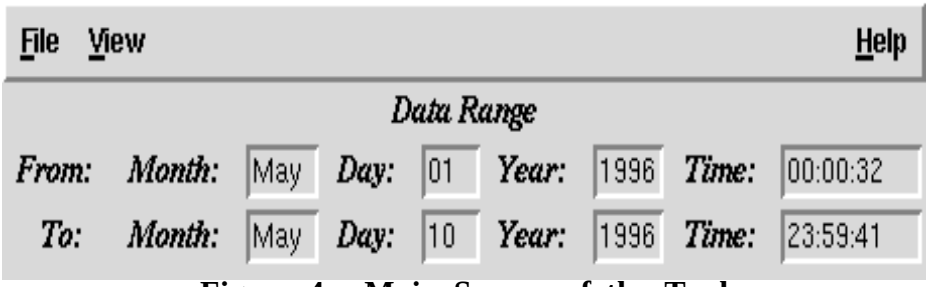

Figure 4 - Main Screen of the Tool

The next step after opening a log file is to view a histogram of the data so that the peak period for this workload can be detected. This is done by selecting the Histogram menu option from the View menu. Figure 5 shows the window that comes up for selecting which workload (get or put) should be considered in the histogram. Figure 6 shows a sample histogram for get requests. The results shown are for the first day in the data range when the histogram computation was executed. The user may modify the data range and run the histogram again. Pressing the "Previous Day" button, loads the data for the previous date and displays the histogram, and pressing the "Next Day"button loads the data for the next date. Pressing "OK" closes the histogram window. Browsing through the histograms for a number of days allows one to select the peak period during the day.

Figure 5 - Histogram Configuration Window

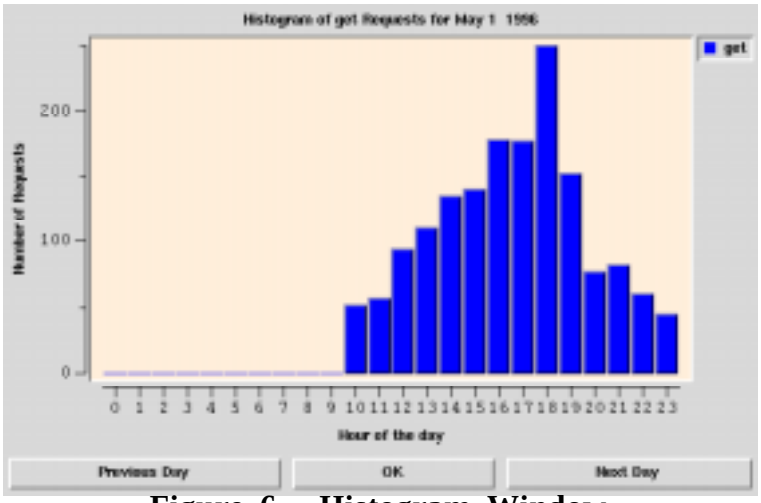

Figure 6 - Histogram Window

Once the peak period has been selected, the data range can be modified and the user can then select the "Tightness" option from the View menu. Figure 7 shows the window which appears for this option Again, it allows the user to select the type of workload to be analyzed and also the range of clusters to be evaluated. Figure 8 shows the sample plot for a cluster range of two through ten for get requests. This plot is very helpful in determining the most appropriate value for the number of clusters. It allows one to visually select a local minimum for the tightness measure which compromises between a fairly accurate workload with as small a number of clusters as possible.

Figure 7 - Tightness Configuration Window

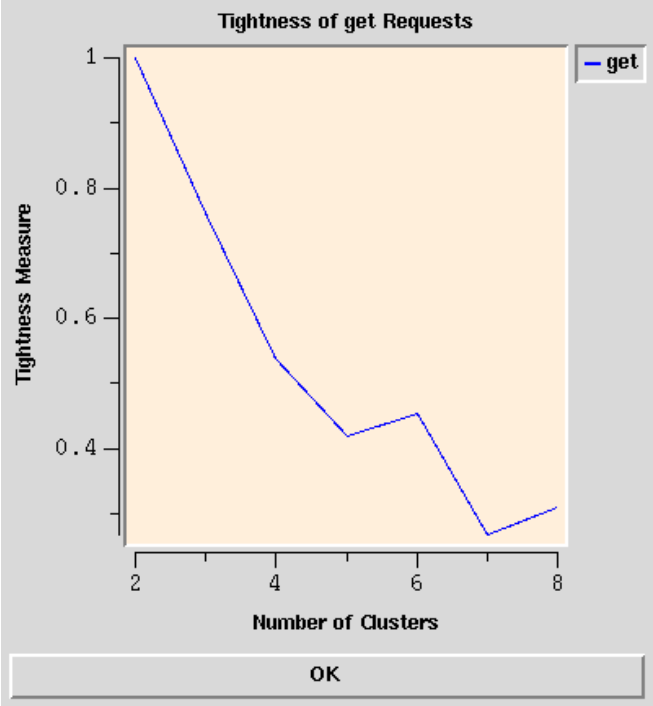

Figure 8 - Tightness Variation Plot

The final step in the workload characterization is to determine the cluster centroids, and cluster sizes, for a given number of clusters. This is done by selecting the "Workload" option from the View menu. The Workload Configuration window comes up, shown in figure 9, which allows one to select the workload type, and number of clusters. Once the workload has been computed the Workload window pops-up which describes the workload parameters as shown in figure 10. The workload window includes information such as the name of the log file, the number of clusters selected, and for each cluster, describes the centroid, the number of points from the log which belong to the cluster, and the frequency.

Figure 9 - Workload Configuration Window

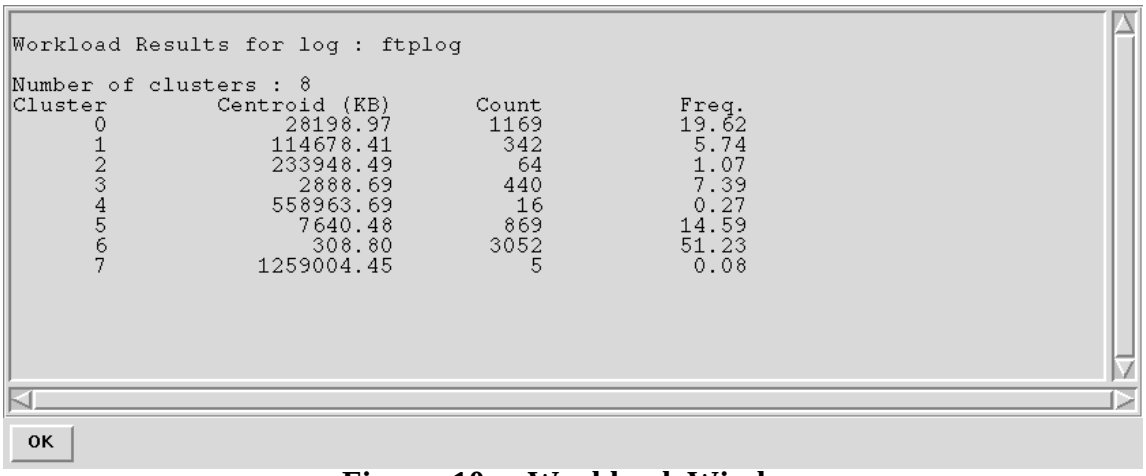

Figure 10 - Workload Window

6. Concluding Remarks

This paper described the design and operation of a tool for automating the workload characterization of mass storage system workloads. The tool is based on a variation of the *K*-means clustering algorithm. In addition to characterization of the workload, the tool allows one to determine the peak-period of system usage by providing a browsing capability through histograms of workload requests, and also simplifies the task of selecting the number of clusters present in the workload by plotting a measure of the accuracy of the clustering as a function of the number of clusters.

Bibliography

[1] Daniel A. Menascé, Odysseas I. Pentakalos, and Yelena Yesha, "An Analytic Model of Hierarchical Mass Storage Systems with Network-Attached Storage Devices," Proc. of the ACM SIGMETRICS'96 Conference Philadelphia, PA, May 23-26 1996.

[2] Odysseas I. Pentakalos, Daniel A. Menascé, Milt Halem, and Yelena Yesha, "An Approximate Performance Model of a Unitree Mass Storage System," 14th IEEE Symposium on Mass Storage Systems, Monterey, California, September 1995, pp. 210--224.

[3] Odysseas I. Pentakalos, Daniel A. Menascé, and Yelena Yesha, "An Object-Oriented Performance Analyzer of Hierarchical Mass Storage Systems," submitted to the 1996 Computer Measurement Group Conference, San Diego, CA, December 1996.

[4] Michael R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York, NY, 1973.

[5] Anil K. Jain and Richard C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[6] Leonard Kaufman and Peter J. Rousseeuw, *Finding Groups in Data*, John Wiley & Sons, Inc., New York, NY, 1990.

[7] Daniel A. Menascé, Virgilio A. F. Almeida, and Larry W. Dowdy, *Capacity Planning and Performance Modeling: from mainframes to client-server systems,"* Prentice-Hall, Englewood Cliffs, NJ, 1994.

[8] E. Forgy, "Cluster Analysis of multivariate data: efficiency versus interpretability of classifications", Biometrics, 21, 768, 1965.