Workload Characterization: A Survey[1]

Maria Calzarossa
Dipartimento di Informatica e Sistemistica
Università di Pavia
I-27100 Pavia, Italy
E-mail: mcc@gilda.unipv.it

Giuseppe Serazzi
Dipartimento di Elettronica e Informazione
Politecnico di Milano
I-20133 Milano, Italy
E-mail: serazzi@ipmel2.elet.polimi.it

ABSTRACT

The performance of a system is determined by its characteristics as well as by the composition of the load being processed. Hence, its quantitative description is a fundamental part of all performance evaluation studies. Several methodologies for the construction of workload models, which are functions of the objective of the study, of the architecture of the system to be analyzed, and of the techniques adopted, are presented. A survey of a few applications of these methodologies to various types of systems (i.e., batch, interactive, database, network-based, parallel, supercomputer) is given.

1 Introduction

The performance of a system is influenced by the characteristics of its hardware and software components as well as of the load it has to process. The analysis of the workload plays a key role in all the studies where the performance indices of a system are to be determined. Indeed, such indices are directly related to the workload and cannot be expressed by quantities independent of it.
Since the behavior of a real workload is very complex and difficult to reproduce, a model is required. Such a model has to capture the static and the dynamic behavior of the real load and it must be compact, repeatable and accurate.

The goal of this paper is to give a survey of the workload characterization by pointing out, through a few applications, the various steps required for the construction of a workload model and the sets of parameters to be considered in various types of studies.

The paper is organized as follows. The state of the art of the techniques and methodologies for the construction of workload models is presented in Section 2. A few application domains for these methodologies are then discussed. Section 3 is devoted to centralized systems, that is, batch, interactive and database systems. Workload characterization of network-based environments is described in Section 4. Section 5 analyzes multiprocessor systems, that is, parallel and supercomputer systems. Finally, a few conclusions are drawn in Section 6.

[1]This work was supported in part by the the Italian Research Council (C.N.R.) "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" under Grants N. 92.01615.PF69 and N. 92.01571.PF69 and by the Italian M.U.R.S.T. under the 40% and 60% Projects.

2 Workload modelling techniques

Although the execution of a workload is typically a deterministic phenomenon, it is often modeled as a nondeterministic one through the application of statistical techniques. This is because of the big amount of data to be measured and of the large numbers of variables interacting together and of phenomena to be considered.
The main steps for the construction of workload models (see e.g., [1], Chapter 2) can be summarized as follows:

1 - Formulation.
The characterization level and the workload basic component (i.e., the smallest level of detail such as job step, interactive command, update/inquiry transactions to a database), together with the parameters to be used for its description, are selected. A criterion for the evaluation of model representativeness is also determined.

2 - Collection of the parameters of the workload to be modeled while it is executed.

3 - Statistical analyses of the measured data. The following sub-steps are identified:

3.1 - Preliminary analysis.
The collected data may include distinct workload populations (e.g., batch, time sharing, transaction). The discovery of natural partitions promotes insights through focusing attention on smaller and more manageable portions of the original data set.

3.2 - Analysis of the parameter distributions.
This type of analysis may lead to the application of various types of transformation (e.g., logarithmic) of the original values of the parameters or to the identification and the removal of the outliers. For example, when the density function of a parameter is highly positively skewed, a logarithmic transformation is required. Furthermore, the outliers, that is, the components whose parameter values are very different from the typical ones, may distort the classification process. Hence, the original data set has to be trimmed by removing the elements having one or more parameters with values greater than a predetermined percentile.

3.3 - Sampling.
In order to process the measured data with reasonable amount of processing time and storage, the number of components to be considered has to be small. Hence, a sample, drawn from the measured data, is used in the following steps.

3.4 - Static analysis.
A robust classification is obtained when the values of the parameters are scaled such that they lie in a common interval [2].
The classification and partitioning of workload components is usually attained by clustering algorithms. Cluster analysis is a statistical technique very useful for discovering homogeneous groups in a given data set. Among the various clustering algorithms [3]

the most commonly applied to workload characterization is the non-hierarchical k-means method.

Other statistical techniques, such as grid techniques and principal components analysis, are also adopted to partition the given data set into classes of homogeneous components. A few representative elements will then be selected from the classes according to some extraction criteria and used in the construction of the workload models.

3.5 - Dynamic analysis.

When the time-varying characteristics of the workload (e.g., dynamic characteristics of the processing requests, evolution in time of the mixes of components in execution) are to be reproduced, the properties of several time series are considered. The application of various techniques, such as numerical fitting, statistical analysis of nonstationary series of events, and stochastic processes, provides a concise representation of the analyzed data sequences (e.g., the arrival and departure patterns of components to the system).

Probabilistic graphs represent another approach for the description of the dynamic behavior of the workload. User behavior graphs (ubgs) [4] are adopted for modelling interactive users in that they are able to capture the dynamic characteristics of the load by reproducing the sequences of commands submitted by the users. The nodes of such a graph, one for each user, correspond to different command types. The arcs, with the associated probabilities, represent the issuance of the next command upon completion of the current one. The duration of a user's stay in each node probabilistically equals the times the user spends typing in a command of that type, waiting for the system's response and thinking what command should be input next.

4 - Representativeness.

Several criteria can be adopted to assess the representativeness of a workload model. The performance oriented criterion, based on a characterization in terms of a vector $\vec{P}$, whose components are performance indices selected according to the objective of the study, is widely applied. Examples of these indices are response time, throughput, and resource utilizations. The accuracy of a workload model is then measured as a function of the difference between $\vec{P}$ and $\vec{P'}$, obtained executing the real workload and its model, respectively.

In [5], the Workload Analyzer Tool (WAT), a tool which allows the construction of a workload model by implementing a few of the steps previously described, is presented.

3 Workload characterization for centralized systems

Workload characterization for centralized systems is a stabilized topic that has been extensively studied and applied since early seventies. Such systems represent the first application domain of workload characterization and many papers on this topic have appeared in the literature

since then (e.g., [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]). In subsections 3.1 and 3.2, a survey of a few studies, where the various approaches and the techniques described in Section 2 have been applied, will be presented.

3.1 Batch and interactive systems

The most popular method adopted for workload characterization of batch and interactive systems is the clustering. One of its earliest applications can be found in [16] where such a technique is used for the construction of a workload model of a dual processors system used in a scientific environment. The significative clusters obtained, ranging from eight up to eleven, are characterized by a set of eight parameters such as, CPU time, number of I/O accesses per device type, number of job steps, number of files.

A resource and functional-oriented procedure for workload modelling is proposed in [17] and an application to data measured on a batch system is described. The program-steps, characterized by six parameters dealing with their resource consumptions, have been analyzed with three different techniques, that is, two clustering methods and the principal components analysis (PCA). Both the nonhierarchical (k-means) and the hierarchical (Minimal Spanning Tree) algorithms produce nine representative clusters where the programs are distributed according to the values of their six parameters.

The composition of each cluster is also investigated from a functional view point. More precisely, the programs have been characterized in terms of additional features related to the programming language used (e.g., FORTRAN, COBOL) and to the type of activity performed (e.g., compilation, execution). The distribution of the programs in the various clusters reflects the resource-oriented subdivision. For example, all the compilations of COBOL programs belong to a single cluster, which also contains a few I/O intensive executions.

The third type of analysis (PCA) allows the identification of six factors able to explain the correlations and the variance existing among the six resource-oriented parameters. The correlations among these features with respect to factor 1 and factor 2 are displayed in Figure 1. The relative positions of the parameters show their loadings on each factor. In the example, we derive that factor 1 is a CPU-I/O factor, that is, it has high loadings on both I/O disk and CPU times (parameters 1 and 4) which are also very highly correlated. The same considerations apply to parameters 2 (number of lines printed) and 5 (number of workfiles used). Furthermore, three of the six factors account for more than 75% of the total variance of the data set. Such an information together with the previous results can help in simplifying the workload model by reducing the number of features to be considered.

The distinct characteristics of the program types, that is, compilations and executions, can be also pointed out by means of the PCA. Figure 2 displays the projections of the programs-steps, which are compilations, on the subspace identified by factor 1 and factor 3. In particular, the programs are grouped according to the different memory requirements (factor 3) placed by the various compilers. The variations within each group are due to the different I/O and CPU times (factor 1) required which are, in turn, a function of the size of the source codes.

A model based only on the static characteristics of the workload does not provide a perfectly

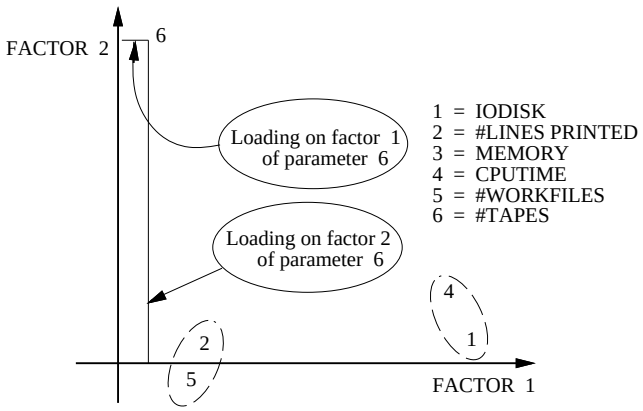

Figure 1: Loadings of the resource-oriented parameters on factor 1 and factor 2 [17].

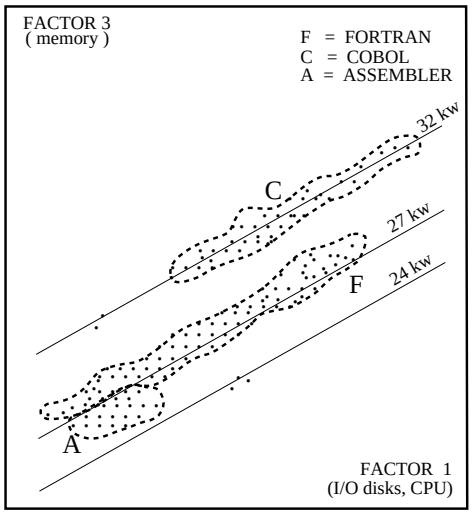

Figure 2: Projections of the compilations within the factor 1 - factor 3 subspace [17].

accurate representation of the load itself; it lacks in capturing the variation in time of the components and it does not reproduce their dynamic behavior.
Stochastic processes, numerical fitting and graph-based techniques are the most commonly used models for obtaining concise representations of the workload dynamic characteristics.

In [18], the hierarchical structure of interactive workloads is analyzed in terms of a stochastic model based on a top-down view of their basic components. According to such a view, a user session is a sequence of runs or jobs, which consist of sequences of tasks (e.g., editing, compilation). Each task can then be seen as a sequence of commands or statements. Finally, at the next lower level, there are the physical resources consumed by each statement.
The workload, i.e., the set of all the runs, is initially grouped by means of clustering techniques into seven subworkloads. The parameters characterizing each job are functions of the software resources (e.g., filing, FORTRAN compiler, link/load, execution). At the task level, a Markov

chain, whose states correspond to the software resources used by the run, is employed for the description of the behavior of the users. The transitions between states denote also the most probable sequence within the runs belonging to the various clusters. Figure 3 shows an example of task sequence measured in a single run. As can be seen, nine different states have been identified, namely, seven corresponding to the software resources employed and two fictitious states (BEGIN and END) used to represent the beginning and the termination of a run. Note that the run reaches the END state after seven steps from the beginning.

Several tests have been run in order to investigate the properties of such sequences. The

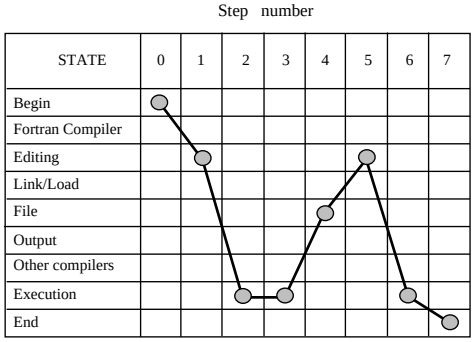

Figure 3: Task sequence from the BEGIN state to the END state measured in a single run [18].

analysis of the order of the Markov chain, which expresses the step dependencies within the sequences themselves, is equal to three in most of the clusters. Low order models have been found in clusters with small numbers of runs and of preferred states.
This type of characterization is particularly suitable for the construction of synthetic scripts to be used for the setup of benchmark experiments of interactive systems.

The alternative approach for the characterization of interactive workloads based on user behavior graphs (ubgs) has been employed in different respects. In [19], the ubgs are adopted as a means for obtaining a synthetic representation of a workload in terms of the command types, their resource demands and their sequence. A functional approach is initially applied to the data measured for all the 60 users of an interactive computer running under the UNIX operating system. The commands are grouped according to their types and the corresponding user behavior graph is constructed. Note that all the users are assumed to be statistically identical, that is, they obey the same ubg.
Since the number of different command types submitted to the system is quite large, namely, 113, an initial grouping according to the frequency of their occurrence is performed. A ubg consisting of 40 nodes, i.e., the first 39 heaviest commands plus a "catch-all" command combining together all the remaining ones, is then obtained. A further subdivision of these 40 commands, based on the mean and the coefficient of variation of their resource demands, is required for dealing with a more manageable model. Hence, a resource-oriented ubg consisting of eight nodes is constructed. The goal of the study is to analyze the causes that do not make the eight-nodes graph a perfectly accurate model of the forty-nodes ubg. The sensitivity of the accuracy of this clustering-based workload characterization to the dispersion of resource

demands of the clustered commands is also evaluated. The results obtained have shown that the clustering method for workload model design is reasonably accurate and the sensitivity is quite low.

Numerical fitting techniques are used in [20] for the analysis of the fluctuations in the arrival patterns of the workload components and the construction of a parametric model able to provide a concise representation of the analyzed phenomena.
The arrival times of the jobs at the system are collected during one month of operation. Some measurements have been discarded because of the occurrence of anomalous events (e.g., crash, holidays, maintenance) occurring on a few days. Hence, 14 one-day periods have been considered. A typical behavior of the variations of the arrival rate over a one-day period (from 8:30am to 6:00pm) is plotted in Figure 4. As can be seen, the peaks value of the morning, reached at about 11:00am, is followed by a decrease till 1:00pm, corresponding to the lunch break. The values of the afternoon rate, which are almost constant, are much lower than the morning ones. The dotted and the solid curves of Fig. 4 refer to the estimated rate function and the corresponding polynomial function provided by the fitting techniques, respectively. The

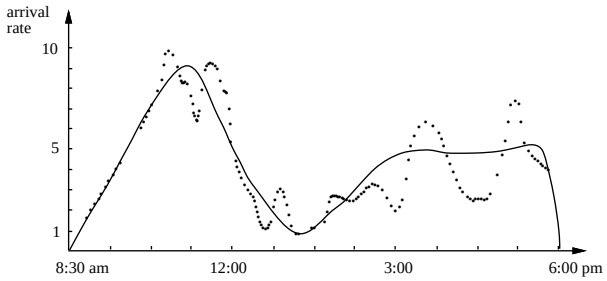

Figure 4: Estimated (dotted curve) and polynomial (solid curve) arrival rate functions [20].

application of these methods to the various days has shown that an eight-degree polynomial function is a suitable representation of all the analyzed arrival processes. Three representative patterns with similar behaviors have been identified in the data by means of the clustering applied to the coefficients of the various polynomial functions. The model obtained can be easily used for the forecast of the system load in the near future and for driving simulations when different dynamic control policies are to be investigated.

3.2 Database systems

A database system can be seen as a subset or as a part of a centralized system in that the users access interactively the database from their terminals by entering messages, called transactions. The workload description commonly adopted in studies dealing with these types of system is based on the analysis of traces, measured on real environments, or of reference strings, generated according to some stochastic processes. The traces are collections of accesses to database items or blocks produced by all the transactions processed during the observation

interval. Various techniques have been applied for their characterization, such as clustering and statistical analyses of non stationary series of events.

The identification of the locality of reference (i.e., to subsets of blocks of the database), of the sequentiality (i.e., sequences of increasing database block addresses) and of the transaction types provides preliminary information on both the performance and the behavior of the database and on the possible techniques to be adopted in studies dealing with buffer replacement and concurrency control policies (see e.g., [21], [22]). In particular, when the transactions are classified according to their functions (e.g., store, query), a finite state machine models can be applied.

Table 1 lists a few examples of parameters collected in a real database environment. The values of these parameters, provided in [22], belong to three of the six traces measured on a system under different configurations and typical load conditions. The mean transaction length refers to the number of different pages accessed per transaction. Only transactions with at least one page accessed are considered.

Parameters	Trace D	Trace E	Trace F
Observation interval	17'6"	14'8"	18'25"
Number of transactions	3695	580	1385
Mean transaction length	9.4	10.0	23.4
Pages accessed	11449	2307	10316
Pages written	1051	39	1838
Transaction types	47	21	34

Table 1: Summary of the parameters derived from three traces collected for a database system under different configurations [22].

For characterizing workloads with different page reference behavior a hierarchical-functional approach can be adopted. The approach is such that, starting from the user viewpoint of the database, the workload description is modified by each level in order to obtain the appropriate physical characterization.

In [22], a methodology, independent of the database and of the application, that is based on the following three abstraction levels, is proposed:

- application level, i.e., the user viewpoint of the database, where the data of the application, the functions and their relationships are defined;
- transaction level, where the transaction types associated to the functions previously identified are defined;
- physical resource level, where the list of read/write page accesses and cpu demands are defined.

In [23] and [24], various statistical approaches for the analysis of a large quantity of data collected on a computer running the IMS database management system are presented.

An exploratory study of the access path lengths, that is, the sequence of segments accessed when searching a database, measured for one-day period is performed in [23]. Graphical displays and simple numerical summaries are used to reveal patterns in the data. An appropriate stochastic process, which can be used as input for simulation models of IMS installations, is then proposed. In [24], the data of the times of the transaction initiations, taken over six whole days, are analyzed and represented by means of a nonhomogeneous Poisson process which can be seen as the superposition of inputs from various users. Figure 5 shows, for day 2, the plots of the mean number of transactions initiated as a function of the time of the day (dashed curve). Note that these values are computed as an average over 4800 adjacent unit time intervals. An

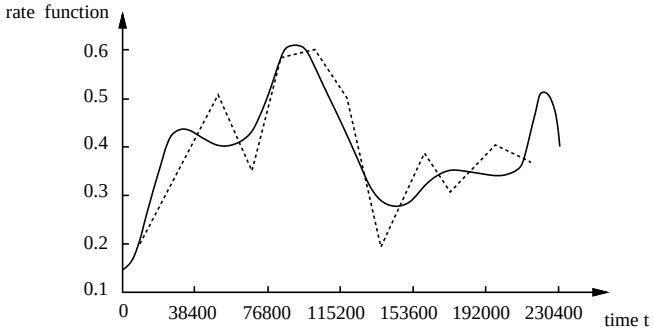

Figure 5: Mean number of transactions initiated in 4800 unit time intervals (dashed curve) and global estimate based on an exponential polynomial of degree 8 (solid curve) [24].

exponential polynomial function has been used for the estimation of the oscillatory nature of the Poisson process (see Fig. 5).

A workload classification aimed at determining the capacity of an IBM MVS system is presented in [25]. The study considers three workload types, namely, batch, timesharing and transaction processing. Examples of parameters used for developing a cluster description of the transaction workload are the total database calls, the number of input and output message segments and the total lock requests. Note that these parameters have to reflect the instantaneous conditions of the system (e.g., database organization, mix interaction, availability of common pool space) which heavily influence the behavior of the transactions themselves. Table 2 reports the values of the centroids of four out of the eight clusters obtained from the analysis of a sample consisting of 2000 transactions. These four clusters account for more than 90% of the sample.

4 Workload characterization for network-based systems

Before starting the overview of the methodologies used for workload characterization of "network-based" environments, we want to clarify that this term is used here to denote both basic computer networks considered in isolation or interconnected together (e.g., heterogeneous wide-area, metropolitan-area and local-area networks with different protocols) and distributed systems (see e.g., [26]).

Parameters	Cluster Number			
	1	2	3	4
Database calls	2.31	16.44	2.46	6.90
Lock Requests	6.69	16.00	8.84	9.71
Input Messages	1.90	1.74	3.94	2.00
Output Messages	0.84	1.29	2.01	25.19

Table 2: Centroids of four of the clusters obtained for a transaction processing workload [25].

The most commonly used performance measures for these environments are throughput, channel utilization and various forms of delay (e.g., transfer time, queueing delay) expressed as a function of the nature of the traffic and of the characteristics of the protocols and of the network. Sensitivity studies of the potential performance of these systems obtained varying the load and the network topology and connectivity are required for capacity planning, design and configuration purposes.

Hence, it is important to identify a set of parameters able to capture and reproduce the nature of the traffic flowing in the networks. This type of analysis is fundamental for multimedia networks, supporting digital audio and digital video components, characterized by a wide variety of traffic types. Furthermore, the complexity of the special equipments (e.g., routers, bridges, gateways, adapter cards) constituting the network-based systems and of their architectures and interconnections, requires the introduction of parameters able to take into account the influence and the mutual interactions between these components. As a consequence, the techniques and the parameters introduced in the previous sections for centralized systems are no longer sufficient in these environments.

The workload characteristics of a network environment can be specified in terms of a few basic parameters, such as, the packet generation rate for each node (e.g., workstation, terminal, server) together with the size and the routing of the packets. The analysis of the source and destination nodes allows us to distinguish between the intranet (i.e., both source and destination on the same network) and the internet (i.e., source and destination on different networks) traffic. Note that the information concerning the routing can be either measured or derived from user profiles that specify the source/destination of the applications.

In addition, there are protocol-dependent parameters, that is, the number of packets generated per message together with their distribution. In the case of store-and-forward networks, the CPU time spent for packet setup and teardown and the time spent for protocol conversion by the gateways has to be considered.

Once defined the set of parameters characterizing the workload of a network, the appropriate measurement tools must be found. In most of the studies presented in the literature (e.g., [27], [28], [29]) two different approaches are adopted: either special-purpose devices are used or ad-hoc test and monitoring tools are constructed.

In what follows, we give a survey of a few studies appeared in the literature which describe how to measure and analyze the parameters above introduced and which of them are to be used for

the input definition of both analytical and simulation models of different types of networks.

In [27] and [28], the authors focus on experimental measurements of Ethernet local-area networks. As we will see, although the two studies are quite similar, the results obtained by Shoch and Hupp are different from Gusella's because of the architectural differences of the two environments.

In [27], the first experimental study aimed at characterizing the traffic and the performance of a 2.94 Mbps Ethernet local network is presented. The network, connecting over 120 machines, was used for different kinds of applications ranging from file transfer and access to shared database systems, to specialized multimachine programs. The load of the network, analyzed in a 24-hours period, is described in terms of its traffic and the inter-packet arrival times. It has been noticed that the behavior of the load is strictly related to the time, that is, it is very light during the night and heavy in the daytime hours with a dip at lunchtime. The length of the packets exhibits a high correlation with the traffic type (e.g., file transfer, acknowledgement, terminal traffic) and shows a bimodal distribution with a mean value of 122 bytes. Note that for a 2.94 Mbps network the maximum packet length is equal to 554 bytes. The source/destination traffic matrix provides information about the most frequent patterns that can be used for identifying unbalanced situations and possible bottlenecks in the network. In the analyzed environment the traffic is concentrated to and from specialized servers.

In [28], a large network mainly consisting of diskless workstations with virtual-memory operating systems is studied. The measurements are collected by instrumenting the kernel of a dedicated UNIX machine. All the packets flowing on the 10 Mbps Ethernet local-area network are read and the protocol headers together with the timestamps of the packets are stored.

As can be seen from Fig. 6, the packet length, with a mean size equal to 578 bytes, is mainly distributed according to the protocols used by the applications (i.e., NFS, ND, TCP). In this case, an initial characterization of the packets based on their functional subdivision is appropriate. The packets belonging to each group can be then repartitioned according to specific traffic they carry. For example, two groups can be further identified within the NFS packets, consisting of the short ones transporting requests and responses for remote procedure calls and of the long ones produced by the data fragmentation performed by the IP during file read/write operations, respectively.

Another important result deals with the distribution of the packet interarrival times, i.e., the

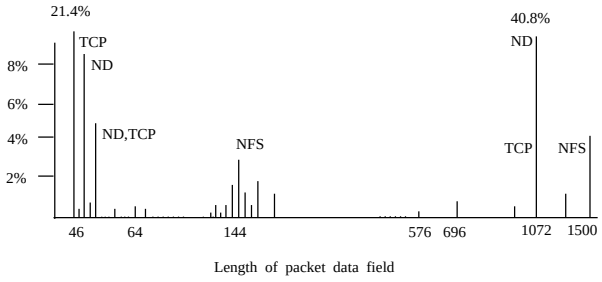

Figure 6: Distribution of the packet length as a function of the protocols [28].

times between two subsequent transmissions. There is some sort of correlation and dependence

on the protocols, the packet size and the traffic intensity. Hence, a Poisson process, typically employed in most of the simulation and analytical models, is not suitable in this case. Figure 7 shows the network utilization, over 1-minute intervals, caused by the traffic generated by a client workstation communicating with a file-server using the NFS protocol. The behavior of the network activity exactly reflects the user behavior which is usually characterized by a certain number of pauses.

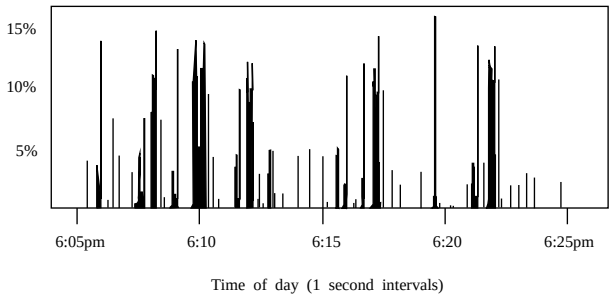

Figure 7: Network utilization as a function of the time of the day for the NFS protocol [28].

A few measurements aimed at determining the time to copy a 100 Mbyte file between two nodes of a local area network under varying load, i.e., normal, heavy and light (contention-free) loads, are presented in [29]. The experiments are intended as a validation means of the results obtained for a campus network model developed and solved with the NetMod design tool. For each workstation type, the rate at which it generates traffic and the average size of the packets are measured by means of a specialized monitor. The sending workstation transmits an average of 60 p/s with an average size of 1460 bytes. The receiving workstation generates 138 bytes acknowledgments with an average rate of 10 p/s. The study has shown that with a background traffic of 1034 p/s the transmission time is larger than the overhead due to the software.

In [30], a typical campus area network is analyzed. Its traffic is hierarchically characterized at various layers, namely, application, transport, and medium access layers. The behavior of intra-LAN, inter-LAN and LAN-to-WAN traffic has also been investigated.

A synthetic, externally-driven model able to generate load to an actual distributed system file server in a UNIX/NFS environment is presented in [31]. The model is based on the analysis of a few measurements, performed in a six week period, which capture the requests and the responses to and from the file server. Four key factors, namely, the frequency distribution of the requests, their interarrival time distribution, the file referencing behavior and the distribution sizes of read and write requests, are identified. A preliminary analysis of these parameters helps in discovering the typical behavior of the requests and in simplifying the model description. For example, an initial simplification is obtained by looking at the various request types. A small number of types, corresponding to the dominating ones, is considered and several others, which are negligible, are ignored. The analysis of the arrival process has shown that an exponential characterization of such a process is not appropriate because the average interarrival time and its variance are equal to 0.50 and 121.23 seconds, respectively. The model is then constructed according to the set of parameters identified in the workload characterization stage.

A survey of the performance issues in token-ring local-area networks with various protocols is presented in [32]. The main workload parameters used for the analysis of the delay-throughput characteristics of the basic IEEE 802.5 and FDDI protocols are the information-field lengths (exponentially distributed) and the frame generation rate (Poisson process).
In the case of the simulations of the flow control mechanism for token rings interconnected through bridges and a "backbone" ring, the following parameters are chosen:

- mean message length of 1 kbyte with a coefficient of variation of 1.5;
- mean frame length of 250 bytes;
- exponentially distributed intervals between generation of messages;
- time for a bridge to process one frame equal to 300 μs;
- size of the two buffer pools of each bridge of 4 kbytes.

Note that the mean message length has been chosen such that the overall frame length distribution (after segmentation and the insertion of control information), resembles the bimodal distribution measured on real systems. Figure 8 shows, as an example of the results of this study, the total throughput versus the total offered data rate for various window sizes W and a completely symmetrical traffic pattern. As can be seen, the throughput initially increases linearly and then it may decrease because of the heavy load of the network and the queues of frames at the bridges.

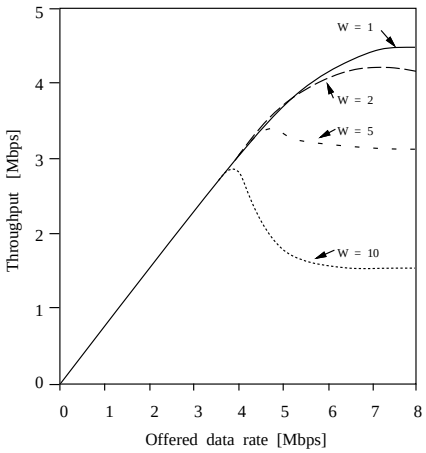

Figure 8: Throughput as a function of the offered data rate for various window sizes [32].

In [33], the performance effects of different distributions of node locations on a linear bus Ethernet are examined. The parameters used in these simulations are mainly related to the protocol and the network itself (e.g., carrier and collision detection time, interframe gap, length of the jam signal, number of buffers for each node). The end-to-end and the signal propagation delays are set such as resembling a 2000 m network. The traffic is characterized by the packet

length that is fixed throughout their study to 40 bytes. The interpacket arrival times are exponentially distributed and are chosen to yield moderately heavy loads.

In [34], a study of real-time services on a packet-switched store-and-forward wide-area network is presented. The communication is based on fixed-route connections (channels), that is, virtual circuits with performance guarantees. The multimedia traffic diversity is captured by considering four types of channels taxing the network's resources differently. The parameters characterizing the channels are:

- maximum service time t in the node for the channel's packets,
- minimum packet interarrival time on the channel x_{min},
- minimum value x_{ave} of average packet interarrival time over an interval of duration I.

Table 3 shows the values of these parameters for the four channel types. We can see that channel 1 imposes a very heavy load, while channel 4 is very light.

Channel type	t	x_{min}	x_{ave}
1	4	10	60
2	4	40	120
3	1	10	60
4	1	40	120

Table 3: Characterizing parameters, expressed in time units, of the four channels.

As a conclusion of this section, a global approach for the workload characterization of "network-based" environments (see [35]) is described. The methodology is based on a layered structure which corresponds to a logical subdivision of the hardware components into three groups, namely, user terminals, processing nodes and communication subsystem. According to this scheme, the load submitted by the users requires services from the local processing nodes and, eventually, goes to the network. Hence, at each of these three layers, i.e., user, processing and network, different basic workload components, which involve different physical resources, are identified.

Figure 9 shows the layered structure of such systems. As can be seen, the load of each processing node (layer 2) comes either from the users (layer 3) and from the network (layer 1). The dashed lines in the figure display two examples of possible paths followed by the local and remote requests, respectively.

For modelling the load at each layer, probabilistic graphs are chosen because of their ability to capture both the static and dynamic properties of the load itself. The user behavior graphs, adopted at layer 3, becomes, at the next lower layer, a system graph which is based on the overall sequence of requests, ordered according to their arrival time and generated by all the users or coming from the network. The requests which need processing from remote nodes cause various types of traffic on the communication subsystem. The load of layer 1 is then characterized by

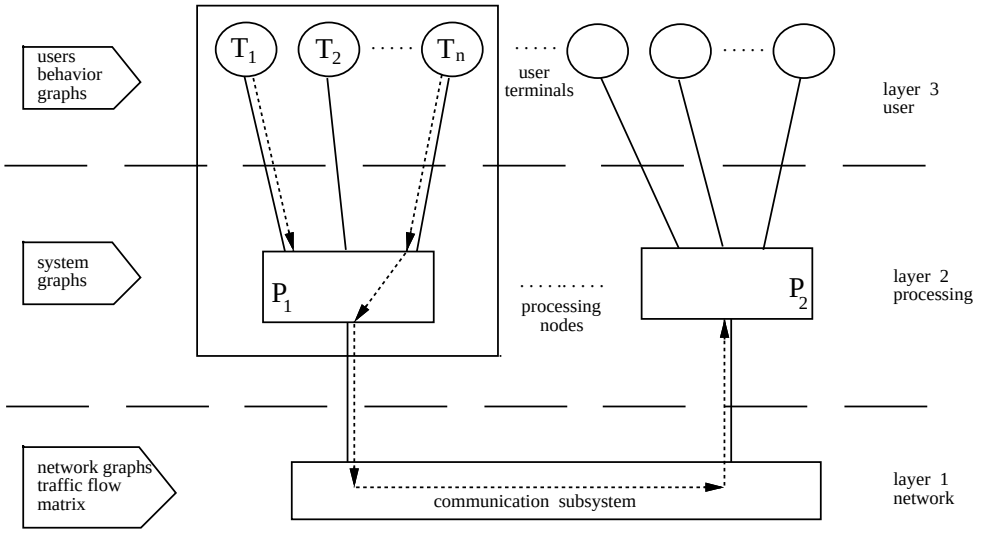

Figure 9: Layered structure for the workload characterization of "network-based" environments [35].

the network graph derived from the flow of messages to and from the communication subsystem and by the traffic flow matrix representing their routing.
The main steps for modelling the load of the various layers by taking into account their mutual interactions can be summarized as follows:

- measurements of the arrival sequence of the command types submitted by each user and construction of the corresponding behavior graphs (layer 3);
- measurements of hardware and software resource consumptions and of the arrival times of all the requests processed by each node and construction of the system graph (layer 2);
- measurements of arrival time, length, type and source/destination addresses for all the messages flowing in the network and construction of corresponding graph and traffic flow matrix (layer 1).

The parameters that can be measured/derived for each of the three layers and the corresponding modelling techniques are listed in Table 4.

Layer	Characterization level	Parameters/Techniques	
		measured	derived
3	user	arrival time command type	arrival rate interarrival time user behavior graph
2	processing node	arrival time request type hw/sw resource consumptions	arrival rate system graph
1	communication subsystem	arrival time message type message length source/destination nodes	generation rate interarrival time network graph traffic flow matrix

Table 4: Parameters and techniques adopted for "network-based" environments.

5 Multiprocessor Systems

Recent technological advances and increasing demands for computing power have led to the development of very fast computers such as parallel systems and supercomputers. The parallelism of the computations is the key factor for the performance of parallel systems. The performance improvements achieved with supercomputers are mainly due to their new architectural aspects. Some of the issues related to workload characterization for these two types of systems will be addressed in subsections 5.1 and 5.2.

5.1 Parallel Systems

In the sequential environments, the performance of a system is adequately described in terms of service demands, i.e., the amount of computation time required together with the processor instruction rate. The performance of parallel systems, characterized by a large number of cooperating processors, is influenced by a multiplicity of factors mainly related to the structure of the applications and to their ability to exploit the parallel features of the system.

The use of graphs for modelling parallel algorithms is a widespread practice. Each node represents a sequence of computations (tasks) and the arcs represent data dependencies. Directed graphs, i.e., *task graphs* [36] [37], are used for representing data-driven computations, while undirected graphs, i.e., *communication graphs* [38], are preferred for modelling control-driven computations.

In the sequel we will limit ourselves to the characterization of the algorithms represented by means of task graphs. Note that this type of approach can be easily extended to communication graphs.

A general classification of metrics for parallel environments can be done in terms of their dependence or independence from system architectures.

From the analysis of the graphs (see Fig. 10) a set of system independent performance metrics related to the static properties of the algorithms can be deduced. Static indices describe the complexity of the structure of an algorithm and capture its inherent parallel characteristics and its suitability to a particular system.
Dynamic indices are system dependent in that they describe the behavior of an algorithm when it is executed on a given system and also reflect how efficiently the parallelism is exploited. These metrics are appropriate for the evaluation of the match between algorithms and architectures.

A summary of a few static indices is given in Table 5. N is a measure of task granularity,

Static Metrics	Description
N	total number of nodes
in-degree	avg. number of direct predecessors of all the nodes
out-degree	avg. number of direct successors of all the nodes
depth	longest path between input and output nodes
maximum cut	max number of arcs taken over all possible cuts
problem size	size of the data to be considered

Table 5: System-independent metrics derived from the task graph of an algorithm.

while the values of *in-degree* and *out-degree* are related to the synchronization complexity. The larger the number of predecessors of a node the higher the probability that the corresponding task has to wait for synchronization. The value of the depth, i.e., of the longest path (in terms of number of nodes) that starts at the initial node and ends at the final node, is directly related to the execution time of the algorithm. The maximum cut, that is, the maximum number of arcs taken over all possible cuts from the initial node to the final node, deals with the maximum theoretical parallelism that can be achieved during the execution. The problem size is a measure of the number of elements in the data space that are to be considered.

To illustrate a workload characterization process, we discuss these parallel metrics with respect to a few algorithms. We consider two typical parallel algorithms, that is, the block decomposition matrix multiplication [39] and the LU decomposition [40] whose task graphs are reported in Figure 10. The values of the static indices derived from the analysis of a matrix size $n \times n = 24 \times 24$ with $p = 9$ processors are summarized in Table 6.

The block multiplication algorithm consists of a sequence of intermixed computation and communication phases. The presence of these two phases will be emphasized by the dynamic indices (see Fig. 15).
Since the task graph of LU decomposition is asymmetric, with high probability the maximum number of processors (maximum cut), will be used only for a small fraction of the global execution time. The tasks are strongly interconnected, as reflected by the in-degree value.

Since static indices lack in representing the parallelism exploited by the algorithm on a given system and its behavior as the execution progresses, dynamic metrics, that can be expressed with a single-value or with a curve (see Table 7), must be used.

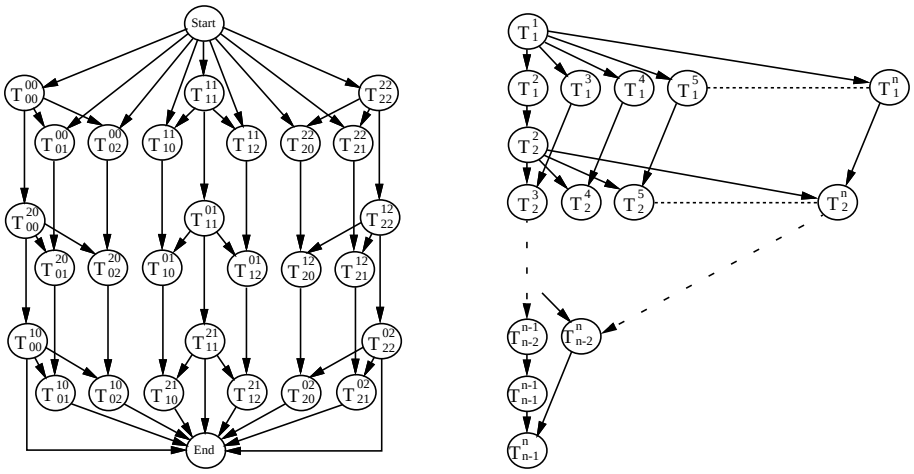

Figure 10: Task graphs for the block matrix multiplication (**a**) and the LU decomposition (**b**) algorithms.

algorithm	problem size	n. nodes	in-degree	out-degree	depth	max cut
Block	8	27	1.66	1.66	4	9
LU	24	299	1.85	1.85	46	23

Table 6: Static indices of the task graphs of the block multiplication and LU decomposition algorithms.

Single-value indices are determined by executing an algorithm with a given number of processors p. The average characteristics of the tasks (t_{comp}, t_{comm}) as well as those of the messages exchanged ($n_{messages}, l_{messages}$), and the global number of I/O operations n_{I/O_op} are particularly useful when the load of the algorithm must be reproduced (e.g., in simulation studies, in the evaluation of mapping and routing strategies).
A more complete characterization of the behavior of parallel algorithms consists of plotting the metrics as a function of the number of available processors (signatures) or as the execution progresses (profiles) [41].
The global execution time $T(p)$ of an algorithm, i.e., the *execution signature* [42], can be subdivided into two components:

$$T(p) = T_{comp}(p) + T_{comm}(p)$$

where the *computation signature* $T_{comp}(p)$ represents the fraction of the execution time in which one or more processors are computing and the *communication signature* $T_{comm}(p)$ is the time spent communicating (including the synchronization delays). Usually, $T_{comp}(p)$ is a monotonically decreasing function of p, while $T_{comm}(p)$ is an increasing function. Figure 11

Dynamic Metrics	Description
	Single-value
t_{comp}	avg. computation time of the tasks
t_{comm}	avg. communication time of the tasks
$n_{messages}$	avg. number of messages sent/received by the tasks
$l_{messages}$	avg. length of the messages
n_{I/O_op}	number of I/O operations
	Signatures
$T_{comp}(p)$	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 ($S(p) = T(1)/T(p)$)
$E(p)$	efficiency vs number of processors ($E(p) = S(p)/p$)
$\eta(p)$	efficacy vs number of processors ($\eta(p) = S(p)^2/p$)
	Profiles
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: System-dependent metrics derived from the execution of the algorithm with p processors.

shows these signatures for the block decomposition algorithm obtained on a transputer-based system with mesh topology and n equal to 48. As can be seen, with this particular problem size, the characterization of the algorithm changes from computation bound to communication bound when more than 9 processors are used. Indeed, with 16 processors the granularity of the tasks is very small and the computation time of each task becomes smaller than the time required to transmit the results to the other processors.

The *speedup* $S(p)$ [36] describes the gain (in time) of the parallel algorithm executed with p processors with respect to the serial one. The serial time may denote the time taken by the best possible serial algorithm or the time required to execute a given parallel algorithm on a single processor. The former definition is used when an absolute evaluation of the algorithm is required, the latter describes how well the algorithm has been parallelized. Since it is very difficult (if not impossible) to derive the optimal serial time even for simple algorithms, the second definition is commonly adopted.

The speedup curves for various problem sizes of the block decomposition algorithm are reported in Fig. 12. The advantage of increasing p is more evident when the problem size is large (the speedup is almost linear for n= 240).

In Figure 13 the speedup curves of the LU decomposition algorithm obtained on an Intel iPSC/2 for different problem sizes are reported. When the matrix size decreases, a reduction in speedup occurs due to the relative increase of communication time compared to computation time. As

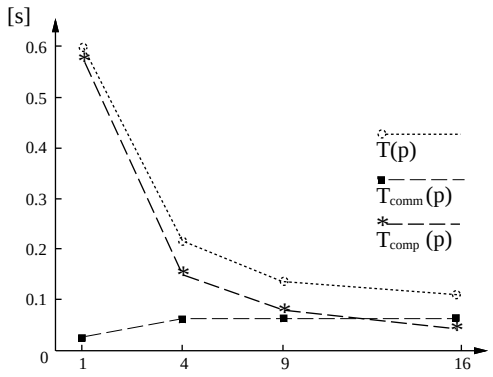

Figure 11: Signatures of the block decomposition algorithm on a mesh topology.

more processors are allocated, the potential reduction in computation time is less than the increase of communication time.
Based on the behavior of the speedup curves, a particular type of workload characterization can be derived. From the curves of Fig. 13, three typical representative behaviors can be considered:
- semi linear (matrix size = 512);
- concave (matrix size ranging from 128 to 256);
- flat (matrix size = 64).

The *efficiency* $E(p)$ is a measure of the fraction of time the processors are busy. The *efficacy* $\eta(p)$ describes how well the processors are used. It increases to a maximum and then decreases. The number of processors which attains the maximum $\eta(p)$ is the processor working set (pws) [45], that is, the number of processors for which the ratio of the benefit (increase in speedup) to the cost (decrease of efficiency) is maximum. The pws coincides with the number of processors corresponding to the knee of the execution time-efficiency profile [46].
Usually the maximum of $\eta(p)$ and $S(p)$ are obtained for different values of p. Figure 14 shows the $S(p)$ and $\eta(p)$ for the odd-even transposition sort of 1024 elements obtained on a transputer-based machine with linear array topology. The maximum speedup is obtained with 12 processors and the maximum efficacy corresponds to 8 processors. This means that the marginal benefit in allocating more than 8 processors (with the considered problem size) is less than the cost associated with the additional processors. The pws maximizes the performance index power, which is the ratio of the throughput to the response time [48]. The characterization of algorithms through their pws is useful in design studies (e.g., when processor allocation strategies are to be investigated) or in system tuning studies (e.g., for the identification of the optimal system operating point).

The parallelism profile gives the number of busy processors as a function of time and can be derived theoretically assuming an idealized machine having an unbounded number of processors. However, since the number of processors in the system determines the algorithm behavior, fixing the maximum number of available processors and measuring its value on a real system is more appropriate when the behavior of an algorithm on a given system must be described.

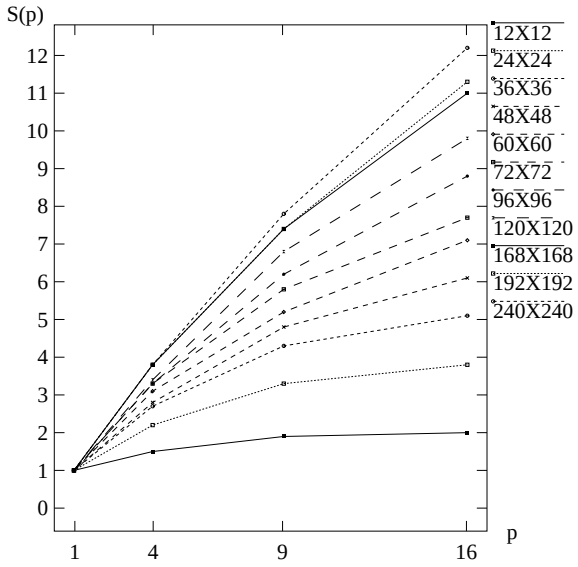

Figure 12: Speedup curves of the block decomposition algorithm on a mesh topology [43].

A parallelism profile can be further decomposed into a computation profile, which plots the number of processors that are computing at each instant of time, and a communication profile, which corresponds to the number of processors that are simultaneously communicating. The parallelism and computation profiles of the block decomposition algorithm executed with 16 processors are shown in Figure 15. The characteristics of this algorithm, that is, the synchronism of the computations and of the communications on all the processors, are emphasized by these profiles. Indeed, a computation phase (i.e., an interval of time in which all the processors are busy) alternates with a communication phase, in which all the processors are exchanging data. The two extreme phases, namely, the initial loading of the processors with the submatrices and the down-loading of the results are also evident from these profiles. The concept of

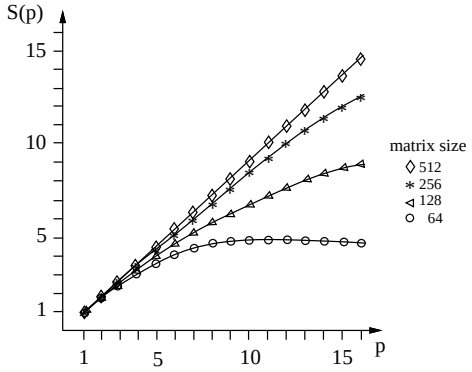

Figure 13: Speedup curves of the LU decomposition algorithm on a hypercube topology [44].

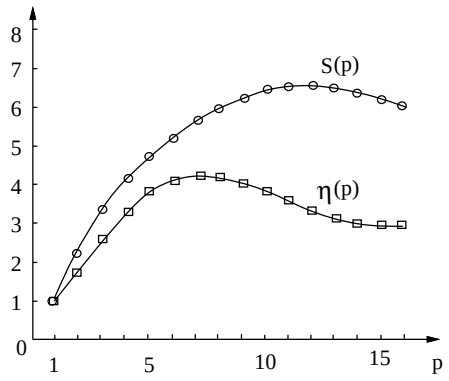

Figure 14: Speedup $S(p)$ and efficacy $\eta(p)$ for a odd-even transposition sort of 1024 elements [47].

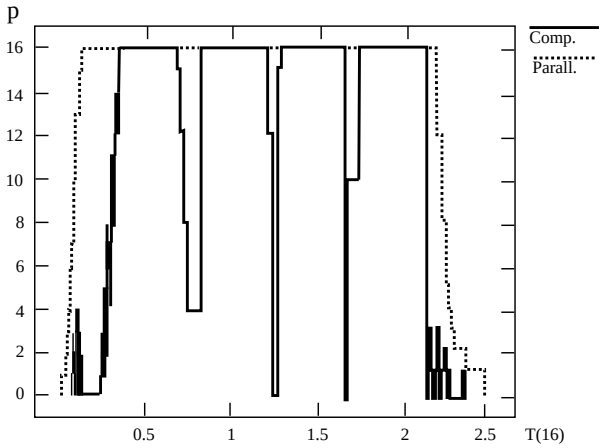

Figure 15: Parallelism and computation profiles of the block decomposition algorithm with 16 processors (problem size $n = 168$) [43].

the phases of a parallel algorithm is discussed in [49] [50].

The information in a profile can be more succinctly captured in another form of representation which is referred to as the shape [51]. This representation is a cumulative plot of the fraction of the execution time where a certain number of processors is busy. An example of shapes derived from a parallelism and computation profiles obtained with $p = 16$ is shown in Figure 16. From the profiles and from the shapes it is also possible to derive several single-value dynamic metrics (e.g., average numbers of busy, computing, and communicating processors). If the variability in the profiles is small, then these average values can be effectively used to characterize the program behavior. If the variability is high, the distribution of the values is required. From the shape of a parallelism profile the fraction of time in which the maximum and the minimum numbers of processor are utilized, is obtained. The fraction of the inherently sequential portion of the algorithm corresponds to the fraction of time in which only one

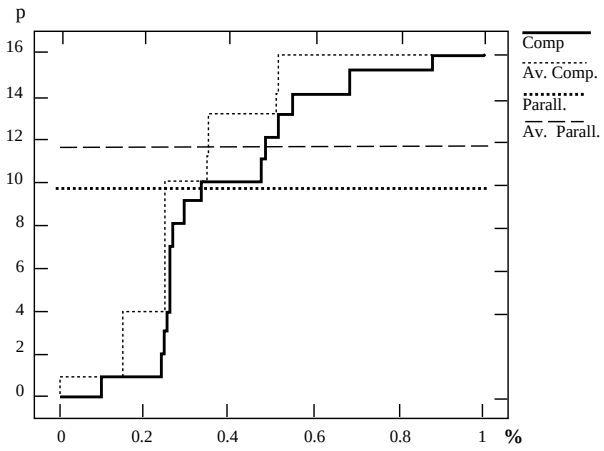

Figure 16: Example of shapes derived from a parallelism and a computation profiles.

processor is active. For example, in the shape of Fig. 16, all the processors are busy for only 15% of the execution time. For more than 25% of the time either one or no processors are computing.

More detailed information on the activities of each processor and on their timings with respect to each other and to the global execution time are provided by the horizontal bar diagrams (see Figure 17). The overlap of processor activities at each instant of the execution time are shown. A solid-line segment indicates that the corresponding processor is computing

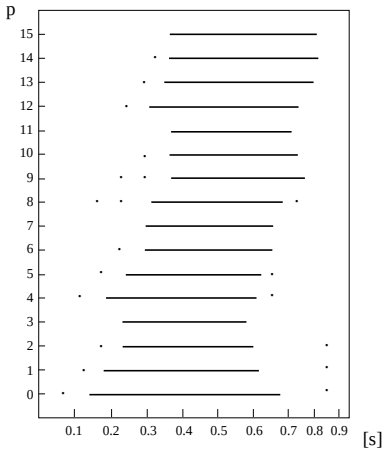

Figure 17: Example of horizontal-bar diagram.

in that period of time. An empty-line segment indicates a communication activity period.

5.2 Supercomputers

Supercomputers are high-performance multiprocessors machines with vector facilities, designed mostly for large scientific and engineering applications. Although the range of their possible performance improvements is quite wide, all the programs will not have equal benefits (see e.g., [52]). The hardware and software implementations of such systems strongly affect the performance experienced by the end-users. Hence, an accurate characterization of the workload of such systems is required.

Depending on the objectives of the study, a workload model can be defined in terms of "traditional" parameters related to the physical resource consumptions (e.g., global and per processor execution times, number of disk accesses) or to the application domains (e.g., fluidodynamics, image processing, structural analysis). Furthermore, the workload processed by supercomputers has to be described by means of parameters able to capture the behavior of the programs with respect to the architectural aspects of the systems themselves.
The analysis of the source code of the applications together with the information provided by tools, such as, profilers, pre-compilers, vectorizers, allows the identification of the program characteristics which are more directly related to the architectures. Examples of such parameters are the number of floating point operations, the average vector length, the percentages of scalar, vectorized, or parallel code, the vector stride, the communication patterns and the number of noncontiguous memory accesses (see e.g., [53], [54], [55]).
Starting from this detailed characterization, statistical analyses can be applied for identifying classes of representative components to be used in performance studies. For example, in the case of benchmarking, the most popular technique employed for assessing supercomputer performance, a careful and accurate definition of the programs to be used in the experiments has to be carried out. Hence, workload characterization must be included as a preliminary step for an effective benchmarking (see e.g., [56]).

A static and dynamic workload characterization study of a CRAY X-MP is presented in [57]. Over one million job trace records have been analyzed at the functional and resource levels. The jobs, described in terms of CPU time, I/O time and memory space-time product, have been subdivided into seven clusters. The changes of the workload intensities over an average weekday are also analyzed and six characteristic periods are identified, namely, three transient periods with increasing/decreasing intensities and three stable periods with fairly constant intensities, respectively.

Before concluding the section, we want to mention a few additional metrics, related to some properties of the programs and to the way they are structured, that can be measured on supercomputers (see e.g., [58] and [59]).
The variation of performance with vector length is captured by the two parameters $(r_\infty, n_{1/2})$, which denote the asymptotic performance (measured in MFLOPS) and the vector length necessary to achieve half of the asymptotic performance, respectively. The actual vector lengths of the programs are then compared with $n_{1/2}$ in order to establish what performance can be achieved.
Figure 18 shows, as a function of the vector length, the values of r_∞ and $n_{1/2}$ obtained on a one CPU of the CRAY X-MP for three different operations, namely, a dyadic and two triadic

operations. r_∞ and $n_{1/2}$ are represented by the inverse slope and the negative intercept with the n-axis of the best fit line, respectively. In particular, r_∞ is equal to 70 and 107 MFLOPS for the dyadic and the all vector triadic operations.

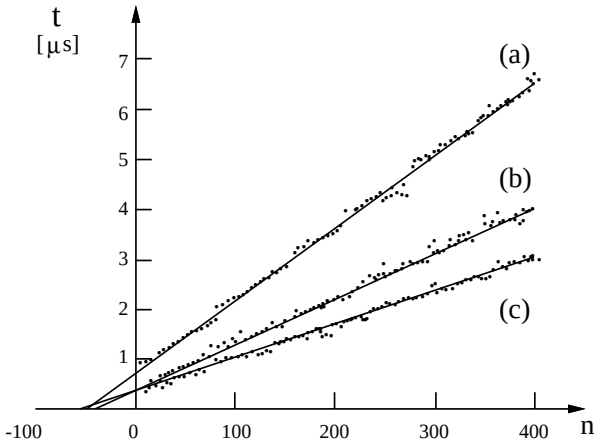

Figure 18: $(r_\infty, n_{1/2})$ for a dyadic operation (a), an all-vector triadic operation (b) and a triadic operation (c) [58].

6 Conclusions

Although workload characterization can be viewed as a settled discipline, it has remarkably progressed only recently.

The methodologies and the techniques applied for constructing workload models are strictly related to the objectives of the studies as well as to the type of system to be tested (e.g., centralized, distributed, parallel systems). As already pointed out, the problems encountered in the characterization of the workload of centralized systems are well-known and have been approached and solved since quite a long time. Similar conclusions cannot be drawn for more recent types of architectures (e.g., multiprocessor systems, client-server architectures, distributed database systems). The increasing number of hardware and software components interacting together makes the performance of such systems heavily dependent on the characteristics of the load. Hence, it is necessary to identify a set of parameters able to capture and reproduce the behavior and the evolution in time of the workload components processed by such systems. Furthermore, the workload characterization process must take into account the continuous evolution of system architectures. For example, in the case of widely-used multi-window enviroments, the workload has to be described in terms of the logical and physical parallelisms existing among the commands simultaneously "active" into the system. Indeed, such parallelisms will heavily influence both the behavior of the user and the patterns of the requests submitted to the system. Appropriate techniques have to be developed for modelling this type of workload.

In general, we can identify a few steps that can be seen as a common basis for any workload characterization study; such steps can be summarized as follows:

- choice of the set of parameters able to describe the behavior of the workload;
- choice of the suitable instrumentation, that is, use of existing performance/monitoring tools or construction of ad-hoc ones;
- experimental measurement collection;
- analysis of workload data;
- construction of static/dynamic workload models.

Then, a sort of "customization phase" of these steps is required for matching the objective of the study as well as the type of system under test.
Hence, we can conclude that the workload characterization still remains a promising research field in that it has to be updated continuously either for benefitting of new techniques and for making it more suitable to the requirements that will be raised by the new architectures.

Acknowledgment

The authors wish to thank Gianni Lo Conti for his drawing ability.

References

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

[2] H. Spat. *Clustering Algorithms for Data Reduction and Classification of Objects.* Ellis Horwood Publ., UK, 1980.

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

[4] D. Ferrari. On the Foundations of Artificial Workload Design. In *Proc. ACM SIGMETRICS Conference*, pages 8–14, 1984.

[5] M. Calzarossa, L. Massari, and G. Serazzi. The Design of the Workload Analyser Tool. ESPRIT-IMSE Document D4.2-1, University of Pavia, August 1990.

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

[7] K. Sreenivasan and A.J. Kleinman. On the Construction of a Representative Synthetic Workload. *Comm. ACM*, 17(3):127–133, 1974.

[8] 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.

[9] D. Ferrari. A Performance-oriented Procedure for Modeling Interactive Workloads. In D. Ferrari and M. Spadoni, editors, *Experimental Computer Performance Evaluation*, pages 57–78. North-Holland, 1981.

[10] G. Haring. On State-dependent Workload Characterization of Software Resources. In *Proc. ACM SIGMETRICS Conference*, pages 51–57, 1982.

[11] M. Calzarossa, M. Italiani, and G. Serazzi. A Workload Model Representative of Static and Dynamic Characteristics. *Acta Informatica*, 23:255–266, 1986.

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

[13] G.M. King. Workload Characterization. *CMG Transactions*, pages 143–153, 1987.

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

[15] M. Calzarossa and G. Serazzi. Construction of Multiclass Workload Models. *Performance Evaluation*, 1993. (to appear).

[16] A.K. Agrawala, J.M. Mohr, and R.M. Bryant. An Approach to the Workload Characterization Problem. *Computer*, pages 18–32, 1976.

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

[18] 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.

[19] M. Calzarossa and D. Ferrari. A Sensitivity Study of the Clustering Approach to Workload Modeling. *Performance Evaluation*, 6(1):25–33, 1986.

[20] 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.

[21] J.P. Kearns and S. DeFazio. Diversity in Database Reference Behavior. In *Proc. ACM SIGMETRICS and PERFORMANCE '89*, pages 11–19, 1989.

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

[23] 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.

[24] 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.

[25] H.P. Artis. Capacity Planning for MVS Computer Systems. In D. Ferrari, editor, *Performance Evaluation of Computer Installations*, pages 25–35. North-Holland, 1978.

[26] A.S. Tanenbaum. *Computer Networks - 2nd Edition*. Prentice-Hall, 1989.

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

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

[29] 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.

[30] 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, *Computer Performance Evaluation'92 - Modelling Techniques and Tools*, pages 113–127, Edinburgh, Scotland, 1992.

[31] R.B. Bodnarchuk and R.B. Bunt. A Synthetic Workload Model for a Distributed System File Server. In *Proc. ACM SIGMETRICS Conference*, pages 50–59, 1991.

[32] W. Bux. Token-Ring Local-Area Networks and Their Performance. *Proc. of the IEEE*, 77(2):238–256, 1989.

[33] T.A. Gonsalves and F.A. Tobagi. On Performance Effects of Station Locations and Access Protocol Parameters in Ethernet Networks. *IEEE Trans. on Communications*, 36(4):441–449, 1988.

[34] 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.

[35] 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.

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

[37] E. Gelenbe. *Multiprocessor Performance*. Wiley Series in Parallel Computing, 1989.

[38] H.S. Stone. Multiprocessor Scheduling with the Aid of Network Flow Algorithms. *IEEE Trans. on Software Engineering*, SE-3:85–93, 1977.

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

[40] R.E. Lord, J.S. Kowalik, and S.P. Kumar. Solving Linear Algebraic Equations on an MIMD Computer. *J. of the ACM*, 30(1):103–117, 1983.

[41] K.H. Park, L.W. Dowdy, and T.D. Wagner. Parallel Workload Characterizations: Comparisons and Mappings. Technical report, Dept. of Computer Science, Vanderbilt University, 1992.

[42] L.W. Dowdy and M.R. Leuze. On Modeling Partitioned Multiprocessor Systems. *Int. Journal of High Speed Computing*, 1993. (to appear).

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

[44] E. Rosti, E. Smirni, L.W. Dowdy, G. Serazzi, and B.M. Carlson. Robust Partitioning Policies of Multiprocessor Systems. *Performance Evaluation*, 1993. (to appear).

[45] 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.

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

[47] P. Lenzi, E. Rosti, and G. Serazzi. Experimental Evaluation of Internal Sorting Algorithms on Transputers. In M. Becker et al., editor, *Int. Conf. Transputers'92 Advanced Research and Industrial Applications*, pages 210–227. IOS Press, 1992.

[48] L. Kleinrock. Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications. *International Conference on Communications*, pages 43.1.1–43.1.10, 1979.

[49] M. Kumar. Measuring Parallelism in Computation Intensive Scientific/Engineering Applications. *IEEE Trans. on Computers*, C-37(9):1088–1098, 1988.

[50] B. 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, *Computer Performance Evaluation'92 - Modelling Techniques and Tools*, pages 83–95, Edinburgh, Scotland, 1992.

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

[52] J.L. Martin. Supercomputer Performance Evaluation: the Comparative Analysis of High-speed Architectures against their Applications. In J.L. Martin, editor, *Performance Evaluation of Supercomputers*, pages 3–20. North-Holland, 1988.

[53] I.Y. Bucher and J.L. Martin. Methodology for Characterizing a Scientific Workload. In *CPEUG'82*, pages 121–126, 1982.

[54] I.Y. Bucher and M.L. Simmons. A Close Look at Vector Performance of Register-to-Register Vector Computers and a New Model. In *Proc. ACM SIGMETRICS Conference*, pages 39–45, 1987.

[55] M. Calzarossa and G. Serazzi. Workload Characterization for Supercomputers. In J.L. Martin, editor, *Performance Evaluation of Supercomputers*, pages 283–315. North-Holland, 1988.

[56] K.E. Jordan. Performance Comparison of Large-Scale Scientific Computers: Scalar Mainframes, Mainframes with Integrated Vector Facilities, and Supercomputers. *IEEE Computer*, 20(3):10–23, 1987.

[57] J. Pasquale, B. Bittel, and D. Kraiman. A Static and Dynamic Workload Characterization Study of the San Diego Supercomputer Center Cray X-MP. In *Proc. ACM SIGMETRICS Conference*, pages 218–219, 1991.

[58] R.W. Hockney. (r_∞, $n_{1/2}$, $s_{1/2}$) Measurements on the 2-CPU CRAY X-MP. *Parallel Computing*, 2(1):1–14, 1985.

[59] R.W. Hockney. Problem Related Performance Parameters for Supercomputers. In J.L. Martin, editor, *Performance Evaluation of Supercomputers*, pages 215–235. North-Holland, 1988.