MEASUREMENT-BASED APPROACH TO WORKLOAD CHARACTERIZATION[1]

Maria CALZAROSSA, Luisa MASSARI
Dipartimento di Informatica e Sistemistica
Università di Pavia
I - 27100 Pavia (Italy)
E-mail: {mcc,massari}@gilda.unipv.it

Abstract:

Measurements are of fundamental importance for performance evaluation since they provide a "picture" of the behavior of the systems and of their load. Experimental approaches to workload characterization are based on the application of techniques for the interpretation and the analysis of measurements. A survey of the methodologies for modelling the load of various types of systems, namely, interactive, distributed and parallel systems, and databases, is given on a few case studies.

1 Introduction

Measurements play a key role in all the studies aimed at assessing the performance of any type of system. Although experimental approaches are typically recommended, much more care is usually devoted to the development and the application of sophisticated techniques for system modelling than to the analysis of the workload.
Measurements represent the basis for workload characterization [FSZ83],[CS93]. Their accuracy reveals particularly important since the workload is one of the major components which determine system performance.
A measurement process can be very complex. The appropriate instrumentation for collecting the information of interest is required. Suitable analysis techniques have to be used for the interpretation of the large amount of measured data and for deriving the workload parameters.

[1]This work was supported in part by the the Italian Research Council (C.N.R.) under Grants N. 93.01592.PF69 and N. 93.01869.CT12 and by the Italian M.U.R.S.T. under the 40% Project.

A compact representation, that is, a workload model has to be obtained through the application of techniques, such as, statistical analysis and numerical fitting, stochastic processes, probabilistic graphs.
Note that the quality of the characterization directly affects the quality of the performance results obtained. Indeed, the input of system models has to be driven by a workload model. The same applies to the load generated for benchmark experiments.
Depending on the system architectures and on the application environments, the characteristics of the workloads may vary either from a functional viewpoint and for their impact on the physical resources of the systems. However, a few commonalities for the construction of workload models of different types of systems are identified. This means that the same analysis techniques can be applied in different frameworks and to different parameters. A correct interpretation of the obtained results has then to be performed.
The paper addresses the measurement-based approach to workload characterization. Various types of systems, namely, interactive, distributed and parallel systems, and databases, are analyzed and a survey of the methodologies adopted for describing their loads is presented on a few case studies.
The paper is organized as follows. Section 2, which focuses on interactive systems, discusses the applications of various techniques to represent either the static and dynamic characteristics of the load. Methodologies for monitoring distributed systems and for the analysis of the measurements are described in Section 3. The characterization of parallel workload is presented in Section 4. In Section 5 a system–independent representation of the load of databases is given. Finally, a few conclusions are drawn in Section 6.

2 Interactive Systems

Workload characterization for interactive systems has been approached for many years because of their widespread diffusion. Such studies have benefitted of the extensive experience and of the results in the field of batch systems (see e.g., [Ferr72], [AMB76], [Bard76], [Sera81], [BCMS82]).
In the case of batch systems, the jobs were sequentially processed without any interaction with the users. The load has been described from a static viewpoint by applying classification techniques, such as clustering [Hart75]. Groups of jobs with homogeneous resource consumptions were identified. A few representative components were then selected for benchmarking and system modelling purposes.
In the case of interactive systems, the focus has been shifted to the analysis of the workload from a dynamic viewpoint. The time–varying characteristics of the load together with the user–system interactions have to be taken into account. Graph–based, numerical fitting and statistical techniques, clustering and stochastic processes have been extensively applied for building workload models (see e.g., [Hari83], [CS85], [RK85], [CF86], [Sera86], [CMT90], [SFMF93], [CS94]). As we will see in what follows, such techniques are the basis for workload characterization studies of any type of system.

Exploratory analyses of measurements of interactive loads represent the starting point for any type of workload characterization in that they help in highlighting patterns or trends in the measured data. In [SCCQ86], such an approach has been adopted for describing the behavior of interactive users during an observation interval of about eight hours. Measurements have been collected by means of accounting routines which, in the case of interactive systems, are a good source of experimental data. From the arrival time of each of the 3915 commands, their interarrival times have been derived and the basic statistical indices have been computed (see Table 1).
A more detailed description is obtained by subdividing the 3915 commands into 20 non–overlapping sub–sequences of 200 commands each (except the last one). Table 1 presents the statistics of three such sub-sequences and of the global one. As can be seen, there is a large variability either within the global sequence and between different sub–sequences. Hence, an assumption of stationarity in the interarrival times is not appropriate. The same applies to the arrival rate function.

Statistics	Global Sequence	Sub_seq1	Sub_seq2	Sub_seq3
Mean	7.04	0.6	4.7	2.5
Coefficient of variation	22.91	1.7	1.9	2.2
Skewness	40.65	6.8	4.4	5.6
Kurtosis	1779.8	67.9	25.7	35.7
50th percentile	0.6	0.3	1.5	1.1
95th percentile	7.	2.1	18.7	6.8
99th percentile	23.1	3.2	37.3	36.5

Table 1: Basic statistics of the interarrival times (expressed in seconds) for the 3915 commands and for three of the obtained sub–sequences

Numerical fitting techniques can then be applied in order to obtain a compact representation of the analyzed phenomenon. Figure 1 shows the plot of the arrival rate function of the commands for an high activity sub–period of two hours. The dots refer to the measurements. The solid curve, obtained by applying fitting techniques, is an exponential polynomial of degree 8. Fluctuations in the arrivals are observed. Such fluctuations can also be observed in Figure 2 where the cumulative distribution of the number of arrivals as a function of time is shown. The dotted curve refers to measured data; the solid curve represents the "theoretical" cumulative function corresponding to stationarity conditions. The number of arrivals is below the average for the initial 49 minutes, then it goes above for about 54 minutes and, finally, it remains close to the average.

A stochastic model of the user behavior, particularly suitable for constructing scripts for benchmark experiments, is proposed in [Hari83]. The composition of user sessions is analyzed from a hierarchical viewpoint. The load consists of multiple runs which can be seen as sequences

of tasks involving different software modules (e.g., compilers, linkers, editors). The tasks are sequences of statements characterized by their usage of physical resources. Clustering techniques together with tests of the order and the homogeneity of Markov chains are applied for identifying sub–workloads and the types of dependencies existing within each sequence.

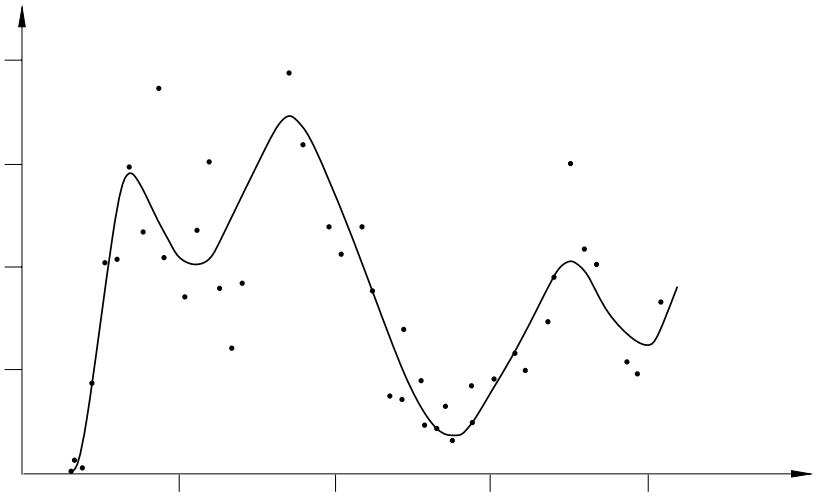

Figure 1: Arrival rate function of the commands [SCCQ86]. The dots and the solid curve refer to the measurements and to the exponential polynomial, respectively

An alternative approach for describing the user behavior from static and dynamic perspectives is presented in [RK85]. Such representation is based on the analysis of how the users interact with the system from a logical viewpoint. A "mode" is first defined as a functional group of commands issued by a user during a session. The mode probability vector, that is, the number of visits to each mode, and the mode transition matrix, that is, the transitions between different modes, are derived for each user. The entropy and mobility are then computed from such parameters. The tuple of these two indices represents a comprehensive classification of the users. The entropy gives a static measure of the order/disorder in the user behavior. The mobility provides a dynamic characterization in that it is a function of the mode transitions. The identification of typical mode sequences together with a system–dependent description of the commands in terms of their resource consumptions can then be used for reproducing the behavior of the users.
Extensive measurements collected in a university environment have been considered for analyzing such behaviors. From 390 sessions, four main categories of users, namely, "casual", "data preparation", "program development" and "advanced" users, are identified. The analysis has shown that values of entropy and mobility close to zero are typical of "casual" users who issue very few command types. Advanced users are characterized by larger values of both indices.

User behavior graph (ubg), introduced in [Ferr84], is a compact representation of the sequence of commands submitted by a user which models and reproduces its static and dynamic char-

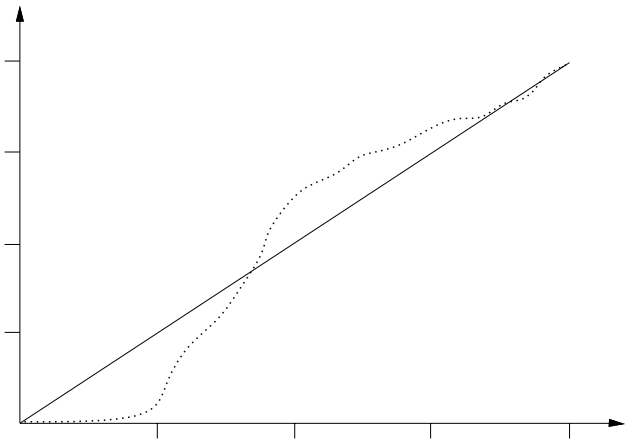

Figure 2: Cumulative distribution of the number of arrivals in a two hours interval [SCCQ86]. The dotted and the solid curves refer to the measured and theoretical functions, respectively

acteristics. The nodes of such graph are the different command types and the arcs, with their associated probabilities, denote the possible transitions among commands. Figure 3 shows an example of ubg consisting of four nodes.
In [CF86], a model of the load based on a ubg representation is adopted for driving the input of a system model used for assessing the sensitivity of the clustering approach to workload characterization. Measurements of 60 users of an interactive computer have been collected from the standard UNIX accounting routines. The commands are initially grouped on a functional basis, that is, according to their type and the corresponding ubg, consisting of 40 nodes, is constructed. Clustering is then applied to the commands described on a physical basis, that is, according to the CPU time consumed, the memory space required and the number of disk I/O blocks transferred, and a more manageable ubg with fewer number of nodes, namely, eight, is obtained.

Recently, workload models of interactive systems have been refined and specialized for taking into account the evolutions in the architectures and in the operation modes. Interactive users have shifted from single–window terminals connected to time–sharing systems to workstations, X–terminals and personal computers characterized by enhanced graphical interfaces based on multiple windows.
Some sort of parallelism is introduced in the user behavior. The workload is no longer strictly sequential since multiple windows can be simultaneously open and many commands can be simultaneously "active".
The problem of reproducing the behavior of a user in a multi–window environment is addressed in [ACRS94]. A workload model consisting of a generative and an executable component is proposed. The generative model provides a system–independent description of the load by considering the user behavior from a logical viewpoint. Figure 4 shows an example of multi–window workload.

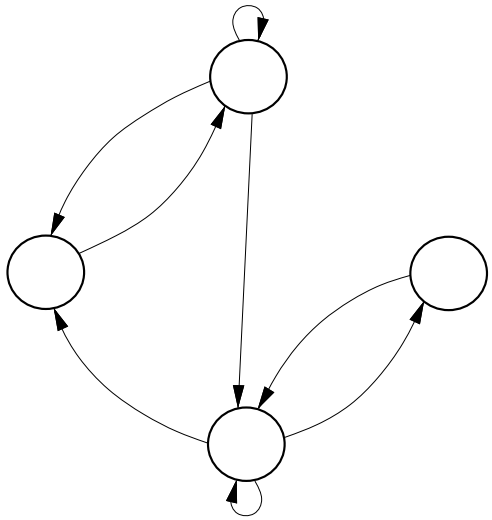

Figure 3: User behavior graph with four nodes

As can be seen, a session is characterized by multiple activities which consist of streams of commands issued by the user from different windows. One or more parallel activities can be generated from a sequential one.
Once the probabilistic description of the sequential as well as of the parallel behavior of each stream is obtained, system–dependent characteristics of the underlying architecture have to be introduced in order to build the executable workload model. The interarrival times of commands belonging to a sequential stream or to parallel streams and the resource usages are examples of the parameters added to the logical description for exploiting the physical parallelism.
Note that these multi–window workload models can be used for describing the input of system models based on simulation and for generating the load of benchmarking experiments of centralized and distributed systems. Indeed, a user can open windows on his local workstation as well as on remote ones connected via networks.
The measurements required for building a multi–window workload model come from various sources. Accounting routines provide only a few of the parameters (e.g., CPU time, number of disk I/Os, start time). The events generated from the various windows (e.g., open, close) have peculiar characteristics and have to be detected and collected by ad hoc monitoring tools based on X–Window system. Furthermore, measurements of distributed systems are also required.

3 Distributed Systems

The large number of hardware and software components of computer networks and distributed systems require appropriate performance management facilities and integrated measurement techniques. An efficient and scalable measurement system is particularly necessary for large

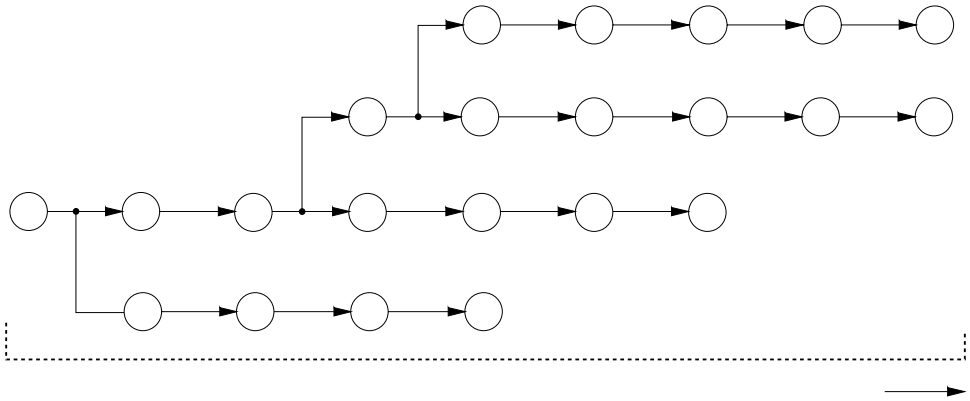

Figure 4: Example of user session consisting of multiple sequential streams of commands, that is, activities [ACRS94]. The circles denote the commands

scale environments.
Distributed systems are ill–suited to monitoring activities [Jain91] because of the heterogeneity of the components and of the synchronization and communication constraints. Measurements, collected on each local node, need to be assembled and matched together as to obtain global information on the system performance.
As already pointed out, measurements are fundamental for any type of workload characterization, which, in the case of distributed systems, becomes quite complex because of the lack of adequate measurement facilities. For example, the types and the patterns of the requests to a file server from the clients or the flow of packets on a local area network are accurately described by measurements. Hence, a systematic monitoring approach for distributed and client/server environments as well as for computer networks has to be defined and applied in any performance study.

Recently, a software architecture for a Distributed Measurement System (DMS) for performance monitoring of heterogeneous distributed environments has been proposed [FMSC94]. DMS measures core and extended services and distributed user workload instrumented with macros. A correlated view of resource consumptions across network nodes is provided.
The DMS has to collect heterogeneous data from such nodes in a consistent manner and with a minimum overhead. Therefore, a common set of performance metrics and instrumentation has to be defined.
Figure 5 shows the architecture of the DMS. Various levels have been defined, in order to provide increasing details in the measurements. For the purpose of this paper, the lowest levels, which are aimed at data collection, are of interest.
The bottom level consists of sensors, which are mechanisms inserted in-line into software services, such as, standard libraries or user codes. Sensors implement the actual instrumentation function by computing and storing primitive statistical quantities, such as, counts and sums, and specialized data, such as, server residence times at the processor.

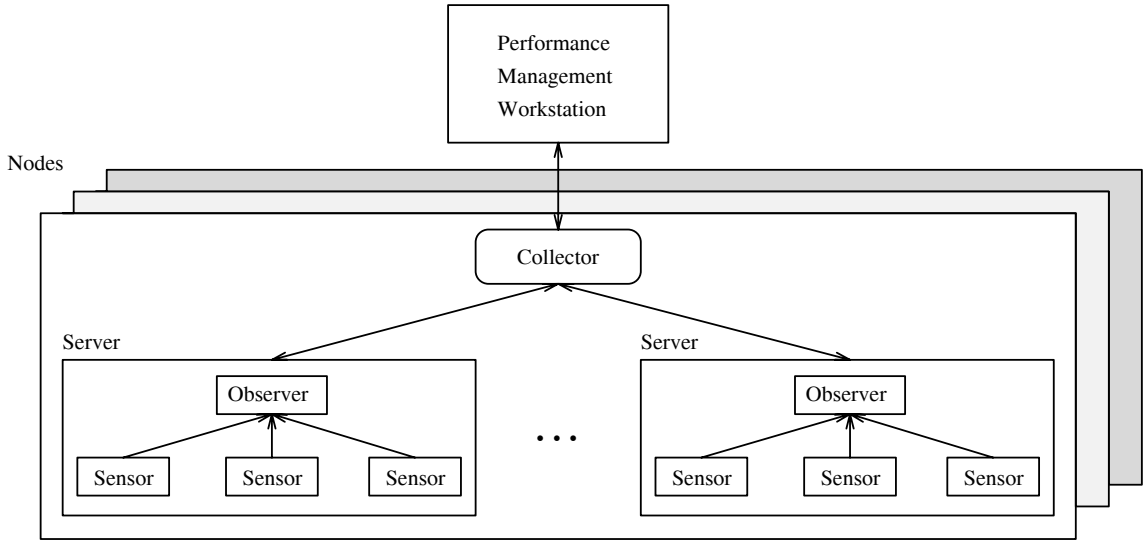

Figure 5: Architecture of the Data Measurement System [FMSC94]

As can be seen from Figure 5, multiple servers (or clients) can reside on each node of the distributed system. Default sensors and, eventually, additional custom sensors which record specific behaviors, such as, cache operations and queue lengths, are associated to each server. Sensors can also accumulate and integrate, within DMS, measurements coming from other subsystems, such as operating systems.
The next level is represented by the observers, which gather raw data from all the individual sensors in the local server. There is one observer for each server. The first level of communication within the DMS is between the sensors and observer, the major elements of the architecture, which are part of the instrumentation.
The collectors, which operate as storage units, control the observers and perform data management operations on each network node where a single collector, multithreaded for concurrency, resides. The measurements, accumulated from all the observers of a node, are then transmitted to the Performance Management Workstation (PMW), which is a centralized component of the DMS.
Statistical techniques for the analysis of the measured data, graphical routines for their visualization, access to the information repository as a common database, and an interface for controlling the system parameters from a performance perspective are provided within the PMW.

Once the measurement methodologies have been defined and monitoring tools are available, the data collection can start. Hence, appropriate techniques for the identification of the parameters representing the workload and for its characterization have to be applied.
Various experimental studies appeared in the literature (see e.g., [SH80], [Gons85], [Sera86], [FV90], [BSST91], [ANLC92]) focus on the analysis of different networks with the aims of investigating the behavior of the traffic and its possible patterns.

In [Guse90], measurements for a 24 hours period on a large Ethernet connecting diskless workstations to file servers, are analyzed. The packet length, the interarrival time between packets, the source and the destination nodes are examples of parameters collected with ad hoc monitoring tools and used to characterize the network traffic as a function of the protocols adopted. Figures 6 and 7 show the network utilization, that is, the fraction of time during which transmitters are active. As can be seen from Figure 6, the overall utilization seldom exceeds 20%. The TCP protocol generate very low utilization with a peak around 10pm corresponding to file system dumps (see Fig. 7).

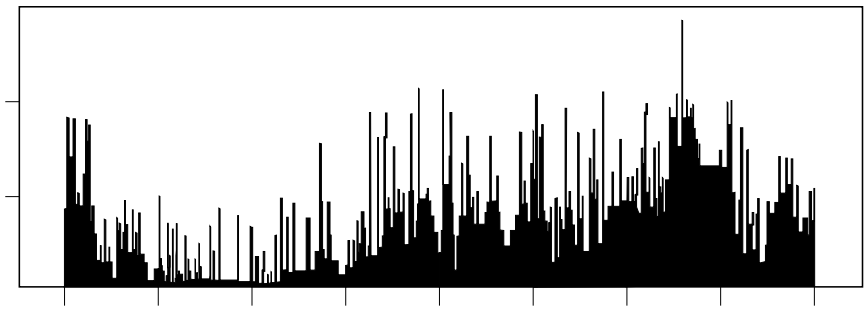

Figure 6: Ethernet utilization over 1 minute intervals [Guse90]

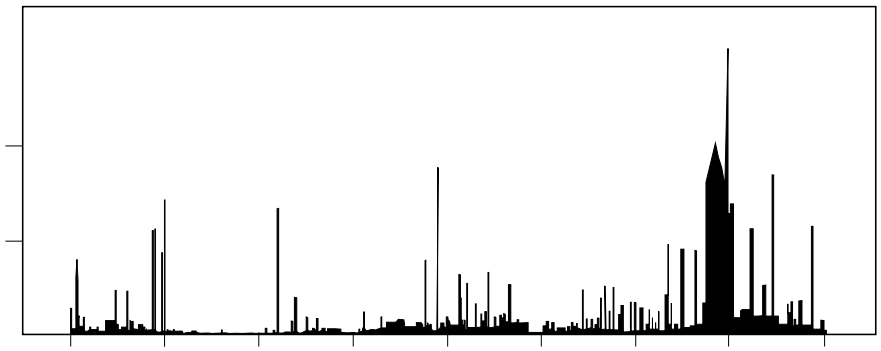

Figure 7: Ethernet utilization for the TCP protocol over 1 minute intervals [Guse90]

Measurements of the load of a distributed system file server in a UNIX/NFS environment are analyzed in [BB91] with the objective of constructing a distribution–driven synthetic model. Workload characterization is the basis for such model in that it helps in discovering patterns in the data which lead to model simplifications. The frequency distribution of file server requests, the request interarrival time distribution, the file referencing behavior and the distribution of sizes of read and write request are the parameters selected for characterizing the workload.
A study of the workload of high-speed networks is described in [Gun94]. Such characterization

relies on simple measurements taken at the access point of the network. Initially, three parameters, namely, peak and mean rates of a connection, and mean duration of a burst period, have been considered for describing the traffic. These parameters do not completely capture the actual behavior of the traffic where idle periods (with transmission at zero bit rate) follow burst states (with transmission at peak rate) of arbitrary lengths and distributions. A new parameter, namely, the equivalent burst length, which captures all such effects, is introduced. Various examples indicate the accuracy of such approximation.

As in the case of interactive systems (see Sect. 2), the workload of distributed environments can be hierarchically decomposed. Measurability is a guideline for choosing the levels of the hierarchy. Workload models have to take into account such structure by using techniques and parameters typical of each level. Similar hierarchical approaches to workload characterization has been proposed in [CHS88] and [RVH94].
In [CHS88], the layered structure of the workload is defined by considering the logical subdivision of the hardware components of distributed systems into three groups, that is, user terminals, processing nodes and communication subsystem. The load submitted by the users in the form of requests can be executed by the local processing node or may require services from remote nodes, thus generating messages towards the network. Figure 8 shows the components of the three layers. Two paths followed by local and remote requests are identified (see the dashed lines).

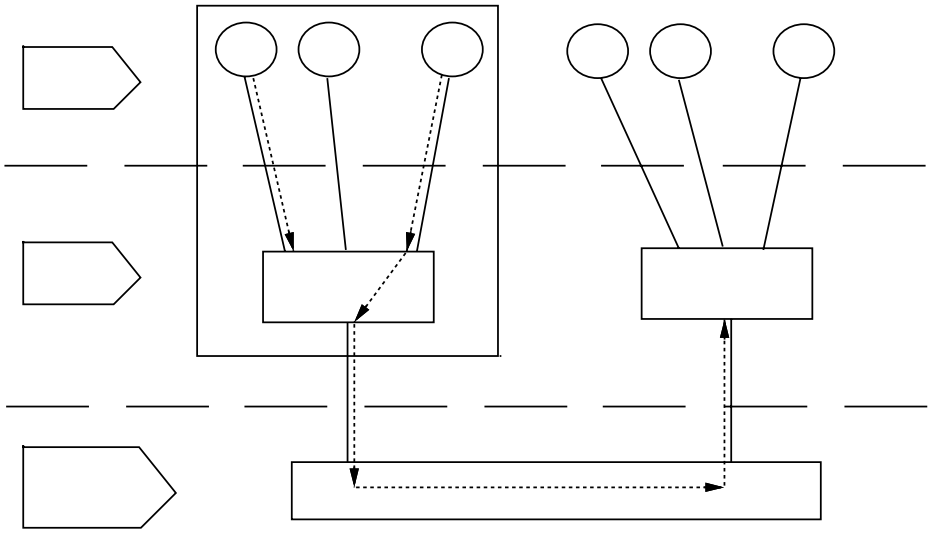

Figure 8: Layered structure for workload characterization of distributed systems [CHS88]

Probabilistic graphs (see Sect. 2) are used for reproducing the dynamic behavior of the various types of "requests". The messages flowing on the network are also characterized by their traffic flow matrix. Different physical resources are involved and different parameters are identified. The message length and type are examples of parameters of the network layer. The hardware and software resource consumptions are used for characterizing the load of the processing nodes. The methodology has been applied for describing the user behaviors and the load of workstations and time–sharing systems which generate traffic on an Ethernet local area network.
In [RVH94], a methodology for describing the networkload, that is, the load generated by all the clients of a network, is proposed. Such hierarchical approach, which differs from the previous one because it is closer to a client/server perspective, identifies three levels by observing the load generation patterns. Figure 9 shows such layered structure. The session is considered at

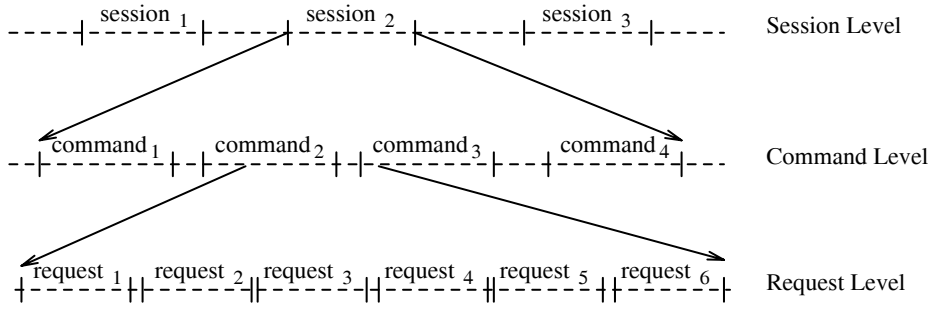

Figure 9: Layered structure for networkload models [RVH94]

the highest level; the commands issued during a session represent the next lower level. The bottom level consists of the requests which deal with the client-server interactions. For example, in systems with fileservers and diskless clients, a user session is a sequence of commands which generate requests to the local client for accessing remote files residing on the fileserver.
The workload at the session level consists of a sequence of sessions. A client can generate multiple parallel activities involving external resources (servers) which provide various types of services, e.g., file services. The workload is described by the following parameters that can be measured through accounting facilities:

- session duration, which represents the busy period of a client, is the time between logon and logoff;
- session interarrival time, which represents the idle time of a client, is the time between the end of one session and the beginning of the next one.

At the command level, the sequence of events which compose a session and deal with the command executions and think times is considered. Various classes of commands, which load the server in different ways, can be identified. For example, commands that access the file server just for file downloading are characterized by different patterns of requests than commands which also use the file server for the creation and the deletion of temporary files.
The parameters typical of the command level are:

- size of the accessed files;
- number of the file accesses.

The request level is the lowest measurable level which directly represents the load at the server due to all clients of the network. The measurements have to be carried out by using monitoring tools, able to capture the flow of packets in a given observation interval. The parameters to be derived are:

- request generation rate of the clients, that is, the number of requests generated by the clients per unit of time;
- service time of the server, that is, the time taken by the server for satisfying the request generated by a client.

The layered approach has been applied in an experimental study aimed at characterizing the load of a network with personal computers and one server. Measurements are collected using the accounting utilities, as well as a special command filter and a network sniffer. The parameters specific of each level are derived from the collected data and clustering techniques are used for identifying classes, that is, SessionGroups, CommandClasses and RequestTypes.
Sessions are classified in 5 classes, based on their frequency of occurrence, their loading density and the probability of write operations. Table 2 summarizes a few statistical values for three classes which account for more than 80% of measured sessions.

Class	Mean Duration	Loading Density [req/min]	Probability of Write	Percentage of Sessions
1	10.7281	291.598	0.124062	52.874
2	42.8571	132.013	0.117344	22.879
3	159.771	26.675	0.924321	5.985

Table 2: Statistics of three of the five classes identified from the measured sessions [RVH94]

The request level description contains details about the generation rate and the service time of the requests to the fileserver. The experimental environment supports more than 50 types of requests, but only 10 of them account for more than 95% of the time.

4 Parallel Systems

Parallel applications consist of multiple processes (or tasks) [FJLO88] interacting together, allocated to different processors, and characterized by dependencies, synchronization constraints, and precedence relationships arising from the distribution of the data and of the execution among distinct processors.

Performance of parallel applications can be limited by such factors as well as by the system architectures. Hence, an accurate characterization of the workload is required for studies such as tuning, performance debugging and compiler and operating system designs. For example, the characterization of the applications can be used by parallel compilers for the specification of automatic data partitioning strategies which minimize communications [CFZ93].
Let us remark that the identification of the parameters and metrics typical of parallel applications and able to capture their peculiar characteristics needs special care. In what follows, a classification of the most relevant parallel parameters is given. A more detailed survey can be found in [CMM94].
Task graphs are of common use for describing a parallel application; the nodes of the graph represent the tasks, and the arcs represent the precedences among them. An example of task graph for an LU decomposition algorithm is reported in Figure 10.

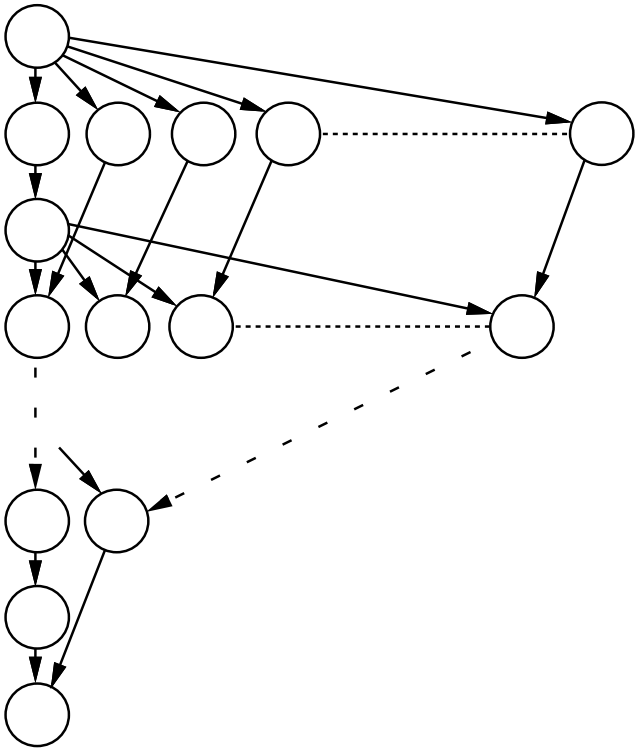

Figure 10: Task graph for an LU decomposition algorithm [CS93]

A task graph can be described by a set of parameters characterizing its structure and independent of the system architecture. Such parameters, which reflect the parallelism inherent in the application, are classified as "static" [BT89] since they can be simply obtained from the analysis of the task graph without any execution of the application.
The total number of tasks the application consists of is a parameter related to its granularity. The in-degree and the out-degree of a node are the number of nodes that are direct predecessors of that node and for which the particular node is a direct predecessor, respectively. In order to

characterize all the tasks, the average in-degree and out-degree of the task graph are computed from the values of each individual node. The standard deviation can also be derived. The in-degree is related to the synchronization complexity of the application. A large value of the in-degree parameter means that many nodes require synchronization activities, hence inducing large overhead. Both the values of the in–degree and out-degree of the graph of Figure 10 are equal to 1.85; they reflect the high interconnection existing among nodes.
The depth, or critical path, is the longest path (in terms of number of nodes) that starts at the input node and ends at the output node. The execution time of an application is influenced by the value of such parameter; indeed, the length of the critical path gives the minimum completion time for the application, and it is attained when there are no constraints on the number of available processors.
The maximum cut is the maximum number of arcs taken over all possible cuts from the input node to the output node. A cut between two nodes is a set of arcs whose removal disconnects the two nodes. This parameter reflects the maximum theoretical parallelism that can be achieved by the application.

Information, dealing with computation, communication and synchronization activities, are derived from real measurements using ad hoc monitoring tools (see e.g., [KS87], [AL90], [LS92]). The tasks of a parallel application may be executed on different processors with alternating phases of computation and communication. Hence, measurements of the communication and the synchronization activities between tasks, which highly influence the performance, have to be collected for workload characterization purposes. The I/O operations as well as the time spent for the allocation of the processors to the tasks, which represents the overhead introduced by the operating system into the global execution time of the application, are also collected.
All these low–level measurements need some sort of processing for deriving high–level information to be directly used for performance studies or to be further analyzed by means of visualization and statistical techniques.
Such parameters are referred to as "dynamic" [RSL91], [RS94] in that they describe the behavior of the application during its execution. They can be represented by single values or by curves. The former is determined by a single run of the application with a given number of processors. Table 3 summarizes a few of such parameters.

Execution time	Communication time
Computation time	Synchronization time
Volume of data exchanged	Number of communication requests
I/O time	Number of I/Os

Table 3: Examples of single value parameters

A more complete characterization of the behavior of a parallel application is derived by considering multiple runs of the application with different number of processors and by representing

the parameters as a function of the number of allocated processors (signatures), or of the execution time (profiles).
When a limited number of processors is available, execution profiles describe the number of active processors as a function of time. Identification and analysis of phases in execution profiles have been investigated in [CWDW92]. Figure 11 illustrates an execution profile, in which distinct phases can be recognized. Identification of phases may be useful when an application needs to repartitioned, i.e., remapped to a different set or number of processors, or when it has to be checkpointed periodically. Phases in the execution with small changes are adequate points for restarting the application.

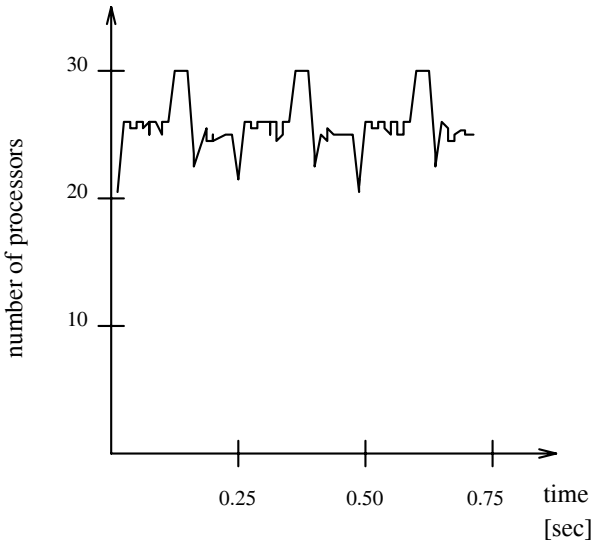

Figure 11: Execution profile with distinct phases [CWDW92]

Profiles can also be represented by means of non–increasing or non–decreasing functions under the assumption that the distribution of processor utilization is maintained. In this case, we obtain execution shapes.
The execution signature expresses the total execution time of an application as a function of the number of allocated processors. In general, the computation and the communication signatures, which are monotonically increasing and decreasing functions, respectively, are identified. Three important dynamic indices [EZL89] are speedup $S(p) = \frac{T(1)}{T(p)}$, efficiency $E(p) = \frac{S(p)}{p}$, and efficacy $\eta(p) = \frac{S^2(p)}{p}$, where $T(1)$ and $T(p)$ denote the time needed for the execution of an application on a single processor and on p processors, respectively.
Speedup represents the gain in the performance of the parallel application executed with p processors with respect to the serial execution. The efficiency is a measure of the fraction of time the processors are busy. Efficacy describes how well processors are used. It helps in the identification of the processor working set (pws) [GST91], defined as the number of processors which attains the maximum of $\eta(p)$. pws gives an optimal system operating point, since it expresses the maximum ratio between the benefit in increasing the number of allocated processors and the corresponding costs due to the communication overhead.

Figure 12 shows the speedup and efficacy curves obtained for a matrix multiplication algorithm [GST91]. As can be seen, the value of pws is equal to 3. Thus, an allocation of 3 processors leads to a maximum efficacy of the algorithm.

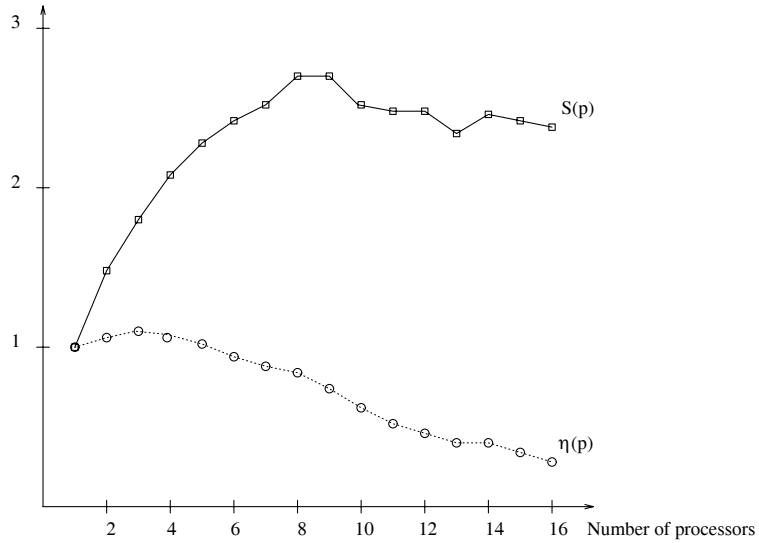

Figure 12: Speedup $S(p)$ and efficacy $\eta(p)$ as a function of the number of allocated processors [GST91]

From the profiles it is possible to derive several statistics, such as, the average, the minimum and the maximum number of busy processors during the execution of an application [Sevc89]. Sensitivity analyses of the parameters adopted for the description of the workload are very important for determining their influence on the application performance. In [Mass94], the execution profiles have been approximated by means of their averages. The study shows that profiles characterized by small coefficients of variation are well represented by average parallelism, in that the errors induced in the execution time of the application are negligible. For larger coefficients of variation, these errors increase. For example, for a profile with a coefficient of variation equal to 0.76 and an average parallelism of 17, the error obtained is 9.2%.

The main problems within the experimental approach deal with the analysis and the interpretation of the large amount of the measurements. Some sort of pre–processing is required for deriving the parameters to be used for workload characterization. Through the application of visualization and statistical analysis techniques, a compact representation of the behavior of the workload is obtained.
Note that the techniques employed for describing parallel applications are standard workload characterization techniques used with different perspectives and meaning. For example, the applications are usually considered in isolation, that is, they are characterized one at a time, and performance statistics for the individual tasks are obtained (see, e.g., [RCG72], [LA87], [Sevc89], [MEB91]).
In [MW94] a workload characterization study aimed at evaluating whether overlapping of com-

munication and computation activities could improve the performance of an application, with respect to different architectures, has been carried out. A message-passing benchmark code testbed, instrumented using the PICL library [Worl92], is considered. The measurements related to the computation and communication activities are collected and the execution, computation, communication, receive and transmit times are selected as representative parameters for the workload characterization process. Statistical analysis and visualization techniques, provided within the MEDEA (MEasurements Description Evaluation and Analysis) tool [CMMP94], are applied. Parallel metrics, such as, profiles, are obtained and used for a preliminary characterization of the different experimental runs. The statistical properties of the measurements are examined by means of cluster analysis, which provide a classification of the tasks as function of their resource usages, and of functional description, which provide insight into the logical composition of each cluster.

5 Databases

Databases have increased their popularity in the framework of either centralized, distributed and parallel systems. Hence, in such a complex scenario the performance requirements become more and more stringent and critical.
A better understanding of the workload of database systems is needed for effective tuning and design activities which have to lead to performance improvements. For example, for building benchmark workloads to be used for evaluating alternative hardware and software design tradeoffs of such systems, a deep knowledge of the behavior and of the characteristics of the load is required. Furthermore, the workload must be strictly related to the production environment where the database is used.
The workload of a database is usually considered from a hierarchical viewpoint [AKB87], [Klaa92]. Depending on the objectives of the study, various levels are identified. For example, the top level, that is, the user level, is represented by the applications characterized by the functions they perform. The next lower level considers the transactions and the statements which implement the applications. Finally, the bottom level deals with the physical resource usages that are functions of the underlying architecture (e.g., distributed, parallel systems) and of the storage organization adopted.
Note that in this section the analysis of the database workloads will be independent of the underlying architectures and thus applicable to various types of environments.

Early studies on IMS databases [GLP76], [LS76] provide a detailed characterization of the workload in terms of data sequences that can be measured and identified at various levels. Sequences of transactions, of segments, of blocks and of access path lengths (i.e., number of segments accessed) are studied by means of statistical analysis of series of events, which provides a graphical and mathematical description of the behavior of a database. Such analyses take into account the individual sequences as well as the transformations among them. In particular, the arrival rate of the transactions has been represented by means of nonhomogeneous Poisson processes.

Measurements of IMS environments collected with the aim of analyzing reference patterns are also presented in [KDF89]. Four transactions have been selected and their average locality and run length distributions were derived. A certain stability in the references has been found.

In the case of relational databases, measured events refer to subsystem activities, buffer manager I/O requests and lock information as well as to SQL statements. The analyses of the structure and complexity of SQL statements, and the makeup and behavior of transactions and queries and the composition of relations are part of the workload characterization process.
The typical parameters of a transaction are the number of SQL statements of each type, the number of tuples scanned and retrieved/updated/inserted/deleted, the number of pages accessed, the length (i.e., the number of different pages accessed) and the execution time for each transaction invocation. In the case of a more detailed characterization, that is, at the SQL statement level, parameters related to the start and end times of each statement, its type (e.g., OPEN, SELECT, DELETE) and its number have to be collected. The description of each statement can be further refined from a functional viewpoint. The numbers of relations involved, of columns selected and of predicates appearing, the existence of subqueries and the use of aggregate functions are also to be taken into account.

A few experimental studies, where the behavior of the parameters above identified has been analyzed, will be described in what follows. As we will see, a few parameters exhibit high variability which depends either on the database and on the transaction characteristics.
Parameters, such as, page accesses, write transactions, pages written/accessed, have been derived by the six traces collected on a CODASYL database running an insurance company application [Klaa92]. The study was aimed at characterizing the page reference behavior. Each trace covers a time interval ranging from 10 to 18 minutes. Figure 13 shows the distribution of the transaction length which is clearly non-uniform.
In [YCHL92], an experimental study of a DB–2 like database system running accounting type applications is presented. Such study shows that, due the complex and variable nature of the load of relational databases, an accurate characterization is required. Hence, a software tool, REDWAR (RElational Database Workload AnalyzeR), based on the DB–2 runtime tracing facilities, has been developed for the analysis and the classification of measured data.
Simple statistical analysis techniques have been adopted for identifying, from a logical perspective, the composition of the transactions and of the statements.
The average and maximum numbers of statements collected for the 305 transactions measured in a peak period of two hours are equal to 17 and to 551, respectively. Figure 14 shows the distribution of the number of SQL statements per transaction. As can be seen, approximately 70% of the transactions contain less than 10 statements and very few transactions are characterized by more than 100 statements which lead to a different impact on system resources.
The analysis of runtime traces provides information for a classification of static and dynamic SQL statements as a function of their type. A global number of 5381 static and of 459 dynamic statements has been collected during the measurement interval.

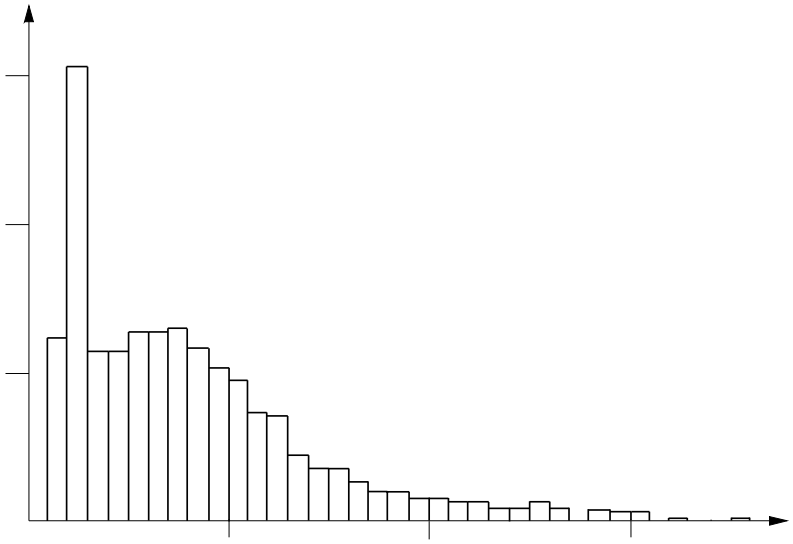

Figure 13: Distribution of the transaction length [Klaa92]

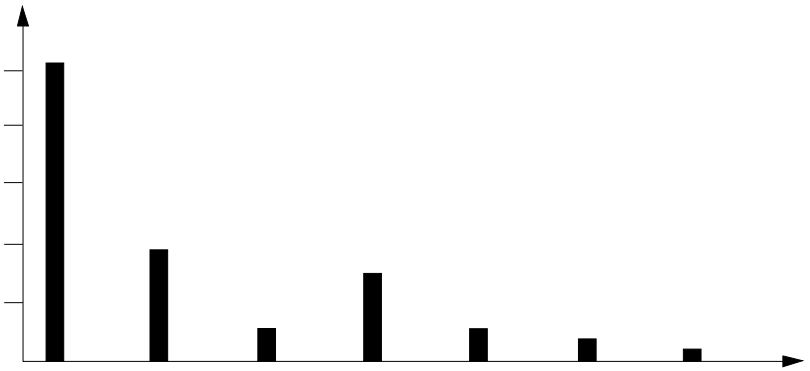

Figure 14: Distribution of the number of SQL statements in a transaction [YCHL92]

Figures 15 and 16 show the distributions of the types of static and dynamic SQL statements, respectively. About 30% of static statements are data manipulative–type (i.e., SELECT, DELETE, UPDATE, INSERT). In the case of dynamic ones, such percentage increases up to 96%.
The analysis of the statements can be further refined as to obtain a classification based on their structural characteristics. For example, the data manipulative statements can be grouped according to the type of clauses or subqueries used (e.g., SELECT with WHERE Clause).

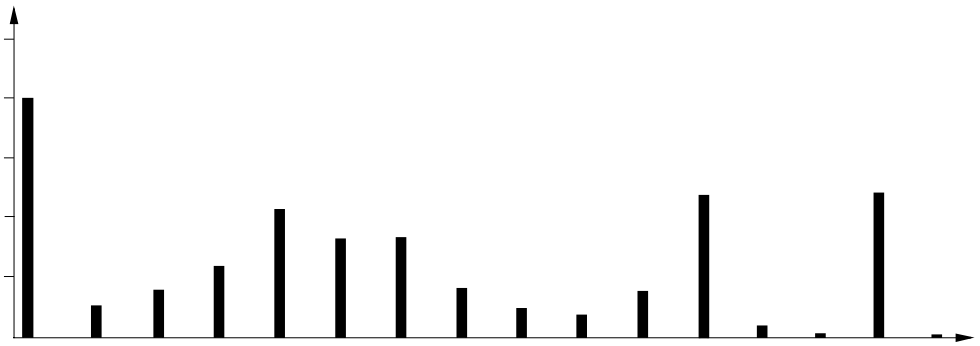

Figure 15: Distribution of static SQL statements as a function of their types

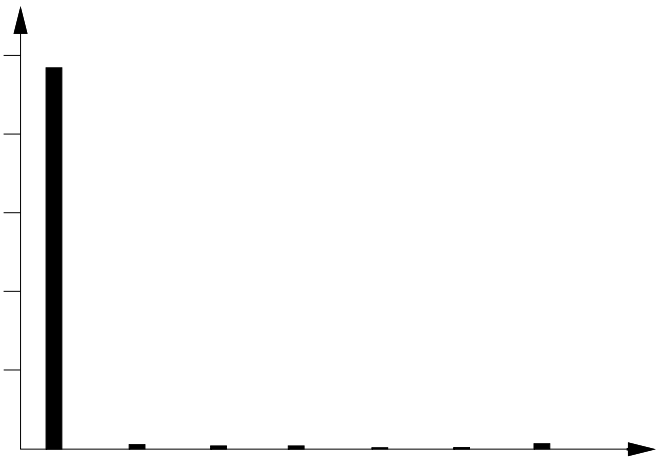

Figure 16: Distribution of dynamic SQL statements as a function of their types

From the analysis of traces collected in real production environments, we can conclude the importance of the characterization of database workloads performed either from functional and from physical perspectives.

6 Conclusions

Experimental approaches to the characterization of the workload of any type of system are typically recommended. Indeed, studies which rely on measurements are much more accurate and precise because of their adherence to real production environments. The influences of the architectural and of the software components on the performance of the workload are automatically considered.

Various problems, which make experimental studies harder, arise. The availability of appropriate monitoring tools for collecting the information of interest is required. This is not always the case. For example, when applications executed in parallel or distributed systems are to be measured, particular events, such as, start/end of a transmission from a processor, or time spent by a server to process a remote file access, are to be detected. Specific tools, built for the purpose, must be used. From the measurements, parameters able to describe and reproduce the static characteristics of the load and its dynamic behavior are derived. Various types of techniques are then applied in order to derive a compact and manageable representation of such aspects.
Although workload characterization of different types of systems benefits of a common set of well–known analysis techniques, some sort of specialization is required either in the parameter identification and in the techniques applied for taking into account new architectural features as well as new operation modes.

Acknowledgment

The authors are indebted to Gianni Lo Conti for his great help in preparing the drawings.

References

[ACRS94] A.K. Agrawala, M. Calzarossa, P. Rossaro, and G. Serazzi. Workload Modelling in Multi-Window Environments. (in preparation), 1994.

[AKB87] W. Alexander, T.W. Keller, and E.E. Boughter. A Workload Characterization Pipeline for Models of Parallel Systems. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 186–194, 1987.

[AL90] T.E. Anderson and E.D. Lazowska. Quartz: A Tool for Tuning Parallel Program Performance. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 115–125, 1990.

[AMB76] A.K. Agrawala, J.M. Mohr, and R.M. Bryant. An Approach to the Workload Characterization Problem. *Computer*, 9(6):18–32, 1976.

[ANLC92] M.K. Acharya, R.E. Newman-Wolfe, H.A. Latchman, R. Chow, and B. Bhalla. Real-time Hierarchical Traffic Characterization of a Campus Area Network. In R. Pooley and J. Hillston, editors, *Proc. 6th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 113–127, 1992.

[Bard76] Y. Bard. A Characterization of VM/370 Workloads. In E. Gelenbe, editor, *Modelling and Performance Evaluation of Computer Systems*, pages 35–55. North-Holland, 1976.

[BB91] R.B. Bodnarchuk and R.B. Bunt. A Synthetic Workload Model for a Distributed System File Server. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 50–59, 1991.

[BCMS82] M.L. Bolzoni, M. Calzarossa, P. Mapelli, and G. Serazzi. A Package for the Implementation of Static Workload Models. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 58–67, 1982.

[BSST91] D.W. Bachmann, M.E. Segal, M.M. Srinivasan, and T.J. Teorey. Netmod: A Design Tool for Large-Scale Heterogeneous Computer Networks. *IEEE Journal on Selected Area on Communications*, 9(1):15–24, 1991.

[BT89] D.P. Beretsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation - Numerical Methods*. Prentice-Hall, 1989.

[CF86] M. Calzarossa and D. Ferrari. A Sensitivity Study of the Clustering Approach to Workload Modelling. *Performance Evaluation*, 6:25–33, 1986.

[CFZ93] B.M. Chapman, T. Fahringer, and H. Zima. Automatic Support for Data Distribution on Distributed Memory Multiprocessor Systems. In *Proc. Sixth Annual Workshop on Language and Compilers for Parallel Computing*, Lecture Notes on Computer Systems. Springer-Verlag, 1993.

[CHS88] M. Calzarossa, G. Haring, and G. Serazzi. Workload Modeling for Computer Networks. In U. Kastens and F.J. Ramming, editors, *Architekture und Betrieb von Rechensystemen*, pages 324–339. Springer-Verlag, 1988.

[CMM94] M. Calzarossa, L. Massari, and A. Merlo. Analysis and Characterization of the Workload of Parallel Systems. Technical Report CNR Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" n. 3/127, 1994.

[CMMP94] M. Calzarossa, L. Massari, A. Merlo, M. Pantano, and D. Tessera. MEDEA: A Tool for Workload Characterization of Parallel Systems. (in preparation), 1994.

[CMT90] M. Calzarossa, R. Marie, and K.S. Trivedi. System Performance with User Behavior Graphs. *Performance Evaluation*, 11:155–164, 1990.

[CS85] M. Calzarossa and G. Serazzi. A Characterization of the Variation in Time of Workload Arrival Patterns. *IEEE Trans. on Computers*, C-34(2):156–162, 1985.

[CS93] M. Calzarossa and G. Serazzi. Workload Characterization: a Survey. *Proc. of the IEEE*, 81(8):1136–1150, 1993.

[CS94] M. Calzarossa and G. Serazzi. Construction of Multiclass Workload Models. *Performance Evaluation*, 1994.

[CWDW92] B.M. Carlson, T.D. Wagner, L.W. Dowdy, and P.H. Worley. Speedup Properties of Phases in the Execution Profile of Distributed Parallel Programs. In R. Pooley and J. Hillston, editors, *Proc. 6th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 83–95, 1992.

[EZL89] D.L. Eager, J. Zahorjan, and E.D. Lazowska. Speedup versus Efficiency in Parallel Systems. *IEEE Trans. on Computers*, 38(3):408–423, 1989.

[Ferr72] D. Ferrari. Workload Characterization and Selection in Computer Performance Measurement. *Computer*, 5(4):18–24, 1972.

[Ferr84] D. Ferrari. On the Foundations of Artificial Workload Design. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 8–14, 1984.

[FJLO88] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker. *Solving Problems on Concurrent Processors*, volume I. Prentice-Hall, 1988.

[FMSC94] R. Friedrich, J. Martinka, T. Sienknecht, M. Chelliah, and A. Ranganathan. A Distributed Performance Measurement System for Large-Scale Heterogeneous Environments. Technical Report NSA-93-031, Hewlett-Packard Company, 1994.

[FSZ83] D. Ferrari, G. Serazzi, and A. Zeigner. *Measurement and Tuning of Computer Systems.* Prentice-Hall, 1983.

[FV90] D. Ferrari and D.C. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Area on Communications*, 8(3):368–379, 1990.

[GLP76] D.P. Gaver, S.S. Lavenberg, and T.G. Price Jr. Exploratory Analysis of Access Path Length Data for a Data Base Management System. *IBM Journal on Research and Development*, 20:449–464, 1976.

[Gons85] T.A. Gonsalves. Performance Characteristics of 2 Ethernets: an Experimental Study. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 78–86, 1985.

[GST91] D. Ghosal, G. Serazzi, and S.K. Tripathi. Processor Working Set and its Use in Scheduling Multiprocessor Systems. *IEEE Trans. on Software Engineering*, 17(5):443–453, 1991.

[Gun94] L. Gün. An Approximation Method for Capturing Complex Traffic Behavior in High Speed Networks. *Performance Evaluation*, 19:5–23, 1994.

[Guse90] R. Gusella. A Measurement Study of Diskless Workstation Traffic on an Ethernet. *IEEE Trans. on Communications*, 38(9):1557–1568, 1990.

[Hart75] J.A. Hartigan. *Clustering Algorithms*. J. Wiley, 1975.

[Hari83] G. Haring. On Stochastic Models of Interactive Workloads. In A.K. Agrawala and S.K. Tripathi, editors, *PERFORMANCE '83*, pages 133–152. North-Holland, 1983.

[Jain91] R. Jain. *The Art of Computer System Performance Analysis*. John Wiley & Sons, 1991.

[KDF89] J.P. Kearns and S. DeFazio. Diversity in Database Reference Behavior. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 11–19, 1989.

[Klaa92] O. Klaassen. Modeling Data Base Reference Behavior. In G. Balbo and G. Seraz zi, editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, pages 47–60. North-Holland, 1992.

[KS87] T. Kerola and H. Schwetman. Monit: A Performance Monitoring Tool for Parallel and Pseudo-Parallel Programs. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 163–172, 1987.

[LA87] S-Y.L. Lee and Aggarwal. J.K. A Mapping Strategy for Parallel Processing. *IEEE Trans. on Computers*, 36(4):433–442, 1987.

[LS76] P.A. Lewis and G.S. Shedler. Statistical Analysis of Non-stationary Series of Events in a Data Base System. *IBM Journal on Research and Development*, 20:465–482, 1976.

[LS92] P. Lenzi and G. Serazzi. PARMON: Parallel Monitor - Release 1.0. Technical Report CNR Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" n. 3/95, 1992.

[Mass94] L. Massari. Performance Analysis of Applications in Parallel Systems. In M.E. Woodward, S. Datta, and S. Szumko, editors, *Computer and Telecommunication Systems Performance Engineering*, pages 32–39. Pentech Press, 1994.

[MEB91] S. Majumdar, D.L. Eager, and R.B. Bunt. Characterisation of Programs for Scheduling in Multiprogrammed Parallel Systems. *Performance Evaluation*, 13(2):109–130, 1991.

[MW94] A.P. Merlo and P.H. Worley. Analyzing PICL Trace Data with MEDEA. In G. Haring and G. Kotsis, editors, *Proc. 7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer-Verlag, 1994.

[RCG72] C.V. Ramamoorthy, K.M. Chandy, and M.J. Gonzalez. Optimal Scheduling Strategies in a Multiprocessing System. *IEEE Trans. on Computers*, 21:145–154, 1972.

[RK85] S.V. Raghavan and R. Kalnakrishnan. On the Classification of Interactive Users Based on User Behaviour Indices. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 40–48, 1985.

[RS94] E. Rosti and G. Serazzi. Workload Characterization for Performance Engineering of Parallel Applications. In *Proc. 2nd Euromicro Workshop on Parallel and Distributed Processing*, pages 457–462, 1994.

[RSL91] E. Rosti, G. Serazzi, and P. Lenzi. Characterization of Numerical Algorithms for Distributed Memory Multiprocessors. Technical Report CNR Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" n. 3/72, 1991.

[RVH94] S.V. Raghavan, D. Vasukiammaiyar, and G. Haring. Generative Networkload Models for a Single Server Environment. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, 1994.

[SCCQ86] G. Serazzi, M. Calzarossa, V. Comincioli, and P. Quaroni. Exploratory Analysis of an Interactive Workload. In *Proc. CMG '86*, pages 744–749, 1986.

[Sera81] G. Serazzi. A Functional and Resource-Oriented Procedure for Workload Modeling. In F.J. Kylstra, editor, *PERFORMANCE '81*, pages 345–361. North-Holland, 1981.

[Sera86] G. Serazzi, editor. *Workload Characterization of Computer Systems and Computer Networks*. North-Holland, 1986.

[Sevc89] K.C. Sevcik. Characterizations of Parallelism in Applications and Their Use in Scheduling. In *Proc. ACM SIGMETRICS and PERFORMANCE '89*, pages 171–180, 1989.

[SFMF93] T.F. Sienknecht, R.J. Friedrich, J.J. Martinka, and P.M. Friedenbach. The Implications of Distributed Data in a Commercial Environment on the Design of Hierarchical Storage Management. In G. Iazeolla and S. Lavenberg, editors, *PERFORMANCE '93*, pages 1–20, 1993.

[SH80] J.F. Shoch and J.A. Hupp. Measured Performance of an Ethernet Local Network. *Comm. of the ACM*, 23(12):711–721, 1980.

[Worl92] P.H. Worley. A New PICL Trace File Format. Tecnical Report ORNL/TM–12125, Oak Ridge National Laboratory, 1992.

[YCHL92] P.S. Yu, M.S. Chen, H.U. Heiss, and S. Lee. On Workload Characterization of Relational Database Environments. *IEEE Trans. on Software Engineering*, 18(4):347–355, 1992.