Characterizing Computer Systems' Workloads

Said Elnaffar and Pat Martin
{elnaffar, martin}@cs.queensu.ca

Technical Report 2002-461
School of Computing, Queen's University
Kingston, Ontario, Canada K7L 3N6

December 2002

Abstract

The performance of any system cannot be determined without knowing the workload, that is, the set of requests presented to the system. Workload characterization is the process by which we produce models that are capable of describing and reproducing the behavior of a workload. Such models are imperative to any performance related studies such as capacity planning, workload balancing, performance prediction and system tuning. In this paper, we survey workload characterization techniques used for several types of computer systems. We identify significant issues and concerns encountered during the characterization process and propose an augmented methodology for workload characterization as a framework. We believe that the surveyed case studies, the described characterization techniques, and the proposed framework give a good introduction to the topic, assist in exploring the different options of characterization tools that can be adopted, and provide general guidelines for deriving a good workload model suitable as an input to performance studies.

1 INTRODUCTION

The performance evaluation of computer systems requires understanding of a system's workload. As shown in Figure 1, the workload is a set of requests, or components, that place different demands on various system resources. Workload characterization provides a model of a system's workload by means of quantitive parameters and functions. The model should be representative, compact, and

---

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

CONTENTS

accurate [Calzarossa and Serazzi 1993], and should be able to describe and reproduce the dynamic behavior of the workload and its most essential static features.

Workload characterization dates back to the 1970's, when computers were mainframes and their workloads consisted of transactions and batch jobs. The continuous evolution of computer architectures has pushed the discipline to evolve accordingly. The advent of networks and time sharing systems, along with the increased processing power of computers and the growth of graphical user interfaces, have changed the way users deal with the system and introduced new processing requirements. Furthermore, the Internet and its numerous applications have multimedia workloads. These workloads are very complex because they consist of a mix of different types of applications such as audio/video conferencing, text/voice chat, file transfer, and telephony, which are characterized by different performance demands on the system resources.

Workload characterization is a requirement for many performance studies such as scheduling, capacity planning [Menascé et al. 1994], workload balancing, ensuring system scalability [Jain 1991], system tuning and configuration [Ferrari et al. 1983], performance prediction, and the construction of benchmarking suites. A benchmark suite, in particular, is the intuitive result of characterizing the

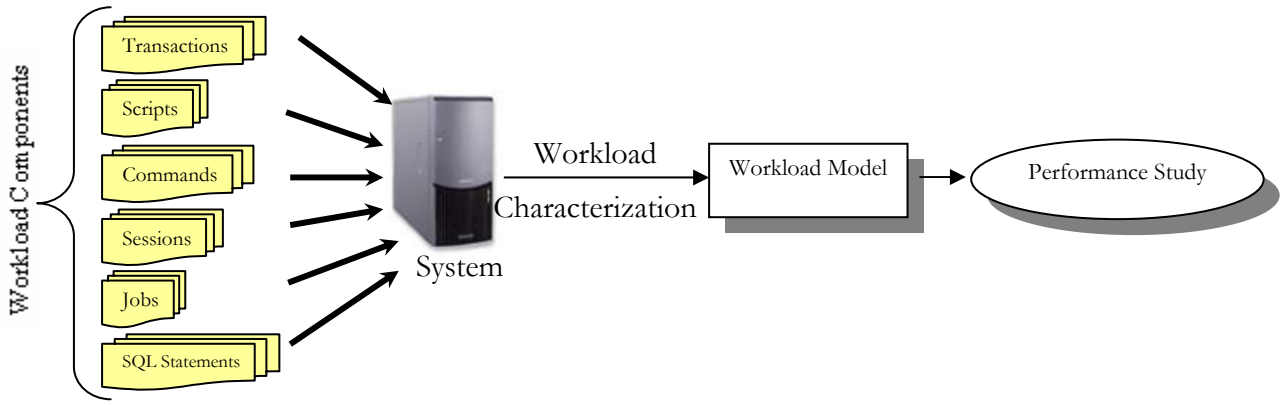

Figure 1. *The workload consists of components submitted to the system. The workload characterization process derives a workload model which can be used in further performance related studies.*

system's workload. It can be viewed as a synthesized, controlled workload, which can be ported to different environments in order to evaluate the system's performance in that environment.

The design and evaluation of resource management policies, such as caching policies in the World Wide Web (WWW) or disk storage layout for database servers require the knowledge of the characteristics and the behavior of the requests to be processed. In general, knowing what kind of workload a proposed system has to process should assist in its design. For example, the design of a system whose workload consists of batch jobs is different from the design of a system whose workload consists of concurrent interactive requests. Workload characterization helps determine how much computing power is needed and assists in identifying the relationship between the workload and the Quality of Service (QoS). Workload balancing is another terrain where knowledge of the intrinsic characteristics of the workload is essential for distributing the requests to different servers in order to optimize the overall performance of the system [Nikolaou et al. 1998]. Characterizing the workloads of today's web-based systems helps with improving system designs, recommending similar web pages, reducing latency, and understanding user reaction and motivation [Zaïane et al. 1998].

Over the years, workload characterization has addressed all kinds of new application domains. As a result, the characterization techniques have evolved to cope with the more complex workloads. In

the human computer interaction field, for example, contemporary graphical user interfaces are smart enough to dynamically customize themselves to suit a user's needs. Such smart interfaces are the results of characterizing the user's workload by analyzing the run-time behavior and the access pattern of each individual user [Hudson and Smith 1997].

Calzarossa and Serazzi examined a number of workload characterization case studies [Calzarossa and Serazzi 1993]. In this paper, we revisit some of those case studies to emphasize the techniques used in them and then present more recent case studies, including ones from application domains such as the WWW and client/server systems. Ultimately, our study aims to achieve three main objectives. First, we survey case studies across different types of computer systems and enumerate the most common techniques used to characterize workload, such as graph-based techniques, stochastic processes, clustering, and numerical fitting. We give a brief description of these techniques and classify them according to their ability to extract different aspects of the workload, that is, the static properties or the dynamic behavior. Second, we organize these techniques within a common framework. To this end, we propose a general methodology for workload characterization. Our third aim is to point out the potential problems and concerns that may be encountered during the characterization process.

The rest of the paper is organized as follows. Section 2 gives a brief description of the common techniques used in workload characterization. Section 3 examines workload characterization case studies in different types of computer systems, namely batch and interactive systems, client/server systems, databases systems, parallel systems, and WWW systems. Section 4 explains our workload characterization framework and highlights the most significant concerns and potential problems that researchers encounter during the characterization process. Section 5 presents our conclusions, and Section 6 outlines future directions for research and envisions future systems in the context of workload characterization.

2 CHARACTERIZATION TECHNIQUES

In this section, we briefly describe the techniques most commonly used to analyze system workloads. The selection of a particular technique depends mainly on the purpose of the performance study, and on the level of detail required. It might be necessary, in some cases, to evaluate more than one technique in order to select the best one.

Functionally, we can classify the characterization techniques into two main categories: *static* and *dynamic*. Static techniques explore the intrinsic characteristics of the workload, such as transaction classes, the correlation between workload parameters and component dispersion, which do not change over time. Examples of these techniques are clustering, principal component analysis, averaging, and correlations. Dynamic techniques, such as Markov models, user behavior graphs, and regression methods, focus on describing the behavior of the workload and the way it fluctuates over time. These techniques usually analyze the historical data of the workload and, as a result, aid in forecasting its behavior in the future.

Throughout the workload characterization process, adopting one technique is usually not sufficient to obtain a complete analysis; several techniques may be used in combination in order to come up with an approach that satisfies the research needs. For example, clustering techniques might be used to classify the transactions submitted to the system. Afterwards, each class may become a node in User Behavior Graphs [Calzarossa and Serazzi 1994], or a transitional state in a Markov model. This example raises another issue, namely the importance of obtaining both static and dynamic properties of the workload in order to obtain a complete picture.

Visualization tools, such as graphs, histograms, and fitting curves, are a key means of highlighting significant features in the workload under investigation while simple techniques, like averages, may

smooth out some details such as *burstiness*. Sections 2.1 and 2.2 describe static and dynamic characterization techniques respectively. Table 1 summarizes the techniques examined in these sections.

Technique Type	***Technique***	***Advantages***	***Disadvantages***
Static	Descriptive Statistics (average, variance/standard deviation, correlations, distributions)	o Provides preliminary description o Easy to calculate	o May not be sufficient; further analysis is needed
	Single-parameter Histogram	o Expressive visual means o Shows frequencies of each bin o Frequency distribution can be used in simulation models	o Incapable of expressing the correlation among different parameters
	Multi-parameter Histogram	o Illustrates the correlation between different parameters o Expressive visual means	o Difficult to plot the correlation between more than two parameters
	Factor Analysis (e.g., Principal Component Analysis)	o Simplifies performance data and reduces their dimensionality	o Complex to calculate
	Clustering	o Identifies homogeneous classes of workload components based on certain criteria	o Difficult to choose the appropriate number of clusters
Dynamic	Markov Models (Markov chains, Markov processes)	o Predicts the order in which the requests are executed	o Complex to calculate
	Prediction Using Neural Networks	o Performs short-term and long-term forecasting of workload parameter values	o Difficult to design and to configure
	Moving Average	o Useful for short-term, single value prediction o Easy to calculate	o Cannot perform long-term forecasting o Cannot predicate more than one single value o No special consideration for the most recent observations o Difficult to determine the best number of observations
	Exponential Smoothing	o Useful for short-term, single-value forecasting o Places more weight on the most recent observations o Easy to calculate	o Cannot perform long-term forecasting o Cannot predict more than one single value o Difficult to determine the best smoothing weight
	Regression Methods (linear and non-linear fitting)	o Predicts the value of a parameter as a function of others o Identifies trends	o Can be complex to calculate
	User Behavior Graphs	o Used mostly in interactive systems o Describes the user's probable transition to a particular command/transaction type	o Requires clustering to compose the nodes
	Probabilistic Attributed Context Free Grammar	o Used in hierarchical systems (e.g., client/server) o Translates views of higher layers to lower layers	o Cannot be used to map lower layers to higher ones

Table 1. *Static and dynamic workload characterization techniques.*

2.1 STATIC TECHNIQUES

Static techniques, such as descriptive statistics, single-parameter histogram, multi-parameter histograms, principal component analysis, and clustering, help explore the static characteristics of the workload. In this section we give a brief description of each type.

Descriptive Statistics. Parametric descriptive statistical techniques are used to identify the static properties of the workload. Using these techniques helps describe what the workload parameters look like: where their center (average) is, how broadly they are spread (dispersion or variance), and how they are related to each other (correlation).

Averaging, or *arithmetic mean*, is the simplest method to characterize a workload parameter such as user think time, number of active users, number of I/O operations required to execute a query, or inter-arrival time of transactions. Averaging presents a single number that summarizes the parameter values observed. However, it is not always appropriate to count on arithmetic mean; the *median*, *mode*, *geometric mean*, or *harmonic mean* should be used in some cases.

The average alone is not adequate if the performance data has high variability. Variability is usually specified by the *variance*. However, the *standard deviation*, which is the square root of the variance, is more useful in expressing the variability because it has the same unit as the mean. The ratio of the standard deviation to the mean is called the *coefficient of variance* (C.O.V). A zero C.O.V. indicates that the measured parameter is constant. In this case, the mean gives the same information as the complete set. A high C.O.V. indicates high variance, in which case it may be useful to look at the complete histogram (discussed below). There are also other alternatives for specifying variability like *range* (minimum and maximum), *10^{th}- and 90^{th}- percentile*, *semi-interquartile range*, and the *mean absolute deviation*.

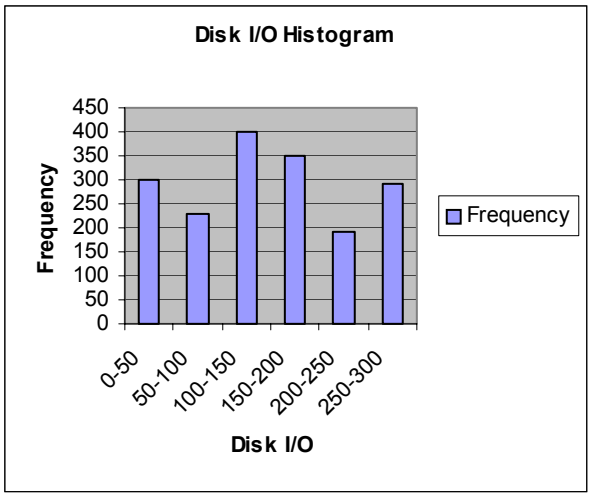

Figure 2. *A simple histogram which shows the frequency distribution of disk accesses of jobs.*

Correlation is another useful statistical technique that helps discover the relationship between different workload parameters. It is a decimal fraction, called *correlation coefficient*, which indicates the degree to which the parameters are related. There are numerous ways (e.g., Biserial, Point Biserial, Tetrachoric, Spearman rank-order, etc.) to calculate the coefficient of correlation. *Pearsonian product moment*, commonly called *Pearsonian r*, is the most popular one [Schiff 1995].

Single-parameter Histograms. A histogram is a visual representation of a parameter where the range of values is divided into intervals called *bins*. As shown in Figure 2, the histogram displays the frequency of the observations of each bin. This frequency distribution is used in simulation models to generate a test workload. However, one of the drawbacks with a histogram is that it is incapable of expressing the correlation among different parameters. Therefore, multi-parameter histograms can be used instead.

Multi-parameter Histograms. Multi-parameter histograms illustrate the correlation between different workload parameters. The distribution of *n* workload parameters can be described by an *n*-dimensional matrix or histogram. Figure 3 shows an example of a two-parameter histogram that represents the number of read and written pages in a database system. Each dot in the figure

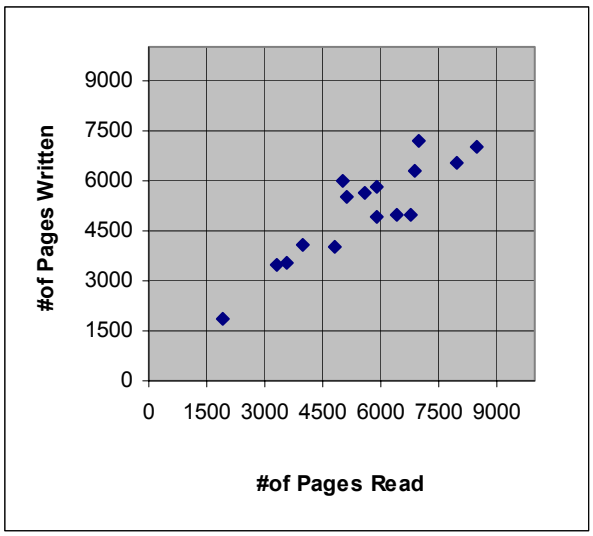

Figure 3. *A two-parameter histogram showing the correlation between two parameters. The number of dots in a square represents the number of nodes that read and wrote pages in the range corresponding to the cell.*

represents a system node. The number of dots in a cell of the grid represents the number of nodes that read and wrote pages in the range corresponding to the cell. As can be seen, the nodes reading a large number of pages are also the ones that write a large number of pages. Therefore, a significant correlation may exist between the two parameters. On the other hand, we should note that it is difficult to plot multi-parameter histograms that correlate more than two parameters.

Principal Component Analysis. The term *factor analysis* usually refers to statistical techniques that describe multidimensional sets of data by means of geometric representation. Their goal is to help choose a subspace of the variable space such that the projection of the data set on that subspace preserves as much information of the original set as possible. Consequently, factor analysis is beneficial for simplifying data and reducing their dimensionality.

Principal Component Analysis (PCA) [Harman 1976; Kline 1994] is a factor analysis technique that maps a set of parameters, or variables, into another set, called principal components, characterized by orthogonality among the components and by linear dependence on the parameters in the original set. PCA is an iterative process in which the first component is chosen such that it maximizes the

variance of the linear function expressing the dependence of the transformed parameters on the original ones. The second component is chosen such that it maximizes the remaining variance while this component must be orthogonal to the first, and so on.

Clustering. Clustering is one of the most widely adopted techniques in workload characterization (e.g., [Calzarossa and Serazzi 1994; Pentakalos and Menascé 1996; Nikolaou et al. 1998; Elms 1980]). Clustering identifies homogeneous groups, or classes, of workload components, based on the similarity of resource demands. In general, clustering methods can be classified as hierarchical or non-hierarchical. Hierarchical techniques, like the *Minimal Spanning Tree (MST)* [Rohlf 1973] method, start by assuming that each component of a workload is a cluster. Then, the two clusters with the minimum distance are merged to form a single cluster. The process iteratively continues until either all the workload components are grouped into a single cluster or the desired number of clusters is reached. On the other hand, the non-hierarchical techniques, like the *k-means* algorithm [Hartigan and Wong 1979], start from an initial partition that consists of the exact desired number of clusters. Workload components are reassigned among clusters so that a particular cluster criterion, known as *distance function*, is optimized.

Deciding about the number of clusters is a common problem in any cluster analysis study. Generally, it depends on the goal of the study and it is desirable to keep this number small for practicality. Various clustering algorithms are available in the literature [Jain et al. 1999].

2.2 DYNAMIC TECHNIQUES

Next, we examine techniques commonly used to describe and predict the behavior of the dynamic aspects of the workload.

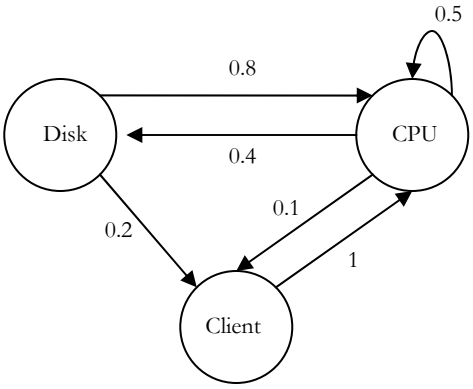

Figure 4. *A state transition diagram representing a Markov model.*

Markov Models. Knowing the number of requests of each type, or class, is not sufficient. It is also important to know the order in which requests are executed in the system. If it is assumed that the next request depends only on the current one, then the requests follow a *Markov model* [Howard 1960]. This model can be represented by a *transition matrix*, which gives the probability of moving to the next state given the current one. A corresponding *state transition diagram* can be easily constructed from the transition matrix. Figure 4 shows an example of a transition diagram in which the probability of a job using the disk after visiting the CPU is 0.4, the probability of it returning to the CPU from the disk is 0.8, and so on.

Markov models are used to describe the transitions between any system states, not just between system resources. For example, in a software development environment that provides several types of software tools, we can use a transition matrix to describe the probability of transitions between the different types of development tools like editors, compilers, linkers, and debuggers.

Prediction Using Neural Networks. Although getting a perfect prediction is a very hard problem, neural networks can be used to obtain reasonably good predictions in some cases [Mehrotra et al. 1997]. *Feedforward* as well as *recurrent* networks are commonly used for this purpose. The prediction problem can be viewed as a function approximation problem, in which the function values are represented as *time series*, that is, a sequence of values measured over time. Based on the knowledge

of the most recent *d* values of a time series, the neural network can be trained to predict the *d*+1 future value. The accuracy of predicting the values of a parameter may increase if a multivariate time series and the correlations among all workload parameters are taken into account [Menascé et al. 1994].

Typically, two types of predictions are considered: short-term, or *one-lag,* and long-term, or *multi-lag*, predictions. In one-lag predictions, the forecasting of the future value is based just on the past actual values. Multi-lag prediction also exploits some of the predicted values in order to predict future values. An example of multi-lag prediction is forecasting the value of a time series a year from today while the values for the next eleven months are unknown.

Moving Average. This is a simple prediction technique in which the next forecasted value is the average of the previous ones. This method shows very good results if the data is almost stationary, that is, with little variation [Letmanyi 1985]. However, it is not suitable for long-term prediction as it is not capable of predicting more than a single value at a time. The forecasted value can be calculated as follows:

$$f_{t+1} = \frac{x_t + x_{t-1} + \ldots + x_{t-n+1}}{n}$$

where f_{t+1} is the forecast value for period *t*+1, x_t is the actual value at time *t*, and *n* is the number of previous observations. It is not always easy to determine the number of periods, *n*, that should be used. Thus, different values of *n* may be examined in order to find the one that achieves the least *mean squared error* (MSE), which is calculated as follows:

$$MSE = \frac{\sum_{t=1}^{n} (x_t - f_t)^2}{n}$$

Exponential Smoothing. Exponential smoothing is similar to the moving average described earlier in terms of using the average to predict the next value. It is particularly useful for short-term forecasting and when the data is stationary. However, it differs from the moving average in the way it calculates the forecast value; it puts more weight on the most recent historical observations. The idea stems from the hypothesis that the latest observations give a better indication of the future. Here, the forecast value f_{t+1} is calculated as follows:

$$f_{t+1} = f_t + \alpha(x_t - f_t)$$

where α is the smoothing weight ($0<\alpha<1$). Again, some values of α are better than others in terms of getting the least MSE, and additional tests help to choose a suitable one.

Regression Methods. The value of a variable, called the *dependent* variable, can be predicated as a function of other variables, called *independent* variables, using regression models. Many mathematical forms exist, which describe the relationship between these variables. A linear relationship is a common assumption used to estimate the values of the dependent variable [Menascé et al. 1994].

User Behavior Graphs. User Behavior Graphs (UBG) are considered as the basis for several workload models [Calzarossa et al. 1990; Calzarossa and Ferrari 1986]. They are similar to the state transition diagrams used in Markov models and are commonly used to describe the workload of interactive systems, such that each user has her own UBG [Ferrari 1984]. A UBG is a probabilistic graph whose nodes represent the different command types issued by the user, and whose arcs represent the transition from one command type to another throughout a user session.

Probabilistic Attributed Context Free Grammar. A Probabilistic Attributed Context Free Grammar (PACFG) [Fu 1974] is a central means of constructing generative workload models, especially in systems that have a hierarchical nature, like client/server and WWW environments [Kotsis et al. 1997; Raghavan et al. 1994]. A PACFG can translate views between the different layers

of the system hierarchy. For example, a PACFG can map the client-oriented view of the workload, such as commands submitted during user sessions, to a low-level system view like TCP/IP protocol requests.

A PACFG is a 3-tuple $G_A = \{G, A, Q\}$ where G is the regular grammar defined as $G = \{V_N, V_T, P, S\}$. V_N and V_T are a set of non-terminal and terminal symbols, respectively, P represents a set of production rules, S is the start symbol, A is a set of attributes and Q is a set of probabilities associated with P. At each layer in the hierarchy, the system supports a set of operations that are represented by non-terminals. The mapping of a particular layer's operations to the operations of the next layer is achieved by expanding each of the non-terminals to a sequence of non-terminals or terminals at the next lower level. Such an expansion is controlled by the production rules (P) and the associated set of probabilities (Q). Each non-terminal has two attributes s and e, which respectively denote the start and end times of an operation, such as a user session, occurring at a particular layer. The duration of an operation is the difference between s and e.

3 CASE STUDIES

In this section, we survey case studies of different types of computer systems, namely batch and interactive systems, client/server systems, database management systems, parallel systems, and WWW systems. Throughout these case studies, we focus on the workload characterization aspects of each system type, and identify the most commonly used techniques. We examine different case studies for each type of system and summarize our findings in a table. A row in these tables represents a major characterization technique (e.g., clustering) used to analyze the workload of that type of computer system. Respectively, the columns of the table represent the technique used, the workload properties explored (i.e., static or dynamic), the methods used (e.g., k-means), the approach followed (i.e., functional or resource-oriented), the basic workload component considered

(e.g., transaction, URL, session, etc.), the input parameters analyzed (e.g., # of files), the workload type (i.e., interactive, batch, or scientific), the purpose of the study, some of the useful results obtained, other techniques used in combination with this technique, and some references to case studies that used this technique.

3.1 BATCH AND INTERACTIVE SYSTEMS

A number of characterization techniques appeared in early studies of interactive and batch computer systems. Interestingly, these techniques are still the basis of approaches adopted in recent studies of various computer systems. In general, the different clustering and factor analysis techniques are commonly used to describe the static aspects of the workload while stochastic processes, numerical fitting techniques and graph based approaches are used to capture the dynamic behavior of the workload as it changes over time.

Agrawala et al. [1976] rely on clustering to extract the significant clusters in a dual processor scientific environment. The analysis depends on parameters like the CPU time, number of files, number of job steps, and number of I/Os per device type. A functional and resource-oriented clustering procedure is proposed by Serazzi [1981]. He uses clustering methods to distinguish program classes according to the consumption of six resources. In this study, the *k*-means method is used as a non-hierarchical clustering technique, and the Minimum Spanning Tree (MST) technique as a hierarchical one. As a result of this resource-oriented analysis, nine clusters are identified. These clusters are further investigated and refined by considering the functionality, such as editing, compilation or execution, of the programs and their programming languages like COBOL or FORTRAN. Thus, more descriptive clusters, like compilation of COBOL programs with an I/O intensive execution, are extracted.

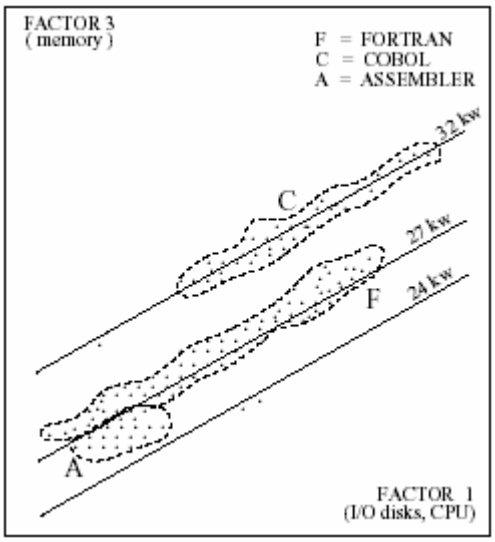

Figure 5. *Distinguished clusters of compilers projected within the factor 1-factor 3 subspace [Serazzi 1981].*

Factor analysis methods can be useful in reducing the dimensionality of the analyzed data, and in extracting distinct characteristics of the programs types. For example, Serazzi [1981] uses the Principal Component Analysis (PCA) to identify six factors that explain the correlations and the variance among the six resource-oriented parameters under consideration. Three of the six factors account for more than 75% of the total variance of the data set. Such information, together with the clustering results, help to simplify the workload model by reducing the number of parameters to be considered.

Furthermore, PCA helps distinguish the characteristics of program types such as compilation and execution. Figure 5 displays the projections of the demands of different programs' compilation on the subspace represented by factor 1 and factor 3. The programs are clustered according to the different memory demands (factor 3) placed by various compilers. The internal variations within each cluster are due to the different I/O and CPU times (factor 1).

As we mentioned earlier, other techniques, such as stochastic processes, numerical fitting and graph-based techniques, can be used to obtain a compact representation of the dynamic

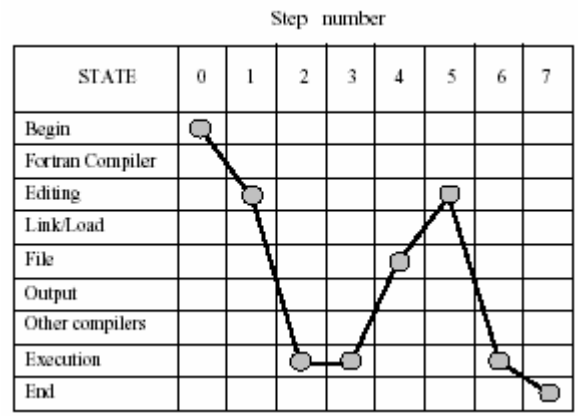

Figure 6. *Using Markov chain in modeling an interactive workload [Haring 1983].*

characteristics of workloads. Haring [1983], for example, looks at an interactive workload as a hierarchy that starts at the user session and is composed of a sequence of jobs. Each job consists of a set of tasks, and each task is comprised of a sequence of statements or commands. Eventually, the lowest level of the hierarchy consists of the physical resources consumed by each statement.

At first, clustering techniques are used to group the jobs according to the software type (e.g., compiler, linker, or loader). Seven groups are identified. At the task level, a Markov chain, whose states correspond to the software types used by the job, is employed to describe the users' behavior. The transitions between the states denote the probable sequence within the jobs belonging to the different clusters. Figure 6 illustrates the nine states in this model: seven states correspond to the identified software types, and the two extra states, BEGIN and END, are artificially introduced to denote the beginning and the termination of a job. Further experiments show that, in most of the clusters, a third order Markov chain can adequately describe the properties of the sequences.

Some analytical models can be produced by using numerical fitting techniques in order to construct a parametric model for the interactive workloads. These analytical models are capable of expressing the fluctuation in the arrival patterns of the workload components. For example, Calzarossa and Serazzi [1985] collect the arrival times of the jobs for a month. Figure 7 depicts the

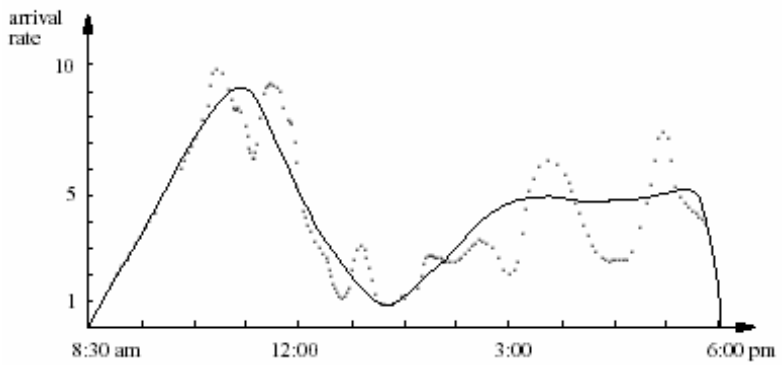

Figure 7. *The estimated rate (dotted curve) and the polynomial arrival rate (solid curve) for a one-day activity [Calzarossa and Serazzi 1985].*

arrival rate of jobs in a typical day, from 8:00am to 6:00pm. As we can see, peaks take place in the morning, around 11:00am, followed by a decrease until 1:00pm. The afternoon seems constant but still lower than the morning rates. The dotted curves in the diagram refers to the estimated rate function while the solid one refers to the corresponding polynomial function produced by the fitting technique. Adopting and applying this technique for several days shows that an eight-degree polynomial function is a suitable representation of all the analyzed arrival jobs. As a result of this analysis, various polynomial functions are reached. By clustering the coefficients of the various polynomial functions, three of them are recognized as good representatives of the whole arrival pattern.

Another dynamic technique commonly used to characterize the workload in interactive systems is User Behavior Graphs (UBG). Calzarossa and Ferrari [1986] adopt UBGs as a means of constructing a synthetic representation of a workload in terms of the command types, their resource demands and their sequence. The study considers data measured for 60 users of an interactive system running UNIX operating system. Initially, a functional approach is adopted in which the commands are grouped into clusters according to their types. A user behavior graph is then constructed in which each node denotes one of the formed clusters. Table 2 summarizes the commonly used characterization techniques in batch and interactive systems.

Technique	Static/ Dynamic	Method	Approach	Component	Parameters	Workload Type	Purpose	Results	Other Techn. Combined	Case Study Examples
Clustering	Static	*k*-means, Minimal Spanning Tree	Resource-oriented (Mostly), Functional	Job, program	CPU time, #of files, #of Job steps, #of I/Os per device type	Batch, Interactive	Distinguish program/command /task classes according to their functionality or resource consumption	(7-9) homogeneous groups	Markov Models, User Behavior Graphs	[Agrawala et al. 1976; Serazzi 1981; Haring 1983; Serazzi 1981; Calzarossa and Serazzi 1985; Calzarossa and Ferrari 1986]
Factor Analysis	Static	Principal Component Analysis	Resource-oriented	Program	CPU time, disk I/Os, language, memory	Batch	Reduce dimensionality of Data; Identify characteristics of each program types	(2-3) factors accounting for the majority of the variance in the data set	Clustering	[Serazzi 1981]
Prediction Models	Dynamic	Markov Chain	Functional	Job	Task type, user state, timestamp	Interactive	Describe user/system behavior as it transits from one state to another	Transition matrix or state diagram describing the probable sequence of jobs or tasks	Clustering	[Haring 1983]
Numerical Fitting	Dynamic	Linear and non-linear Regression	Resource-oriented	Job	Arrival rate, timestamp	Interactive	Model workload arrival pattern	Parametric model of high-degree polynomial functions	Clustering	[Calzarossa and Serazzi 1985]
Graphs	Dynamic	User Behavior Graph	Functional , resource-oriented	Command	Command type and sequence, resource consumption	Interactive	Describe user/system behavior as it transits from one state to another	A probabilistic graph model	Clustering	[Calzarossa and Ferrari 1986]

Table 2. *Characterization techniques used in batch and interactive systems.*

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

3.2 CLIENT/SERVER SYSTEMS

A client/server system consists of clients connected to servers through a network. Distributed file systems, distributed database systems, distributed multimedia systems, and WWW[1] applications, are examples of client/server systems. They can be seen as a hierarchical structure composed of three layers: client, network, and server. The clients generate requests that get transmitted through the network and are received by a server. The server fulfills the requests and sends back the replies to the client. Hence, depending on which layer is under consideration, workload consists of requests, as viewed at the client and the server layers, or packets, as viewed at the network layer.

In general, in the network layer, packet generation rate, packet size, and the routing of packets are used as parameters. We also may consider some protocol-dependent parameters like the number of packets generated per message together with their distribution. After deciding what parameters are to be monitored, the measurement tools are determined. Either special purpose devices are used or ad hoc test and monitoring tools are instrumented.

At the client and server layers, if the available counters and monitoring tools are not sufficient for collecting data then extra tools must be devised by instrumenting the client or the server. For example, web browsers can be instrumented to capture the arrival time of the requests, the size of the received file, the URL, and the user session [Crovella and Bestavros 1996]. At the server layer, Dilley et al. [1998] instrument the HTTP daemon to capture CPU and disk usage for each request. Likewise, Baker et al. [1991] instrument the kernel in a distributed file system to obtain information about file lengths and access times from each remote file system.

At the network layer, different software and hardware monitors are used to collect performance data. The software monitor, *tcpdump* [Northcutt et al. 2000], for example, can capture the flowing

[1] Covered in more detail in Section 3.5.

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

packets on the network, and provide user level control of the collected data. This control includes filtering on per host, protocol, or port basis. Parameters such as packet arrival times, packet lengths, port number, and source and destination hosts are recorded. *Sniffers* [Northcutt et al. 2000] are an example of a hardware-monitoring tool that has similar data gathering capabilities.

By analyzing the measures collected from the client, network, and server layers, a model of the overall workload of a client/server system can be obtained. Most of the existing studies characterize the workload of each layer separately. In the rest of this section, we survey some case studies and describe how they measure and analyze the parameters at each layer.

Characterization at the Client and Server Layers. At the client layer, parameters like file size and arrival times are usually considered. It has been noted that these parameters have heavy-tailed distributions, especially in the Web environment [Cunha et al. 1995; Crovella and Bestavros 1996]. The heavy-tailed distribution of the file sizes may be caused by multimedia files. Nevertheless, even pure text files have the same characteristics, but using multimedia files, like audio, video, and images increases the tailing of the distribution. Analyzing the distributions of the file sizes and the number of accesses to these files may reveal some characteristics in the user behavior. For example, an inverse correlation has been observed between file accesses and file sizes; in other words, most of the users tend to request small files.

Bodnarchuk and Bunt [1991] apply a synthetic workload model to a distributed system file server in a UNIX/NFS environment. This model is the result of analyzing the actual workload for a period of six weeks, which was obtained by capturing the requests to, and responses from, the file server. Four key parameters, namely, the frequency distribution of the requests, their inter-arrival time distribution, the file referencing behavior and the distribution sizes of read and write requests, are analyzed. The analysis simplifies the description of the model by understanding the behavior of the

requests. For example, some request types are considered significant because they are dominant while others can simply be ignored. An exponential process is not an appropriate representation for the request arrival times because the average inter-arrival time and its variance are equal to 0.50 and 121.23 seconds, respectively. The model was then constructed based on a set of parameter values observed during the workload characterization stage. In Section 3.5, we provide more examples of workload characterization that take place at the server layer.

Characterization at the Network Layer. At the network layer, the early studies collected measurements from Ethernet networks [Shoch and Hupp 1980; Gusella 1990]. The workload consists of a sequence of network packets. To characterize such a workload, several parameters are considered: packet length, packet interarrival time, and error rate. For these parameters, basic statistics along with their distributions are computed. Results show that the packet lengths have a bimodal distribution (as it has two peaks), whereas the inter-arrival times of the packets have a heavy-tailed distribution.

Shoch and Hupp [1980] characterize the traffic of a 2.94 Mbps Ethernet local area network. The network consists of 120 machines used for different kinds of applications like file transfers, file sharing and shared databases. The workload of this network is analyzed for a 24 hour period in terms of the traffic and the inter-arrival time of the packets. It is noticed that the behavior of the load is related to the time, that is, it is very heavy during the daytime, with a dip at lunch time, and light at night. The sources and destinations are encoded in a traffic matrix, which reveals the frequent patterns and the potential bottlenecks in the network. The packet size varies according to the traffic type (i.e., acknowledgements, file transfers, etc.).

Network traffic workload can also be characterized from a functional view point. For example, Gusella [1990] analyzes the network load per protocol in a large 10-Mbps Ethernet local area

network consisting of diskless workstations. The measurements are collected by instrumenting the kernel of a dedicated UNIX machine. The study reveals that protocols like TPC, NFS and ND (Network Disk) account for most of the network traffic. The data collected consists of the protocol headers together with the timestamps of all the packets flowing in the network. By observing the distributions of the interarrival times of the packets, it has been discovered that TCP is slow compared to NFS and ND.

The *self-similarity* notion is considered a fundamental characteristic of network traffic in a number of studies [Leland et al. 1994; Paxson and Floyd 1995]. Packet arrivals are characterized by a bursty nature. The network traffic can be viewed as the aggregation of bursty traffic generated by independent sources. Increasing the number of these sources increases the burstiness of the traffic. The self-similarity notion is manifested by the burstiness of the packet arrivals. By plotting the packet arrival counts, that is, the number of packet arrivals per time unit, and changing the time unit, the arrival pattern maintains the same bursty structure on different time scale. It has been discovered that this bursty arrival process characterizes the overall and the per protocol traffic, like telnet and ftp protocols, and applies to local and wide area networks. This network behavior may be an exact reflection of the user's behavior, which is characterized by a certain number of pauses. Statistical methods, namely time-domain analysis based on R/S statistic, the variance-time analysis, and the spectral-domain method using periodograms, are typically used to estimate the *Hurst parameter* which is used to measure the degree of self-similarity.

Modeling the Whole System. It should be noted that an overall model of a client/server system should take into account the analysis of client, network, and server layers together. An early study by Calzarossa et al. [1988] reflects this notion by introducing a general approach for characterizing the workload in network-based systems. Their methodology is based on a layered structure that logically

Layer	**Parameters**	**Technique**
User Terminals	Arrival Time Command Type	User Behavior Graph
Processing Node	Arrival Time Request Type Hw/sw resource consumptions	System Graph
Communication Subsystem	Arrival Time Message Type Message Length Source/Destination Addresses	Network Graph Traffic Flow Matrix

Table 3. *The characterization technique and the measured parameters used in the layered structure scheme [Calzarossa et al. 1988].*

subdivides the hardware components into three classes: user terminals, processing nodes, and communication subsystems. At each of these three layers, different basic workload components, which demand different physical resources, are identified. Probabilistic graphs are used to model the workload at each layer. Table 3 summarizes the parameters and the techniques used to characterize the workload at each layer in this study. This type of characterization is performed at each layer separately, without defining relationships or mapping mechanisms between one layer and another one.

Based on this hierarchical view, the term *networkload* was introduced [Raghavan et al. 1994]. The networkload is a collection of inputs generated by the user or the client. Figure 8 shows the session, command, and request layers identified in the networkload notion. Raghavan et al. use a Probabilistic Attributed Context Free Grammar [Fu 1974] to build a generative model that considers the hierarchical nature of the client/server environments and is capable of generating these sequences. In order to obtain such a model, data are collected at different layers and reduced using clustering techniques. They are also analyzed independently to estimate the characterizing parameters. The derived model can be used as input to a simulation model of a single server/multiple clients environment. Table 4 summarizes the commonly used characterization techniques in client/server systems.

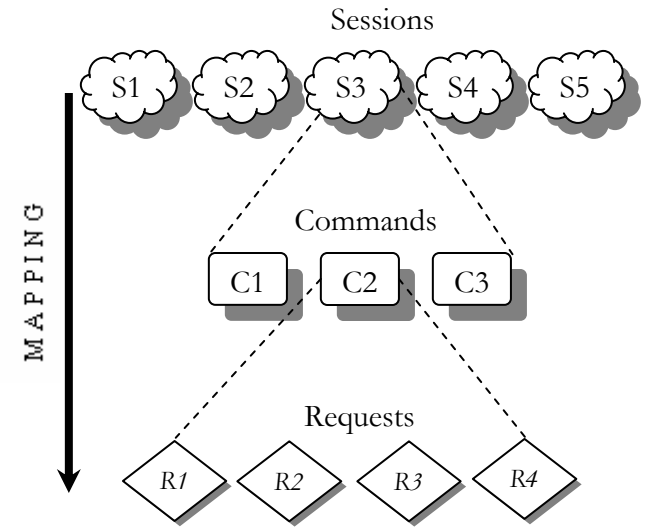

Figure 8. *The hierarchical structure of the networkload. A user session consists of a set of commands. Each command is translated to a set of low-level system requests.*

Technique	Static/ Dynamic	Method	Approach	Component	Parameters	Workload Type	Purpose	Results	Other Techn. Combined	Case Study Examples
Statistics	Static, dynamic	Exponential/ Poisson processes, Bimodal distributions, variance, averages, time-domain analysis based on R/S statistic	Functional – characterizing load per protocol or file type	File, packet	#of requests, inter-arrival time, file references, #of I/Os, packet lengths, error rate	Interactive	Model the reference behavior at file servers	Distributions of requests and their arrival; discovering self-similarity and burstiness; understand user behavior	-	[Cunha et al. 1995; Crovella and Bestavros 1996; Bodnarchuk and Bunt 1991; Shoch and Hupp 1980; Gusella 1990; Leland et al. 1994; Paxson and Floyd 1995]
Graphs	Dynamic	User Behavior Graph (UBG)	Functional, resource-oriented	Command	Arrival time, command type, message length, source/destination addresses	Interactive	Model the workload at user, system, network levels (separately)	A model representing user and system behavior	Clustering	[Calzarossa et al. 1988]
Grammar	Dynamic	Probabilistic Attributed Context Free Grammar (PACFG)	Functional	Session, command, network request	Arrival time, command type, network request type	Interactive	Build a generative model that considers the hierarchical nature of client/server	A model used as input to a simulation model	Clustering	[Raghavan et al. 1994]
Clustering	Static	*k*-means	Functional, resource oriented	User sessions, commands, network requests	Command type, network request type	Interactive	Reduce the analyzed data	Distinctive groups of commands and network requests	PACFG, UBG	[Raghavan et al. 1994]

Table 4. *Characterization techniques used in client/server systems.*

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

3.3 DATABASE MANAGEMENT SYSTEMS

As database management systems (DBMS) become increasingly popular and are often part of larger systems, they require considerable tuning to get them working at an optimal performance level [Lo et al. 1998]. As in any computing system, identifying the characteristics of the workload should aid in tuning and configuring these systems more effectively.

The significance of characterizing the DBMS transactional workload has been recognized by the DBMS community and by the Transaction Processing Performance Council (TPC), which is a non-profit organization that produces benchmarks that measure the performance of a system [TPC]. These benchmarks also provide researchers with synthesized test workloads for conducting experiments.

The most important classes of workloads in the database market are online transaction processing (OLTP) and online analytical processing (OLAP). OLTP workloads support day-to-day business activities, like banking, airlines reservations, and point of sale. They are characterized by a large number of clients who access and update a small fraction of the database through running transactions. The TPC-C benchmark simulates this type of workload [TPCC 2001]. On the other hand, OLAP workloads support business analysis and consist of long read-only queries operating on information already stored by an OLTP system. As opposed to OLTP, OLAP queries span large portions of the database. The TPC-H and TPC-R benchmarks are examples of OLAP workloads [TPCH 1999; TPCR 1999]. Recently, due to the popularity of the Internet and the e-commerce applications, a new benchmark, TPC-W, has been introduced, which has the flavor of both OLTP and OLAP workloads [TPCW 2001].

Characterizing the workload in database systems is usually based on analyzing traces and reference sequences. Traces are a collection of measures, such as pages read/written, number of locks and number of SQL statements, produced by all transactions being processed by the DBMS

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

within a time interval. Various techniques have been applied to characterize these traces, such as clustering, prediction models, numerical fitting and descriptive statistics.

Clustering is one of the common techniques used in characterizing the workload in DBMS environments. Transactions can be grouped according to their consumption of system resources [Yu and Dan 1992], or according to their database reference patterns [Yu and Dan 1994], which is called *affinity clustering*.

Artis [1978] characterizes the workload of an IBM MVS system with the aim of determining its capacity. Several parameters are used for developing cluster descriptions of the transaction workload, such as total database calls and total number of locks. By analyzing a sample of 2000 transactions, eight clusters are obtained. Four clusters among those eight account for more than 90% of the sample.

The idea of workload clustering provides valuable input to dynamic transaction routing (or load balancing) algorithms that are responsible for assigning each incoming unit of work to a processing node in the system. For example, Nikolaou et al. [1998] introduce new clustering approaches by which the workload can be partitioned into classes consisting of units of work exhibiting similar characteristics. The paper presents the CLUE and HALC clustering environments. CLUE has a set of clustering algorithms that classify OLTP transactions into classes according to their database reference patterns. HALC is a batch-mode heuristic clustering algorithm, designed to cope with the large volume of input data that is typical for real-life applications. A third on-the-fly clustering algorithm based on neural networks is also introduced. This algorithm can be used in an online fashion in systems whose workload characteristics change over time, such as a banking transaction processing system where people may perform different transactions depending on different days of the month. The traditional batch-mode clustering algorithms are not adequate in this case. However, such an algorithm should be fast, so that it does not degrade the performance of the system. For this

reason, an artificial neural network, called Optimal Adaptive K-Means [Chinrungrueng and Séquin 1995] is implemented. Experiments show that the heuristic clustering algorithm (HALC) and the Optimal Adaptive K-Means perform very well in terms of the quality of clustering, for both synthetic and real-life workload traces. On the other hand, the classic K-Means algorithm produces disappointing results.

Some studies focus on characterizing the database access patterns in order to predict the buffer hit ratio [Dan et al. 1993; Dan et al. 1995] and the user access behavior [Sapia 2000a]. A new approach, Predicting User Behavior in Multidimensional Information System Environment (PROMISE) [Sapia 2000a], identifies the user's access patterns in order to improve caching algorithms of OLAP systems. This approach models the OLAP query patterns using Markov chains and accordingly provides a prediction model for such patterns.

The idea of PROMISE is to provide the OLAP cache manger with information about the high-level (i.e., not resource-oriented) access patterns of the queries in order to make predictive prefetching. At some point during an OLAP session, the algorithm efficiently computes the possibilities for a set of queries to be executed in the near future. However, in order to perform predictive prefetching, the workload must be navigational and the think time between two accesses must be long enough to prefetch the results. In OLAP systems, the *navigational nature* of the workload is guaranteed as long as the users interactively formulate their next request using the results of the previous results [Sapia 2000b]. This navigational sequence of queries is called a *session*. In order to verify that the length of the think time is adequate, a workload of a real system, which has the interaction behavior of 18 users, is monitored over a two-month period including 260 sessions containing 3,150 queries. The final analysis shows that typical OLAP sessions have considerable think time length and are thus suited for prediction approaches.

Various statistical approaches are broadly adopted to analyze a large amount of data collected in DBMS environments [Lewis and Shedler 1976; Gaver et al. 1976; Lo et al. 1998; Barroso et al. 1998]. For example, Lewis and Shedler collect the arrival times of the transactions recorded over six days. They analyze and represent these times by means of a non-homogenous Poisson process. Using fitting techniques, an exponential polynomial function of degree 8 was obtained that shows a good approximation of the oscillatory nature of the Poisson process.

Many recent studies tend to characterize DBMS workloads on different computer architectures in order to diagnose performance degradation problems [Keeton et al. 1998; Ailamaki et al. 1999; Lo et al. 1998]. For example, Lo et al. [1998] examine the database performance on simultaneous multithreading (SMT) processors. SMT [Eggers et al. 1997] is a computer architecture in which the processor issues instructions from multiple threads in a single cycle. This study characterizes the memory-system behavior of database systems running Online Transaction Processing (OLTP- using TPC-B benchmark [TPCB 1994]), and Decision Support System (DSS- using TPC-D [TPCD 1996]) workloads by collecting traces from the Oracle database management system.

To better understand memory behavior, the different memory access patterns of both OLTP and DSS workloads are compared in the different memory segments of programs. For each segment, L1 cache miss rate, memory footprint, average number references per 64-byte block, and average number of accesses to a block until a cache conflict, are measured. By visualizing the graphs that depict the references patterns made to blocks, it has been noticed that, within each segment, cache reuse is not uniformly distributed across blocks, and for some segments is highly skewed, a fact hidden by the averaged data mentioned above. The characterization shows that while DBMS workloads have large footprints, particularly for OLTP, in the main memory, there is still substantial reusability of data in a small, critical working set, which can be cached.

A similar study [Barroso et al. 1998] presents a detailed characterization of the memory system behavior of three important classes of commercial workloads: OLTP, OLAP, and Web index search. Barroso et al. use the Oracle commercial database engine for the OLTP and OLAP workloads, and the AltaVista search engine for the Web index search. A wide variety of monitoring and profiling tools available on the Alpha platform (e.g., IPROBE [Cventanovic and Bhandarkar 1994] and ATOM [Srivastava and Eustace 1994]) were used. For example, the study provides a few general statistics gathered with the DCPI (Digital Continuous Profiling Infrastructure) [Anderson et al. 1997] profiling tool, which is an extremely light overhead sampling-based profiling system based on the processor event counters.

Analyzing reference strings in the database has been the focus of many studies. The identification of locality of reference (i.e., accessing subsets of blocks of the database), of the sequentiality (i.e., the accessing contiguous blocks) and of the transaction types provides preliminary information on both the performance and the behavior of the database. Such studies are useful in identifying the possible techniques to be adopted to enhance buffer replacement and concurrency control policies [Kearns and DeFazio 1989; Klaassen 1992]. For example, a hierarchical-functional approach is proposed by Klaassen to characterize workloads with different page reference behavior. Several parameters like number of transactions completed, number of pages read/written, the mean transaction length (i.e., the number of different pages accessed per transaction) and transaction type were used. The approach is independent of the database and of the application, and it is based on the following three abstraction levels:

- The application level represents the user's view point of the database, where data of the application and its functions are defined and their interrelationships are identified.
- The transaction level associates transaction types with the functions identified in the application level.

- The physical resource level defines low level resource usage, such as read/write page accesses, CPU demands, and deadlocks, for each transaction type.

Klaassen's approach shows that workload characterization can be done at different levels of abstraction. For example, Yu et al. [1992] focus on characterizing the workload at the transactional level. They study the structure and the complexity of SQL statements, the composition of the relations and the views, and the run-time behavior of the transactions and the queries, in a large industrial production system that runs DB2.

The analysis in this study depends largely on basic statistical summaries such as averages and variations, correlations, and distributions. An SQL trace of a two-hour interval and an image of the database catalog are obtained. To explore the static characteristics, embedded information, such as the various descriptions of tables, views, columns, indices, keys, static SQL statements, table spaces, etc., in the DB2 catalog are analyzed. Useful statistics are also collected about the different parts of the SQL statement such as the FROM, WHERE, GROUP BY, HAVING, and ORDER BY clauses.

The other part of the study focuses on the description of the run time behavior of the workload. Useful summaries, such as the number of static vs. dynamic SQL statements executed, the average response time for each transaction type, the number of rows processed or scanned, number of relations involved in the SQL statements executed, etc., are obtained. These results may provide important information needed to build a benchmark workload to evaluate alterative design trade-offs of database systems.

Extensive use of descriptive statistics is exercised in another empirical study performed by Hsu et al. [2001]. They systematically analyze the characteristics of the workload of the standard benchmarks TPC-C and TPC-D, especially in relation to those of real production database workloads. The characteristics of the production database workload of ten of the world's larges

corporations are examined and compared to TPC-C and TPC-D. The characterization was made more useful for subsequent mathematical analyses and modeling by others, by fitted the data to various functional forms through nonlinear regressions solved by Levenberg-Marquardt method [Press et al. 1986]. In order to reduce the system disturbance, the trace records were collected in a shared-memory before batch writing them asynchronously to disk. This study showed that the production workloads exhibit a wide range of behavior, and in general, the two TPC benchmarks complement each other in reflecting the characteristics of the production workloads. Some aspects of real workloads, however, are still not represented by either of the benchmarks.

Table 5 summarizes the commonly used characterization techniques in database systems.

Technique	Static/ Dynamic	Method	Approach	Component	Parameters	Workload Type	Purpose	Results	Other Techn. Combined	Case Study Examples
Clustering	Static	Heuristic, Neural network, *k*-means	Resource-oriented	DB transaction	#of database calls, #of locks, #of references	Batch, interactive	Capacity Planning, load balancing	(4-8) clusters	Statistics	[Yu and Dan 1992; Yu and Dan 1994; Artis 1978; Nikolaou et al. 1998; Chinrungrueng and Séquin 1995]
Prediction Models	Dynamic	Markov chain	Functional	DB Query	Think time, sequence of executing queries in a session	Interactive	Predict buffer hit ratio; make predictive prefetching; enhance caching	A predictive model for the near future executed queries	Statistics	[Dan et al. 1993; Dan et al. 1995; Sapia 2000a]
Numerical Fitting	Dynamic	Non-linear regression (e.g., Levenberg-Marquardt method)	Functional	DB transaction	Arrival time	Interactive	Model the arrival pattern of transactions	Degree 8 of exponential polynomial	Statistics	[Lewis and Shedler 1976; Gaver et al. 1976; Lo et al. 1998; Hsu et al. 2001]
Statistics	Static, dynamic	Basic statistics summaries (avg., distributions), non-homogeneous Poisson process, histograms	Functional	DB transaction, application, SQL statement	Arrival time, cache miss rate, memory footprint, # of references to a memory block, read/write page accesses, CPU demands, deadlocks, number of transactions completed, #of different pages accessed per transaction	Interactive	Understand memory behavior in different architectures; enhance buffer replacement and concurrency control	Degree 8 of exponential polynomial, cache reuse distribution, identifying locality/sequentiality of reference.	Numerical fitting	[Lewis and Shedler 1976; Gaver et al. 1976; Lo et al. 1998; Barroso et al. 1998; Keeton et al. 1998; Ailamaki et al. 1999; Lo et al. 1998; Barroso et al. 1998; Yu et al. 1992; Hsu et al. 2001]

Table 5. *Characterization techniques used in database systems.*

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

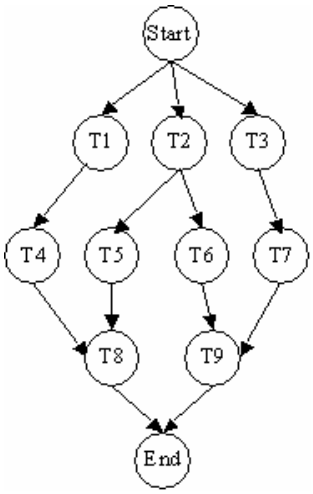

Figure 9. *An example of a task graph. The graph nodes represent parallel tasks, and arcs represent communication and synchronization controls among these tasks.*

3.4 PARALLEL SYSTEMS

Parallel applications are developed to solve problems in a shorter time and/or to solve larger problems in the same time. To meet these objectives, we need to tune, debug and diagnose the performance, which requires characterizing the parallel applications [Calzarossa et al. 2000]. Essentially, the characterization requires collecting measurements by adding instrumentation to the source code of the application, to the operating system scheduler, or to the communication libraries [Hofmann et al. 1994].

Parallel applications consist of a set of interrelated tasks. Each task is a functional unit that can be executed on a processor. Data and functional dependencies exist among tasks and these dependences are represented by communication and synchronization activities. A parallel application can be represented by a *task graph* [Beretsekas and Tsitsiklis 1989] whose nodes represent the application tasks and the arcs represent communication and synchronization controls. Figure 9 shows an example of a task graph.

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

Static Metric	Description
N	Total number of nodes
Problem size	The size of the data set
Depth	Longest path between input and output nodes
In-degree	Average number of direct predecessors of all the nodes
Out-degree	Average number of direct successors of all the nodes
Maximum cut	Max number of arcs taken over all the possible cuts

Table 6. *Static metrics derived from a task graph of a parallel application [Calzarossa and Serazzi 1993].*

In general, we can identify two types of metrics to describe parallel applications: *static* and *dynamic*. Static metrics relate to the static properties of the parallel algorithms and describe the inherent characteristics of parallelism of the applications. On the other hand, dynamic metrics describe the behavior of an algorithm over time when it is executed on a given system and indicate how effectively the parallelism is exploited.

The static and dynamic metrics can give preliminary information about the behavior of the whole application by timing parameters such as execution, computation, communication, and I/O times, and volume parameters, such as number of communications, I/O operations, and floating-point operations. By analyzing these parameters, the behavior of the parallel application becomes more understandable as we discover the correlations and tradeoffs among various performance dimensions such as computation, communication and synchronization activities, and I/O demands.

We can derive static metrics, such as those shown in Table 6, by analyzing the task graph [Majumdar et al. 1991; Sevcik 1989]. *N* reflects the task granularity, while the *problem size* deals with the size of the data set under consideration. The *depth* is the longest path in the task graph, in terms of the number of nodes starting at the initial node and ending at the final node. It is directly related to the execution time of the algorithm. The *in-degree* and *out-degree* relate to the synchronization complexity. If a node has a large number of predecessors, it most likely has to wait for

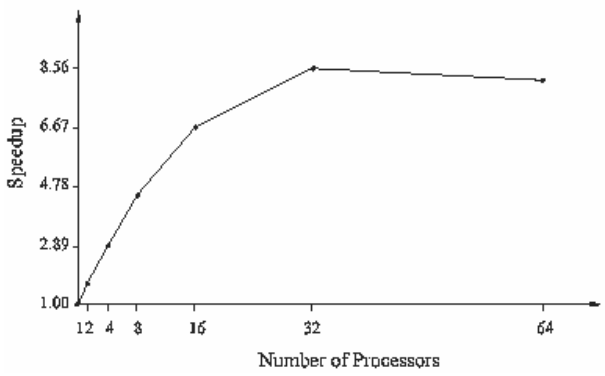

Figure 10. *An example of a speedup signature. The performance degrades as the number of processors increases from 32 to 64 due to the communication and synchronization overhead [Calzarossa et al. 2000].*

synchronization. The *maximum cut*, which is the maximum number of arcs taken over all possible cuts from the initial node to the final node, represents the maximum theoretical parallelism that can be accomplished during the execution.

Other static metrics can describe how parallelism is exploited by the application on a given architecture. Such metrics can be *single values* or *signatures*. Single value metrics, such as average computation time, number of I/O operations, and average message lengths, can be determined by executing a particular algorithm with a given number of processors p. Table 7 contains some examples of single-value metrics.

A *signature* is the result of plotting one of the single value metrics as a function of the number of available processors. For example, the *speedup signature* $S(p)$ describes the gain in time achieved by the parallel algorithm on p processors with respect to the serial execution. Speedup measures how much the execution time decreases with an increase in the number of allocated processors. Figure 10 shows a speedup curve of an application executed with a number of processor ranging from 1 to 64. Note the degradation of the performance when the number of processors is increased from 32 to 64, which shows that increasing the number of allocated processors does not always lead to a performance enhancement. Apparently, the benefit of allocating additional processors does not

Metric Type	Metric	Description
Single-Value (static)	t_{comp}	Avg. computation time
	t_{comm}	Avg. communication time
	$n_{messeages}$	Avg. number of messages sent/received
	$l_{messages}$	Avg. length of the messages
	$n_{I/O}$	Number of I/O operations
Signatures (static)	$T_{comp\ (p)}$	Global computation time vs number of processors
	$T_{comm\ (p)}$	Global communication time vs number of processors
	$T_{(p)}$	Global execution time vs number of processors
	$S_{(p)}$	Speedup vs number of processors ($S(p)=T(1)/T(p)$)
	$E_{(p)}$	Effeciency vs number of processors ($E(p)=S(p)/T(p)$)
Profiles (dynamic)	n_{busy_proc}	Number of busy processors vs execution time
	n_{comm_proc}	Number of communicating processors vs execution time
	n_{comp_proc}	Number of computing processors vs execution time

Table 7. *Other metrics used to characterize parallel applications [Calzarossa and Serazzi 1993].*

compensate for the costs due to the increase in communication and synchronization overhead. In addition to speedup, other performance metrics such as *efficiency*, can be used to assess the effective exploitation of allocated processors. Table 7 describes some examples of signature metrics.

Although the static metrics give insights into the parallelism that can be achieved by an application, they fail to express how parallelism is exploited by the application as the execution progresses. Thus, dynamic metrics, such as *profiles* and *shapes*, are used. Profiles express, as a function of execution time, the number of processors that are involved in a particular activity. Table 7 describes some profile metrics and Figure 11(a) depicts an example. Shapes [Sevcik 1989] are better tools for characterizing the behavior of a parallel algorithm as they derive more information from profiles. A shape, as shown in Figure 11(b), is a cumulative plot of the fraction of execution time when a certain number of processors are busy. The application shape is the fraction of execution time in which a given number of tasks is active.

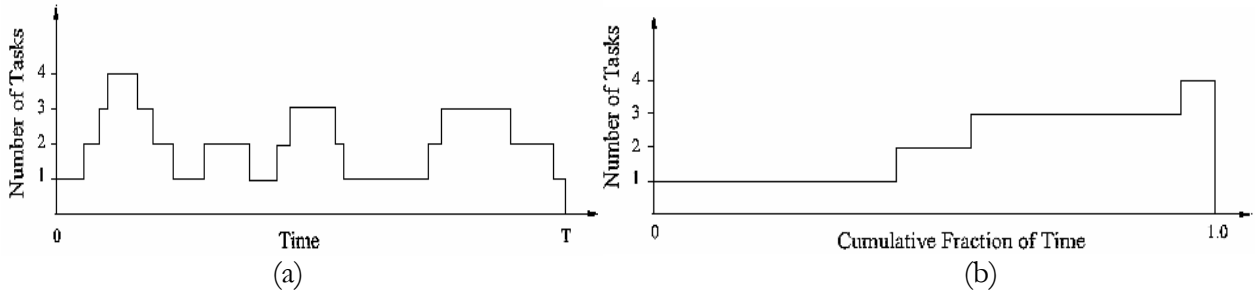

Figure 11. *An example of an application profile (a) and its shape (b) [Sevcik 1989].*

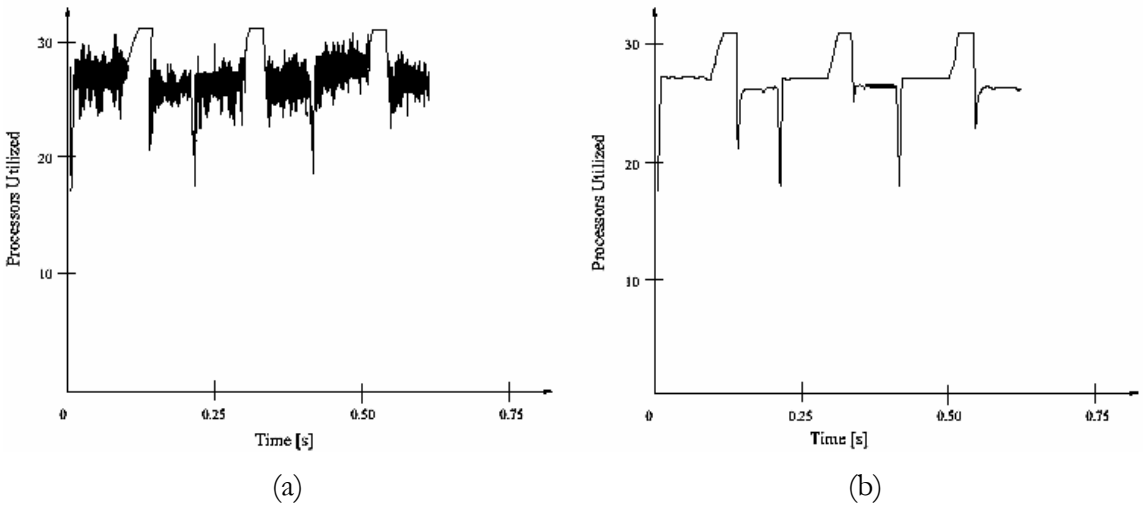

Figure 12. *Example of execution profile (a). Phases are easily identified after smoothing (b) [Carlson et al. 1992].*

Another approach to characterizing parallel applications is to use multiple views [Le Blanc 1990]. For example, the processor and data parallelism views can be obtained. These views describe the behavior of the processor against the data it has to process, and the interrelationships between the processor and the other processors with respect to the communication and synchronization tasks. These views are also useful in identifying any unbalanced load among the processors.

Several characteristics can be recognized by identifying application *phases*. The execution of an application is seen as a sequence of computation, communication, and I/O phases. Carlson et al. [1992] analyze the execution profile to identify the phases of which it consists. The profile is seen as a sequence of alternating periods of roughly uniform processor utilization separated by periods of

sudden transition of processor utilization. Figure 12(a) shows an example of an execution profile. Phases, which this execution profile consists of, are more easily identified in Figure 12(b) after using smoothing techniques.

Recent studies [Rosti et al. 1998; Smirni and Reed 1998] focus on the characterization of scientific parallel applications, which are typically described as computationally intensive and becoming more I/O intensive due to the vast volume of data to be processed. The characterization in these studies focuses on the behavior of I/O requests, such as reads, writes, and seeks. For each of these I/O requests its count and duration are measured. The analysis of the temporal patterns of I/O requests help identify different phases. These phases show the burstiness of the accesses and their non- sequential behavior. They also show the presence of interleaved and strided patterns. Table 8 summarizes the commonly used characterization techniques in parallel systems.

Technique	Static/ Dynamic	Method	Approach	Component	Parameters	Workload Type	Purpose	Results	Other Techn. Combined	Case Study Examples
Graph	Dynamic	Profiles, shapes, phases	Resource-oriented	Application	# of processors being busy computing or communicating at certain time	Scientific	Identify application phases	Graphs of communication, computation, and I/O phases	Statistics	[Sevcik 1989; Carlson et al. 1992; Calzarossa et al. 2000; Calzarossa and Serazzi 1993; Rosti et al. 1998; Smimi and Reed 1998]
Statistics	Static	Signatures, avg. computation/ communication time, avg. # of sent/received messages, avg. message length, # of I/O operations	Resource-oriented	Application	Timing parameters: execution, computation, communication times, I/O times, Volume parameters: # of communications, I/O operations, floating-point operations	Scientific	Obtain inherent characteristics of a parallel application run on a specific number of processors.	Single-value metrics, and signatures	-	[Calzarossa et al. 2000; Calzarossa and Serazzi 1993; Majumdar et al. 1991; Sevcik 1989]

Table 8. *Characterization techniques used in parallel systems.*

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

3.5 WORLD WIDE WEB SYSTEMS

Internet-based systems can be classified as client/server systems as described in Section 3.2, but we prefer to devote this section to them because the literature is rich with research papers pertaining to workload analysis issues. This should not be a surprise as we observe, day after day, the explosive increase of Internet popularity.

E-Commerce applications are complex distributed systems. They combine the functionality of several components, such as web servers, DBMSs and secured payment subsystems. As a result, various methods, within different combinations, have been proposed and used to capture the characteristics of e-commerce workloads.

Krishnamurthy and Rolia [1998] present a case study on Quality of Service (QoS) measures for an electronic commerce server. They study the behavior of an electronic commerce server under several controlled loads and study response time measures for several workload classes: individual Universal Resource Locators (URLs), groups of functionally related URLs, and URL sequences. They use an analytical model combined with empirical knowledge of server behavior to show that mean response time can be a good predictor for the 90^{th}-percentile of response times.

Initially, they chose to use Layered Queuing Models (LQM) and the Method of Layers (MOL) [Rolia and Sevcik 1995] to model the system. LQMs are Queuing Network Models (QNMs) [Lazowska et al. 1984] extended to include contention for software resources such as pools of server processes as well as contention for hardware resources such as CPUs and Disks. Unfortunately, the results obtained using Mean Value Analysis (MVA) show that these queuing models are not sufficient to predict the 90^{th}-percentile of response times. Accordingly, this approach is augmented by using additional statistical manipulations guided by empirical data collected from existing systems.

Pitkow and Pirolli [1999] introduced improved models to predict the navigation behavior of WWW surfers. This prediction ability may be exploited in different applications such as searching

---

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

through WWW content, recommending related WWW pages, predictive prefetching, and analyzing web site designs.

Modeling and predicting user surfing paths typically involve tradeoffs between model complexity and predictive accuracy. Pitkow and Pirolli's goal is to explore predictive modeling techniques that reduce model complexity without sacrificing predictive accuracy. Their proposed techniques predict the future of surfing paths by merging a web-mining method that extracts significant surfing patterns by the identification of *longest repeating subsequences* (LRS) [Crow and Smith 1992] and pattern matching methods. The LRS technique reduces the complexity of the model by focusing on the significant surfing patterns. Compared to various Markov models, the experiments in this research show that longest repeating subsequence models, called Hybrid LRS-Markov models, are able to significantly reduce the model size while retaining the ability to make accurate predictions.

Arlitt and Williamson [1996] present a WWW workload characterization study that focuses on finding workload *invariants*, that is, characteristics that apply across all the data sets studied. Six different data sets are used in this study: three from academic environments, two from scientific research organizations, and one from a commercial Internet provider. These data sets represent three different orders of magnitude in server activity, and two different orders of magnitude in time duration, ranging from one week of activity to one year of activity. By using several basic statistical techniques ten invariants are identified. These invariants are deemed important because they potentially represent universal truths for all Internet Web servers. They are exploited to identify two possible strategies for the design of a caching system to improve Web server performance, and to determine bounds on the performance improvements possible with each strategy. The study identifies the distinct tradeoff between caching designs that reduce network traffic, and caching designs that reduce the number of requests presented to Internet Web servers.

Several relatively recent Web forces may someday undermine or change these ten Web server invariants. These forces include Web crawlers, improved protocols for Web interaction, small scale and Web caching architectures, and a growing trend toward the use of video, audio, and interactivity on the Web (e.g., CGI, Java). It will be interesting to see how long these invariants remain true.

Formal grammars have also been used to model WWW applications' workloads. Kotsis et al. [1997] present an approach for generating a profile of requests submitted to a WWW server, such as GET and POST commands, which explicitly takes into account user behaviors when surfing the net. The *Probabilistic Attributed Context Free Grammar* (PACFG) [Fu 1974] is used to derive a model for mapping user-oriented views of the workload, namely the conversations made within browser windows, to the methods submitted to the Web servers.

This approach identifies six hierarchical layers from the user level, characterized in terms of sessions, to the level of TCP/IP requests, that is, the actual load imposed on the network. This approach differs from others in that it considers both the actual physical characteristics of the system as well as the user-oriented view. Thus, the analyst is able to investigate both changes in user behavior as well as the effects of changes in the system characteristics.

PACFG is general enough to cover any form of web activity (e.g., different browsers, different protocols, JAVA applets, etc.). It can also be parameterized in order to define worst-case scenarios, such as capturing the system behavior under heavy load. However, it would be interesting to assess the expressiveness and representativeness of this grammatical approach in contrast to simpler models of WWW traffic characterization.

Data warehousing and data mining technologies can be used to discover interesting characteristics in Web logs. A knowledge discovery tool, WebLogMiner, for mining web server log files has been developed [Zaïane et al. 1998]. The approach is mainly motivated by the fact that

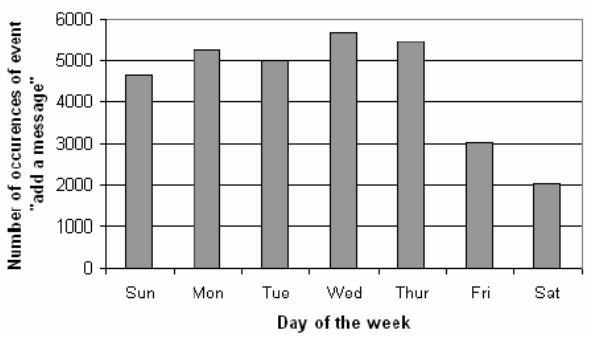

Figure 13. *OLAP Analysis of Web log database. It shows how many new messages were created on each day of the week over a whole semester for a particular site [Zaïane et al. 1998].*

condensing colossal files of raw web log data in order to retrieve significant and useful information is a nontrivial task.

Once the multi-dimensional data cube is constructed, various OLAP techniques, such as drill-down, roll-up, and slice-and-dice, are applied to provide further insights to the target data set from different perspectives and at different conceptual levels. Some typical summarization includes the following: domain summary, event summary, session summary, bandwidth summary, referring organization summary, and browser summary. Figure 13 illustrates a simple yet, typical example of summarization using this OLAP technique. It shows how many new messages were created on each day of the week over a whole semester for a particular site.

In the next stage, data mining techniques are put to use with the data cube to predict, classify, and discover interesting correlations and patterns. This project makes good use of several data mining tasks [Han et al. 1997; Han and Fu 1995; Kamber et al. 1997] including summarization, comparison, association, classification, prediction and time-series analysis. The latter is the most important data mining task in the Web log analysis because all Web log records register time stamps, and most of the analyses focus on time-related Web access behaviors. The time-series analysis

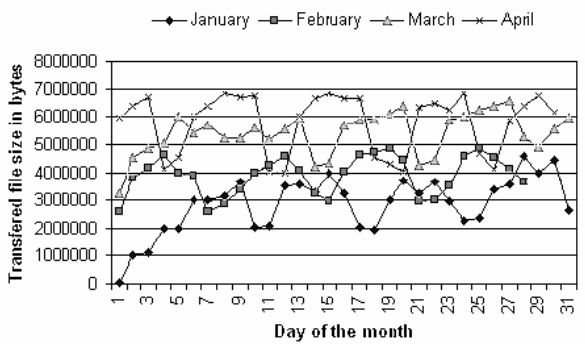

Figure 14. *The analysis of the network traffic of a site over a four-month period [Zaïane et al. 1998].*

includes network traffic analysis, event sequence and user behavior pattern analysis, transition analysis, and trend analysis.

Figure 14 shows the network traffic generated by a particular site over a four-month period. The chart shows that the traffic had a general increasing trend over the four months, with monthly cycles reflecting the peaks during the weekdays and valleys during the weekend. Also transition metrics are obtained from the transition analysis for a given group of users over a certain period of time, which indicates the probabilities one event follows another.

Graphs are commonly used to model the e-commerce workloads [Menascé et al. 1999; Krishnamurthy and Rolia 1998]. Menascé et al. argue that the traditional workload characterization for e-commerce sites in terms of hits/sec, pages viewed/sec, or visits/sec, are not appropriate. According to their perspective, e-commerce workloads are composed of *sessions*. A session is defined as a sequence of requests of different types made by a single customer during a single visit to a site. Examples of requests for an online shopper are: browse, select, add to the shopping cart, search, pay, and user registration. The allowed sequences of requests can be described by a state transition graph called a *Customer Behavior Model Graph* (CBMG). Figure 15 shows an example CBMG. This graph has one node for each possible state and transitions between these states. A probability is assigned to each transition. Thus, different types of users may be characterized by different CBMGs

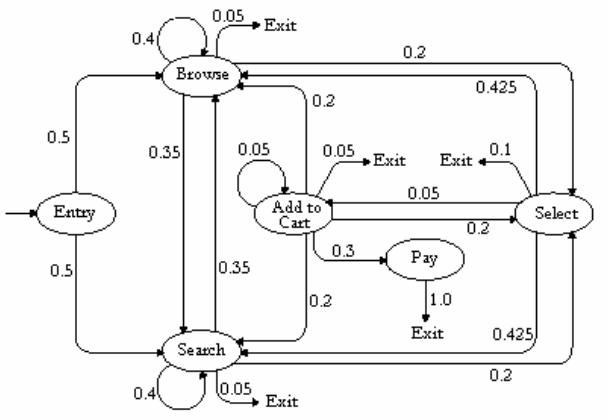

Figure 15. *Customer Behavior Model Graph for an occasional buyer [Menascé et al. 1999].*

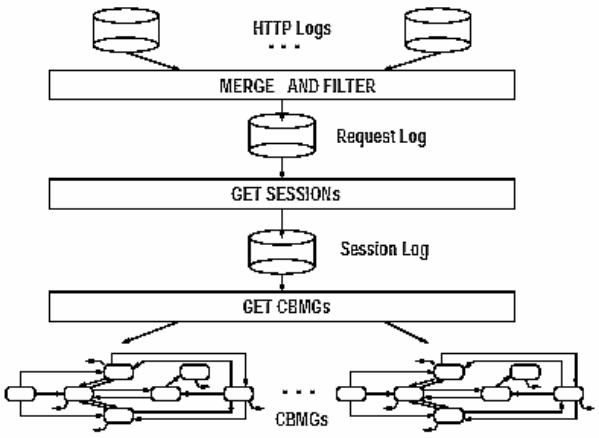

Figure 16. *Workload Characterization Methodology proposed by [Menascé et al. 1999].*

in terms of the transition probabilities. By analyzing the CBMG, useful metrics can be derived like average session length, average number of items bought per customer visit, and buy to visit ratio.

Different classes of visitors to a site exhibit different navigational patterns. Each customer class can be represented by its own CBMG, and workload characterization for e-commerce entails determining the set of CBMGs that best characterize customer behavior. The study proposes a resource-oriented workload characterization methodology for the e-commerce sites. As shown in Figure 16, this methodology is based on two algorithms. The first one takes as input conventional HTTP logs and generates a session log. The second algorithm takes as input the session logs and performs a clustering analysis which results in a set of CBMGs that can be used as a compact representation of the e-commerce workload. This approach is verified by generating artificial e-commerce logs based on different CBMG patterns drawn from data collected form the operation of real online bookstores. Afterwards, clustering algorithms are applied to the logs. The outcome is groups of customers that exhibited similar navigational behavior. The algorithms are also applied to logs of an actual e-commerce site. Table 9 summarizes the commonly used characterization techniques in the World Wide Web systems.

Technique	Static/ Dynamic	Method	Approach	Component	Parameters	Workload Type	Purpose	Results	Other Techn. Combined	Case Study Examples
Analytical Modeling	Static, Dynamic	Queuing Network Model, Layered Queuing Model, Method of Layers	Functional, resource-oriented	URL	CPU time, disk I/Os, Response time, URL sequences, URL type	Interactive	Study QoS measures for E-Commerce Server	The mean response time is a good indicator for the 90th percentile of response times	Statistics	[Krishnamurthy and Rolia 1998]
Prediction Models	Dynamic	Hybrid LRS-Markov models	Functional	URL	URL sequences	Interactive	Predict the navigation behavior of WWW surfers	Producing simple and accurate model better than Markov's	Markov Model	[Pitkow and Pirolli 1999]
Statistics	Static	Averages, distributions, histograms, correlations, COV, Hurst parameter,	Functional	URL, File	URL timestamps, Transfer size, file size, file type, request success rate, client locality	Interactive	Discover Web invariants	Discovering ten web invariants	-	[Arlitt and Williamson 1996]
Grammar	Dynamic	Probabilistic Attributed Context Free Grammar	Hierarchical-Functional, resource-oriented	User session, HTTP request, TCP/IP request	URL time stamps, URL type, think time	Interactive	Profile requests submitted to WWW servers	Map the high user-oriented view to low TCP/IP requests	-	[Kotsis et al. 1997]
Data mining and data warehousing	Static, Dynamic	OLAP techniques: Drill-down, roll-up, slice-and-dice Data Mining tools: association, clustering, classification, transition/trend analysis	Functional, Resource-oriented	URL	Timestamps, IP addresses, transfer size, URL type, file size	Interactive	Discover interesting characteristics in Web log	Discovering web access patterns and trends	Prediction techniques and time-series analysis, Clustering, Statistics	[Zaïane et al. 1998; Han et al. 1997; Han and Fu 1995; Kamber et al. 1997]
Graph	Dynamic	Customer Behavior Model Graph (CBMG)	Functional, Resource-oriented	Session	Timestamps, #of hits, # of pages viewed, #of visits	Interactive	Model E-commerce workloads	Deriving useful metrics from CBMG like avg. # of visits per user state, avg. session length, buy to visit ratio.	Clustering	[Menascé et al. 1999; Krishnamurthy and Rolia 1998]

Table 9. *Characterization techniques used in World Wide Web systems.*

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

4 CHARACTERIZATION FRAMEWORK

Based on the case studies surveyed in this paper, we propose a general framework for the workload characterization process. The framework makes the process more systematic, emphasizes some essential steps needed to derive a representatively good workload model, and prevents some common problems.

The main difficulties that may be encountered throughout the workload characterization process are:

- **Difficulty of System Instrumentation**. Systems need to be instrumented in order to obtain performance measurements. This may require the insertion of some probes, like counters, into the system itself or into the operating system. This task is challenging due to the complexity of the systems and the typical absence of the source code.
- **System Disturbance**. Instrumenting the system is an intrusion that adds extra overhead. Hence, the degree of intrusion should be minimized to reduce the perturbation of the system's behavior under the investigated workload.
- **Complexity of Analyzing Large Volume of Performance Data**. A large amount of system measurements are needed to construct a workload model [Calzarossa et al. 2000], which increases the complexity of managing and analyzing the data.
- **Validating Model Representativeness**. Assessing the workload model representativeness, that is, how accurately the model represents the real workload, is a key issue [Menascé et al. 1994]. Normally, modeling tends to hide some details that might be desirable to study. Hence, a careful decision should be made about the model's abstraction level in the *requirements analysis phase* (explained next). This should help identify how much information loss can be tolerated and what important features must be included in the model.

We gratefully acknowledge the financial support from IBM, CITO, and NSERC.

- **Model Compactness**. The characterization process should result in a compact model. It is impractical for the workload model to incorporate all the basic components of the real workload. Ideally, a compact workload model should place a much smaller demand on the system than the actual workload [Menascé et al. 1994].

Ferrari et al. [1983] describe a methodology for constructing a workload model. We augment their methodology to produce a framework that introduces the following additional concepts:

1. *Creating a Performance Database*. Building a database for the workload parameter values provides a robust way of storing and managing large volume of performance data. It also provides a solid foundation for the application of any analytical technique that might be adopted in the subsequent phases.
2. *Distinguishing between the static and dynamic techniques*. This distinction is sometimes important in the analytical phase in order to choose the appropriate tool, and to create an adequately descriptive executable workload model.
3. *Using data warehousing and data mining technologies*. In addition to the traditional analytical and statistical techniques commonly used in workload characterization, we suggest in this framework exploiting the capabilities of the data warehouse technology [Chaudhuri and Dayal 1997] and data mining tools.

The *multi-dimensional data cube* in a data warehouse provides operations such as *drill-down*, *roll-up*, and *slicing and dicing*. These operations offer online analytical processing (OLAP) capabilities, including an engine for deriving various statistics, and a highly interactive and powerful data retrieval and analysis environment. The data warehouse approach also overcomes the complexity problem stemming from processing large data sets.

Besides the OLAP tools, the analytical capacity can be extended further by adopting data mining techniques, which can discover implicit knowledge in the performance data that can be expressed in

terms of rules, charts, tables, graphs, and other visual forms for characterizing, classifying, comparing, associating, or predicting the workload. Data mining techniques have been used to discover interesting patterns and features in customers' data that may lead to better marketing strategies. Similarly, in the workload characterization framework, we mine for interesting patterns and key characteristics in the system's workload. The integrated use of data warehousing and data mining has proven useful in analyzing web logs [Zaïane et al. 1998] and we believe that using both technologies as part of the workload characterization methodology would be beneficial too. Figure 17 shows the framework of the workload characterization process. Deriving a workload model consists of three phases: requirements analysis phase, construction phase, and validation phase. Next, we describe these phases and explain the tasks involved in each of them.

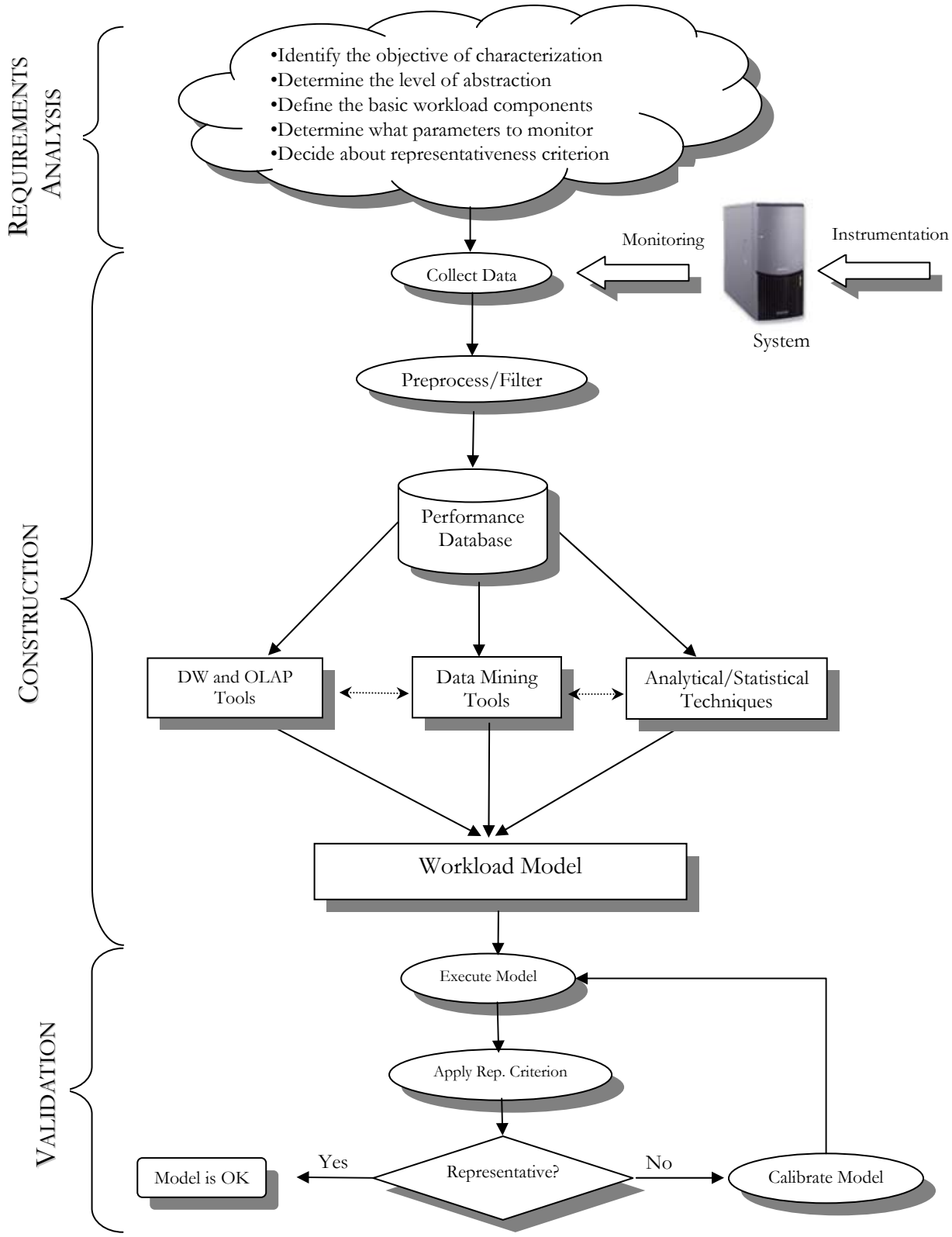

Figure 17. *Workload Characterization Methodology.*

4.1 REQUIREMENTS ANALYSIS PHASE

The reasons for characterizing a system's workload should be clear from the beginning because they help derive the appropriate workload model. Therefore, based on a clear identification of the goals of the performance study, analysts must determine the following:

Abstraction Level. Depending on the intended use of the model, the level of abstraction at which the characterization will take place should be determined. The system can be viewed as a hierarchy; the highest level in this hierarchy is *functional* and the lowest one is *physical*. At the functional level, for example, the analyst may focus on identifying the types of applications executed in the system, identifying the kinds of web objects that are requested frequently, or grouping database transactions according to their functionality. At the physical level, they may categorize workload components, such as transactions, TPC/IP requests, or user interactive commands, according to their resource consumptions (e.g., CPU time, I/Os, and memory space). The higher the level, the lower the amount of detail with which the workload can be described. The selection of the level of detail helps in making other decisions like the choice of the *basic workload component*.

Basic Workload Component. The smallest unit of work must be determined. As shown in Figure 1, a workload component can be an application, a script, a command, a SQL statement, a user session, a transaction, a CPU instruction, a request, or a job. For example, applications and CPU instructions can be considered as basic workload components at the functional and physical levels, respectively.

Workload Parameters. Depending on the abstraction level and the basic workload component, parameters are chosen to give a quantitive description of the workload components. Examples of workload parameters are packet size, arrival time, number of I/O instructions, memory space demand, and number of file handles required. It is preferable to choose parameters that are dependent on the workload rather than on the system. For example, response time and CPU time

are not appropriate as workload parameters since they are highly dependent on the system currently executing the workload. In particular, those characteristics that have an impact on the system performance should be included in the workload parameters. Parameters selection may also be restricted by the capability of the monitoring tools currently available in the system.

Criteria of Evaluating Model Representativeness. The criteria of evaluating the accuracy and representativeness of the derived model should be determined. They are used to validate the model as explained in Section 4.3.

4.2 MODEL CONSTRUCTION PHASE

This phase consists of three main tasks:

Collecting and Preprocessing Performance Data. During the measurement interval, the workload parameter values are collected from the system. The raw data may not be ready for direct analysis, so, further processing may be needed to put the data in a clean state and an appropriate format. For example, the raw data set usually contains noise and outliers that may distort the results of the subsequence analysis. Furthermore, some type of transformations might be needed in this step. For example, if one of the parameter's density functions is highly positively skewed, a logarithmic transformation is needed.

Creating A Performance Database. After preprocessing and filtering the raw data, a relational database is created to store the performance data. The database facilitates information extraction and data summarization based on individual attributes.

Analysis Stage. Analyzing the workload parameter values aims to extract the workload's static and dynamic features. In Section 2, we described some of the tools commonly used to perform the static and dynamic analyses. The static analysis tools explore the intrinsic features of the workload and partition the workload components into homogeneous classes or groups. However, in order to make

the derived workload model executable we need to capture the characteristics of the workload over time in order to reproduce the correct workload mixes. Hence, the dynamic properties of some time series are considered. Stochastic processes, numerical fitting techniques, and the various predictive models are useful in describing the behavior of the workload over time. As depicted in Figure 17, the traditional analytical/statistical techniques and the proposed data mining and OLAP tools can be used, separately or together, to analyze the performance data in order to characterize the workload.

Analyzing the static characteristics helps to choose representative components (mixes) that can reflect the key properties of the real workload. Analyzing the dynamic behavior of the workload completes the picture by describing the distribution and the sequence of execution of these workload components. Determining the static and dynamic characteristics of the workload can be the ultimate goal of the workload characterization because such knowledge can be adequate to facilitate tuning and enhancing the system's performance. Hence, the characterization process may stop at this point. However, the model can be further processed to generate an executable, runnable model that can be practically ported to different systems to assess their performance. A benchmark is an example of an executable workload model. The executable format of the workload model is also a means of its validation, as explained next.

4.3 MODEL VALIDATION PHASE

Validating the workload model is sometimes not straightforward. One way of examining the accuracy of the derived workload model is to assess its effect on the system compared with the effect of real workload [Ferrari et al. 1983]. As can be seen in Figure 18, if the performance measurements resulting from the application of the workload model and the real workload are the same or proportional, then we have a good model. For example, Keeton and Patterson [2000] proposed and evaluated simplified *microbenchmarks* for studying the architectural behavior of database

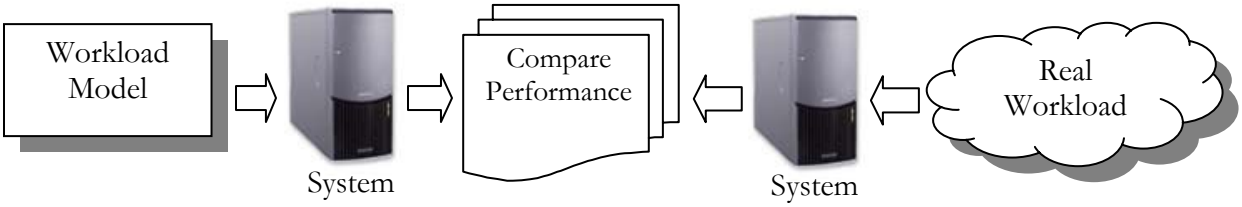

Figure 18. *Validating the representativeness of a workload model.*

workloads. These microbenchmarks poses simple queries of the database to generate the same dominant I/O patterns exhibited in more complex, fully-scaled workloads like TPC-C and TPC-D. One of the potential benefits from this microbenchmark approach is smaller hardware requirements. The representativeness of the new models was evaluated by comparing the processor and memory system characteristics of the microbenchmarks with that of fully scaled workloads running on similar hardware. These metrics were selected because most fully scaled database servers are configured with enough disks to be CPU bound; hence processor and memory behavior are important factors in determining database performance [Barroso et al. 1998].

Other techniques of validation may take into account criteria like arrival time of components and the resource usage profile [Jain 1991]. If the derived workload model does not provide sufficient accuracy then some calibration of its parameters (static characteristics) or for its component mixes (dynamic characteristics) is required. The calibration process is repeated until a satisfactory level of representation is reached.

5 CONCLUSIONS

Characterizing the system's workload is an essential early step in any performance study. Although workload characterization, like performance evaluation, is still considered more of an art than a science, the methodology discussed in this paper can be deemed a general framework for deriving a workload model. A substantial amount of details in this framework are highly dependent on the objectives of the performance study as well as the type of system. We propose using data

warehousing and data mining technologies as a promising analytical approach. It may provide a potential solution for some of the well-known problems in workload characterization like the difficulty of managing large volumes of performance data sets and the complexity of analyzing them. This should lead to a better scalability, more interactivity, and a variety of different analyses possible to perform.

The case studies surveyed in this paper show that a wide range of analytical techniques can be used to extract the static and dynamic characteristics of the workload. More than one technique may be combined in order to obtain the desired model. In general, we have noticed that identifying distinct classes in the workload using the various clustering techniques is the main goal of many studies.

The notion of multi-layer workload characterization has been adopted by many workload characterization studies. It is based on viewing the system as a hierarchical structure, which allows the characterization process to take place at any level in this hierarchy. For example, in network-based systems, characterization can be accomplished at many levels: user level, application level, protocol level, and network level.

A multi-layer characterization allows insight into how changes at the upper levels can affect the lower levels, and enables the prediction of the impact of new applications or systems. By analyzing the measures collected from each layer, a model of overall workload of the system can be obtained. Nonetheless, we have found that most of the studies characterize the workload of each layer separately. Probabilistic graphs techniques, such as User Behavior Graphs, have been commonly used for modeling the workload at each layer.

Workload characterization typically relies on analyzing performance data collected from the system. The choice of what to measure depends on the objective of the study, the workload features to be evaluated, the level of abstraction (or details) required, and the availability of monitoring tools

to collect the proper measures. The selection of what to measure is critical. Indeed, there is a tradeoff between the amount of detail to be measured and the perturbations caused by monitoring. Measurements collected from the system are not only important to the analysis phase; they are also useful for parameterizing the derived models with empirical data drawn from the real system. In some cases, such parameterization is essential to obtaining a successful model.

However, and as already pointed out, obtaining the *proper* measurements from the system is sometimes challenging. For example, web logs have been used as the primary source of system data to model the workload of WWW applications. While this may reflect the actual use of the resources on a site, it does not record reader behaviors like frequent backtracking or frequent reloading of the same resource if the resource is cached by the browser or a proxy. Other means of data gathering like client-site log files collected by the browser, or a Java Applet have been suggested. However, while these techniques solve such problems, they demand the user's collaboration, which is not always available. In some systems, for example networks, special equipment such as network cards, bridges, routers, and gateways, constituting the network-based systems make the characterization process much harder. As a result, new measuring tools have been devised in order to collect parameter values from the system.

We believe that workload characterization will remain the focus of researchers and will constantly keep progressing in order to exploit newly introduced techniques and to cope with the requirements of new computer architectures. We also believe that no matter what new performance-oriented architectures have to offer toward enhancing performance, the notion of characterizing the workload and identifying its features should always lead to better improvements.

6 FUTURE DIRECTIONS

Some of the trends and challenges we expect to witness in the workload characterization field are the following:

- New technologies and architectures will keep changing the way users approach computer systems and interact with them, which will accordingly keep varying the workload characteristics and constantly adding extra levels of complexity to the characterization process. For example, ubiquitous computing, such as PDAs and appliances attached to the Internet, will add at least an order of magnitude to the number of traffic sources, and will add different characteristics to the workload [Crovella and Lindemann 2000]. New local communication technologies such as Digital Subscriber Lines (DSL), cable modem, and Universal Mobile Telecommunications System (UMTS), will affect the traffic patterns and intensities of workload components. Understanding the impact of these changes, in terms of workload characteristics and system performance, is challenging. Working on deriving a model that represents the consolidated traffic from millions of such sources is another problem that must be solved.
- We need more experiments to assess the benefits of creating a relational performance database and to evaluate the effectiveness of using data warehousing and data mining technologies in the workload characterization framework.
- Although many case studies have adopted the notion of multi-layer workload characterization, the relationships between these layers are currently not well understood. Mechanisms that transform and map the different views of the layers are lacking. The existence of such mechanisms will help us understand the impact of changes in the workload of one layer on the rest of the layers. A potential solution may be to use consistent formalisms across layers. Usually,

going from higher layers to the lower layers is easier due to the causality relationship, but going from lower layers to higher ones is useful too and is still deemed an important challenge.

- Workload characterization at the user level should take into account the mutual influence of the user behavior and the performance of the system on each other. In other words, many of the studies derived models that describe how the system would perform when the users behavior changes. Unfortunately, these studies overlooked the potential impact of the system performance on the user's behavior and access patterns. For example, a long response time may discourage a user from requesting consecutive URLs or queries, which leads to less interactivity. Such mutual impact should be taken into consideration.
- We look forward to seeing systems that have integrated environments and tools that are able to address the performance issues encountered in the various application domains. Such tools should be able to provide automated monitoring capabilities, collect performance measurements, and to analyze them. Having such features makes systems self-manageable and dynamically self-adaptable to their workload. Furthermore, by understanding the dynamic behavior of the workload, these systems will be able to forecast the upcoming load and adjust their resource allocations to efficiently handle the future workload. In other words, these systems will be **workload-aware systems,** which can analyze their workload characteristics in these environments and therefore configure themselves properly to attain the desired quality of service.

7 REFERENCES

AILAMAKI, A., DEWITT, D., HILL, M., AND WOOD, D. 1999. DBMSs On A Modern Processor: Where Does Time Go?. In *Proc. of Int. Conf. On Very Large Data Bases* (*VLDB '99)*, (Sept 1999), 266-277.

AGRAWALA, A., MOHR, J., AND BRYANT, R. 1976. An approach to the Workload Characterization Problem. In *Computer*, (1976), 18-32.

ANDERSON, J., BERC, L., DEAN, J., GHEMAWAT, S., HENZINGER, M., LEUNG, S., SITES, R., VANDERVOORDE, M., WALDSPURGER, C. AND WEIHL, W. 1997. Continuous Profiling: Where Have All the Cycles Gone?. In *Proc. of the 16th International Symposium on Operating Systems Principles*, (Oct. 1997), 1-14.

ARLITT, M. AND WILLIAMSON, C. 1996. Web Server Workload Characterization: The Search for Invariants. In *Proc. of SIGMETRICS '96*, (May 1996), 126--137.

ARTIS, H. 1978. Capacity Planning for MVS Computer Systems. In *Performance Evaluation of Computer Installations*, North Holland, 25-35.

BAKER, M., HARTMAN, J., KUPFER, M., SHIRRIFF, K., AND OUSTERHOUT, J. 1991. Measurements of a Distributed File System. In *Proc. of ACM Symposium on Operating Systems Principles,* 198-212.

BARROSO, L., GHARACHORLOO, K., AND BUGNION, E. 1998. Memory System Characterization of Commercial Workloads. In *Proc. Of the 25th International Symposium on Computer Architecture*, (June 1998), 3-14.

BERETSEKAS, D. AND TSITSIKLIS, J. 1989. *Parallel and Distributed Computation – Numerical Methods*, Prentice Hall.

BODNARCHUK, R. AND BUNT, R. 1991. A Synthetic Workload Model for a Distributed System File Server. In *Proc. of ACM SIGMETRICS*, 50-59.

CALZAROSSA, M. AND SERAZZI, G. 1985. A Characterization of the Variation in Time of Workload Arrival Patterns. In *IEEE Trans. On Computers* 34, 2, 156-162.

CALZAROSSA, M. AND FERRARI, D. 1986. A Sensitivity Study of the Clustering Approach to Workload Modeling. In *Performance Evaluation* 6, 1, 25-33.

CALZAROSSA, M., HARING, G., AND SERAZZI, G. 1988. Workload Modeling for Computer Networks. In *Architekture und Betrieb von Rechensystemen*, Kastens, U. and Ramming, F., Eds., Springer-Verlag, 324-339.

CALZAROSSA, M., MARIE, R. AND TRIVEDI, K. 1990. System Performance with User Behavior Graphs. In *Performance Evaluation* 11, 155-164.

CALZAROSSA, M. AND SERAZZI, G. 1993. Workload Characterization: a Survey. In *Proceedings of the IEEE* 81, 8 (August 1993), 1136-1150.

CALZAROSSA, M. AND SERAZZI, G. 1994. Construction and Use of Multiclass Workload Models. In *Performance Evaluation* 9, 4, 341-352.

CALZAROSSA, M., MASSARI, L. AND TESSERA, D. 2000. Workload Characterization – Issues and Methodologies. In *Performance Evaluation - Origins and Directions,* volume 1769 of *Lecture Notes in Computer Science,* Haring, G., Lindemann, C. and Reiser, M., Eds., Springer-Verlag, 459-484.

CARLSON, B., WAGNER , T., DOWDY, L., AND WORLEY, P. 1992. Speedup Properties of Phases in the Execution Profile of Distributed Parallel Programs. In *Modelling Techniques and Tools for Computer Performance Evaluation,* Pooley, R. and Hilston, J., Eds., Antony Rowe, 83—95.

CHAUDHURI, S., AND DAYAL, U. 1997. An Overview of Data Warehousing and OLAP Technology. In *SIGMOD Record* 26*,* 1 (March 1997).

CHINGRUNGRUENG, C., SÉQUIN, C. 1995. Optimal Adaptive K-Means Algorithm with Dynamic Adjustment of Learning Rate. In *IEEE Transactions on Neural Networks* 6, 1 (January 1995), 157-169.

CROVELLA, M. AND BESTAVROS, A. 1996. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *Proc. ACM SIGMETRICS Conf.,* 160-169.

CROVELLA, M. AND LINDEMANN, C. 2000. Internet Performance Modeling: The State of the Art at the Turn of the Century. In *Performance Evaluation* 42, 91-108.

CROW, D. AND SMITH, B. 1992. DB Habits: Comparing Minimal Knowledge and Knowledge-Based Approaches to Pattern Recognition in the Domain of User-Commuter Interaction. In *Neural Networks and Pattern Recognition in Human-Computer Interaction*, R. Beale and J. Finlay, Eds., Ellis Horwood, New York, 39-63.

CUNHA, C., BESTAVROS, A., AND CROVELLA, M. 1995. Characteristics of WWW Client-based Traces. *Technical Report BU-CS-95-010,* Computer Science Dept., Boston University.

CVENTANOVIC, Z. AND BHANDARKAR, D. 1994. Performance Characterization of the Alpha 21164 Microprocessor using TP and SPEC workloads. In *Proceedings of the 21st Annual International Symposium on Computer Architecture,* (April 1994), 60-70.

DAN, A., YU, P., CHUNG, J. 1993. Database Access Characterization for Buffer Hit Prediction. In *Proc. 9th Intl. Conf. on Data Engineering*, Vienna, Austria, (April 1993), 134-143.

DAN, A., YU, P. AND CHUNG, J. 1995. Characterization of Database Access Pattern for Analytic Prediction of Buffer Hit Probability. In *Very Large Data Bases (VLDB) Journal* 4, 1, 127-154.

DILLEY, J., FRIEDRICH, R., JIN, T., AND ROLIA, J. 1998. Web Server Performance Measurement and Modeling Techniques. In *Peformance Evaluation* 33, 1, 5-26.

ELMS, C. 1980. Clustering - One method for Workload Characterization. In *Proceedings of the International Conference on Computer Capacity Management*, San Francisco, Calif., 1980.

EGGERS, S., LEVY, H., LO, J., EMER, J., STAMM, R. AND TULLSEN, D. 1997. Simultaneous Multithreading: A Platform for Next-generation Processors. In *IEEE Micro*, (October 1997), 12-19.

FERRARI, D., SERAZZI, G., AND ZEIGNER, A. 1983. *Measurement and Tuning of Computer Systems*, Prentice Hall, Englewood Cliffs, N.J., (1983).

FERRARI, D. 1984. On the Foundations of Artificial Workload Design. In *Proc. of ACM SIGMETRICS Conf. On Measurement and Modeling of Computer systems*, Cambridge, MA, 8-14.

FU, K. 1974. *Syntactic Methods in Pattern Recognition*, Academic Press.

GAVER, D., LAVENBERG, S., AND PRICE JR., T. 1976. Exploratory Analysis of Access Path Length Data for a Database Management System. In *IBM Journal on Research and Development* 20, 449-464.

GUSELLA, R. 1990. A Measurement Study of Diskless Workstation Traffic on an Ethernet. In *IEEE Transactions on Communications* 38, 9, 1557-1568.

HAN, J., CHIANG, J., CHEE, S., CHEN, J., CHEN, Q., CHENG, S., GONG, W., KAMBER, M., LIU, G., KOPERSKI, K., LU Y., STEFANOVIC, N., WINSTONE, L., XIA, B., ZAIANE, O., ZHANG, S. AND ZHU, H. 1997. DBMiner: A System for Data Mining in Relational Databases and Data Warehouses. In *Proc. CASCON '97: Meeting of Minds,* Toronto, Canada, (November 1997), 249-260.

HAN, J. AND FU, Y. 1995. Discovery of Multiple-level Association Rules from Large Databases. In *Proc. 1995 Int. Conf. Very Large Data Bases,* Zurich, Switzerland, (September 1995), 420-431.

HARING, G., 1983. On Stochastic Models of Interactive Workloads. In *PERFORMANCE '83*, Agrawala, A. and Tripathi, S., Eds., North-Holland , 133-152.

HARMAN, H. 1976. *Modern Factor Analysis*, University of Chicago Press, Chicago, IL.

HARTIGAN, J. AND WONG, M. 1979. A K-means Clustering Algorithms. In *Applied Statistics* 28, 100-108.

HOFMANN, R., KLAR, R., MOHR, B., QUICK, A., AND SIEGLE, M. 1994. Distributed Performance Monitoring: Methods, Tools, and Applications. In *IEEE Trans. On Parallel and Distributed Systems* 5, 6, 585-598.

HOWARD, R. 1960. *Dynamic Programming and Markov Processes*, John Wiley.

HSU, W., SMITH, A. AND YOUNG, H. 2001. Characteristics of Production Database Workloads and the TPC Benchmarks. In *IBM Systems Journal* 40, 3, 2001.

HUDSON, S. AND SMITH, I. 1997. Supporting Dynamic Downloadable Appearances in an Extensible User Interface Toolkit. In *ACM Proc. of UIST '97,* New York, (October 1997), 159-168.

JAIN, R. 1991. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley- Interscience, New York, NY, (April 1991).

JAIN, A., MURTY, M., AND FLYNN, P. 1999. Data Clustering: A Review. In *ACM Computing Surveys* 31, 3, (Sept. 1999), 264-323.

KAMBER, M., HAN, J. AND CHIANG, Y. 1997. Meta Rule-Guided Mining of Multi-Dimensional Association Rules Using Data Cubes. In *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (KDD'97),* Newport Beach, California, (August 1997), 207-210.

KEARNS, J. AND DEFAZIO, S. 1989. Diversity in Database Reference Behavior. In *Proc. of ACM SIGMETRICS and PERFORMANCE '89*, 11-19.

KEETON, K., AND PATTERSON, D. 2000. Towards A Simplified Database Workload for Computer Architecture Evaluations. In *Workload Characterization for Computer System Design*, John, L. and Maynard, A., Eds., Kluwer Academic Publishers, (2000).

KEETON, K., PATTERSON, D., HE, Y., RAPHAEL R., AND BAKER, W. 1998. Performance Characterization of Quad Pentium Pro SMP Using OLTP Workloads. In *Proceedings of the 25th International Symposium on Computer Architecture*, Barcelona, Spain, (June 1998), 15-26.

KLAASSEN, O. 1992. Modeling Database Reference Behavior. In *Computer Performance Evaluation: Modeling Techniques and Tools*, Balbo, G. and Serazzi, G., Eds., North Holland, 47-60.

KLINE, P. 1994. *An Easy Guide to Factor Analysis*, Routledge, London, ISBN 0-415-09490-9.

KRISHNAMURTHY, D. AND ROLIA, J. 1998. Site Walker – A Tool Supporting Performance Characterization and Capacity Planning for Electronic Commerce Systems. In *Proc. of The IFIP International Working Conference on Electronic Commerce '98 (Industrial Track),* Hamburg, Germany, (June 1998).

KOTSIS, G., KRITHIVASAN, K., AND RAGHAVAN, S. 1997. A Workload Characterization Methodology for WWW Applications. In *Proc. of International Conference on The Performance and Management of Complex Communication Networks (PMCCN'97)*, 145-159.

LEWIS, P. AND SHEDLER, G. 1976. Statistical Analysis of Non-Stationary Series of Events in a Database System. In *IBM Journal on Research and Development* 20, 465-482.

LE BLANC, T., MELLOR-CRUMMEY, J., AND FOWLER, R. 1990. Analyzing Parallel Program Executions Using Multiple Views. In *Journal of Parallel and Distributed Computing* 9, 2, 203--217.

LELAND, W., TAQQU, M., WILLINGER, W., AND WILSON, D. 1994. On Self-similar Nature of Ethernet Traffic (Extended Version). In *IEEE/ACM Trans. On Networking* 2, 1, 1--15.

LETMANYI, H. 1985. Guide on Workload Forecasting. In *Special Publication 500-123*, Computer Science and Technology, National Bureau of Standards, Washington, D.C., (March 1985).

LAZOWSKA, E., ZAHORJAN, J., GRAHAM, G. AND SEVCIK, K. 1984. *Quantitaive System Performance: Computer System Analysis Using Queuing Networks Models*, Prentice Hall.

LO, J., BARROSO, L., EGGERS, S., GHARACHORLOO, K., LEVY, H., AND PAREKH, S. 1998. An Analysis of Database Workload Performance on Simultaneous Multithreaded Processors. In *Proc. of the 25th Annual International Symposium on Computer Architecture*, (June 1998), 39--50.

MEHROTRA, K., MOHAN, C., AND RANKA, S. 1997. *Elements of Artificial Neural Networks,* Cambridge, Massachussetts, MIT Press**,** (1997).

MAJUMDAR, S., EAGER, D., AND BUNT, R. 1991. Characterization of programs for scheduling in Multiprogrammed Parallel Systems. In *Performance Evaluation* 13*,* 2, 109-130.

MENASCÉ, D., ALMEIDA , V., AND DOWLY, L. 1994. *Capacity Planning and Performance Modeling: From Mainframes to Client-server Systems*, Prentice Hall, USA, ISBN 0-13-035494-5.

MENASCÉ, D., ALMEIDA , V., FONSECA, R. AND MENDES, M. 1999. A Methodology for Workload Characterization of E-commerce Sites. In *Proc. ACM Conference on Electronic Commerce*, Denver, CO, (November 1999).

NIKOLAOU, C., LABRINIDIS, A., BOHN, V., FERGUSON, D., ARTAVANIS, M., KLOUKINAS, C. AND MARAZAKIS., M. 1998. *The Impact of Workload Clustering on Transaction Routing*, Technical Report FORTH-ICS TR-238*,* (December 1998).

NORTHCUTT, S., MCLACHLAN, D. AND NOVAK, J. 2000. *Network Intrusion Detection: An Analyst's Handbook*, New Riders, ISBN 0735710082, (September 2000).

PAXSON, V. AND FLOYD, S. 1995. Wide-Area Traffic: The Failure of Poisson Modeling. In *IEEE/ACM Trans. On Networking* 3, 3, 226--244.

PENTAKALOS, O. AND MENASCÉ, D. 1996. Automated Clustering Based Workload Characterization for Mass Storage Systems. In *Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies*, College Park, MD, (September 1996).

PITKOW, J. AND PIROLLI, P. 1999. Mining Longest Repeating Subsequences to Predict the World Wide Web Surfing. In *Proc. of USITS' 99: The 2nd USENIX Symposium on Internet Technologies & Systems*, Boulder, Colorado, USA, (October 1999).

PRESS, W., FLANNERY, B., TEUKOLSKY, S., AND VETTERLING, W. 1986. *Numerical Recipes*, Cambridge University Press, (1986), 521-528.

RAGHAVAN, S., VASUKIAMMAIYAR, D. AND HARING, G. 1994. Generative Networkload Models for a Single Server Environment. In *Proc. ACM SIGMETRICS Conf.*, 118—127.

ROHLF, F. 1998. Algorithm 76: Hierarchical Clustering Using the Minimal Spanning Tree. In *The Computer Journal* 16, (1973), 93-95.

ROLIA, J. AND SEVCIK, K. 1995. The Method of Layers. In *IEEE Transactions on Software Engineering* 21, 8 (August 1995), 689-700.

ROSTI, E., SERAZZI, G., SMIRNI, E., AND SQUILLANTE, M. 1998. The Impact of I/O on Program Behavior and Parallel Scheduling. In *Proc. ACM SIGMETRICS Conf.*, 56—65.

SAPIA, C. 2000a. PROMISE: Predicting Query Behavior to Enable Predictive Caching Strategies for OLAP Systems. In *Proc. of the Second International Conference on Data Warehousing and Knowledge Discovery (DAWAK 2000)*, 224-233.

SAPIA, C. 2000b. PROMISE – Modeling and Predicting User Query Behavior in Online Analytical Processing Environments. *FORWISS Technical Report FR-2000-001,* (June 2000).

SCHIFF, D. AND D'AGOSTINO, R. 1995. *Practical Engineering Statistics*, Wiley-Interscience, ISBN 0471547689.

SERAZZI, G. 1981. A Functional and Resource-Oriented Procedure for Workload Modeling. In *PERFORMANCE '81*, Kylstra, F., Ed., North-Holland, 345-361.

SEVCIK, K. 1989. Characterizations of Parallelism in Applications and Their Use in Scheduling. In *Proc. of ACM SIGMETRICS and PERFORMANCE '89*, 171-180.

SHOCH, J. AND HUPP, J. 1980. Measured Performance of an Ethernet Local Network. In *Communication of the ACM* 23, 12, 711-721.

SRIVASTAVA, A. AND EUSTACE, A. 1994. ATOM: A System for Building Customized Program Analysis Tools. In *Proc. of SIGPLAN '94 Conference on Programming Language Design and Implementation*, (June 1994), 196-205.

SMIRNI, E. AND REED, D. 1998. Lessons from Characterizing the Input/Output Behavior of Parallel Scientific Applications. In *Performance Evaluation* 33, 1, 27-44.

TPC: TRANSACTION PROCESSING PERFORMANCE COUNCIL. http://www.tpc.org.

TPCB 1994. TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark B Standard Specification Revision 2.0*, (June 1994).

TPCC 2001. TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark C Standard Specification Revision 5.0*, (February 2001).

TPCD 1996. TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark D Standard Specification Revision 1.2*, (November 1996).

TPCH 1999. TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark H Standard Specification Revision 1.3.0.*

TPCR 1999. TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark R Standard Specification Revision 1.2.0.*

TPCW 2001. TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark W Standard Specification Revision 1.6 (August 2001).*

YU, P., CHEN, M., HEISS, H., AND LEE, S. 1992. On Workload Characterization of Relational Database Environments. In *IEEE Transactions on Software Engineering* 18, 4, (April 1992), 347-355,

YU, P., DAN, A. 1992. Impact of Workload Partitionability on the Performance Coupling Architectures for Transaction Processing. In *Proc. Of the 4th IEEE Int. Symposium On Parallel and Distributed Processing*,, Arlington, Texas, IEEE Computer Society Press, (December 1992), 40-49.

YU, P. AND DAN, A. 1994. Performance Analysis of Affinity Clustering on Transaction Processing Coupling Architecture. In *IEEE Transactions on Knowledge and Data Engineering* 6, 5, (October 1994), 764-786.

ZAÏANE, O., XIN, M. AND HAN, J. 1998. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In *Proc. Advances in Digital Libraries Conf. (ADL'98)*, Santa Barbara, CA, (April 1998), 19-29.