Chapter 2

Background and History

To date, there has been little work done in the area of workload characterization and modelling for distributed systems. The majority of the literature carefully examines all aspects of centralized systems, but gives rudimentary attention to distributed systems. It is important, however, to note that the two types of systems do not have to be treated independently with respect to workload characterization and designing workload models. Many of the concepts that are used for workload characterization and modelling of centralized systems are also applicable to distributed systems.

This chapter will outline the research that has been done in the area of workload characterization and modelling. In the first section, a general discussion of the steps that are commonly employed in designing workload models is presented. Following this are sections that discuss previous literature on centralized systems, various types of distributed systems, user classification studies, and cluster analysis, respectively. The final section recapitulates how these topics are relevant to this thesis.

2.1 General Concepts

To evaluate a system, whether it be for a performance study or for another type of study, it is extremely important to have a clear understanding of how the components in the system operate in conjunction with each other. The hardware, software, and other components that contribute to the load that the system must process should be characterized to provide an understanding of the system. This process is referred to as workload characterization; it is commonly defined as the evaluation of all inputs that are presented to a system.

How the system operates under the workload that has been identified in the initial workload characterization phase, and the system's performance as it processes this workload are of primary concern. The complexity and scale of real workloads make it difficult to evaluate the system without using some sort of a *system model*. Models are also practical in that they provide a convenient way to reproduce a workload, which is required when controlled experiments are to be performed on a variation of a validated workload model.

The processes involved in collecting data from a system, performing a workload characterization study, and constructing and validating a model for a system are well understood in centralized systems. Many of these techniques are also applicable to distributed systems. Several references ([Kob78], [Fer78], [FSZ83], [Fer84]) discuss the steps that are commonly followed when constructing a system model. We have already stated the six workload model design stages in Figure 1.1 of Chapter 1, but we will now examine these steps in greater depth. We also clarify some of the terms so that we will have a well-understood common ground on which to develop the discussions of the following chapters.

In the discussion of the workload model design steps that follows, some of the steps have been further divided into several substeps. The workload model design steps should generally be followed in the order listed below. The steps are rarely used in isolation, but rather all steps should be visited at least once. This is necessary if a confident system model is to result.

(1) Outline Model

Step 1a: Carefully plan the study

Not enough can be said about the importance of carefully planning the study in advance. Sufficient time should be taken to predetermine the goals that will be used to drive the study. The intended use of the model will influence the decisions that are made in the remaining steps of the model design. The cost-benefit tradeoffs should be kept in mind when outlining the goals of the study.

Step 1b: Identify basic components of workload/model

The description level of the components that comprise the workload form a hierarchy, as shown in Figure 2.1. At the highest granularity level are jobs or commands, and at the lowest granularity level are the physical resources consumed in the system. In general, the higher the level, the more system-independent the characterization, and the more likely that the characterization would have a loss of accuracy and/or result in a workload model that is difficult to calibrate. Depending on the goals of the study, the analyst must decide upon the type of basic component to be considered in the workload characterization and the parameters that will be used to characterize each component. Tradeoffs between system-dependence, granularity, and accuracy will have to be made.

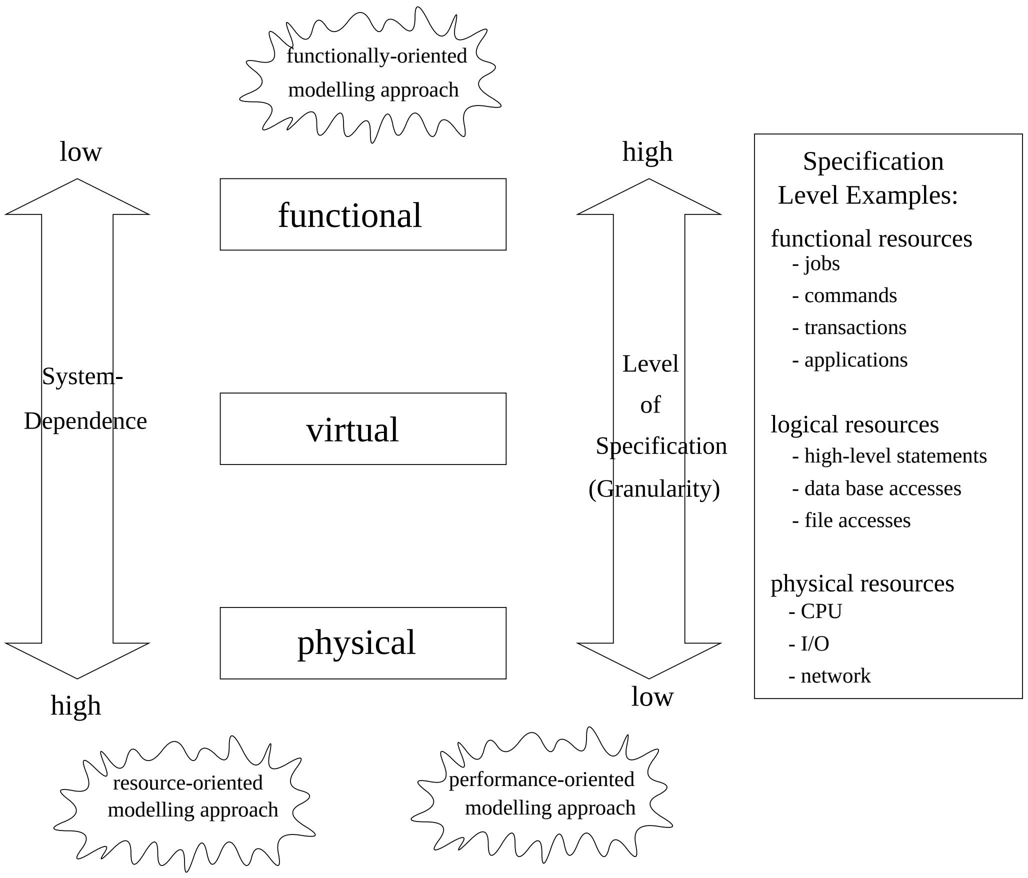

Figure 2.1: Workload Characterization Hierarchy

The three different modelling levels that are usually considered are: functional, virtual, and physical. Applications, such as commands or jobs that make up the workload, provide the characterization at the functional level. At this level, the workload characterization is independent of the system itself. At the virtual level the logical program resources are characterized, and at the physical level the resources (CPU, I/O, network, etc.) consumed are considered. The characterization is system-dependent at the physical level because the physical makeup of the system will affect the workload characterization. A model designed at this level could not be ported to a system with a different hardware or software configuration.

The ultimate goal of any workload model W' is to represent the actual workload W

on system S. Depending on the definition adopted to define this representativeness (accuracy), there are three different modelling approaches that may be taken, each with its own advantages and disadvantages.

Resource-oriented approach: This approach is the most common and the easiest to implement. The representativeness criterion defines that W' will be acceptable if it consumes the same physical resources at the same rate or in the same proportions as W.

Functionally-oriented approach: In this approach, W' will be representative of W if it performs the same *functions*. At first thought, this approach seems intuitively appealing; however, the task of building a functionally similar model is very complicated.

Performance-oriented approach: In this approach, W' is representative of W if the performance indices P_i' [1] produced by W' are the same as the performance indices P_i that are produced on system S by W. This approach seems quite logical, as most models are used in performance evaluation studies. This is the blackbox model that is shown in Figure 2.3 later in this section.

(2) Data collection

In this step, the data are collected from the system. This collection should include all parameters that might possibly be of interest. It is better to collect too much data than not enough, because the hardware or software of a system may change, making it impossible to recollect the missing data. It is also possible that the particular collection period is significant, and that recollecting the missing data at another time would not suffice. The preliminary analysis will be used to narrow down the actual collection duration that will be used in the synthesis of the system model.

(3) Workload Characterization

An exact representation of each and every detail in a complex system would result in a model that is very difficult to calibrate and validate. Such a model would be extremely

[1] P_i is the i^{th} performance index in a set of n performance indices chosen to evaluate the representativeness of the model. These performance indices may be such quantities as utilization, response time, throughput, and queue length.

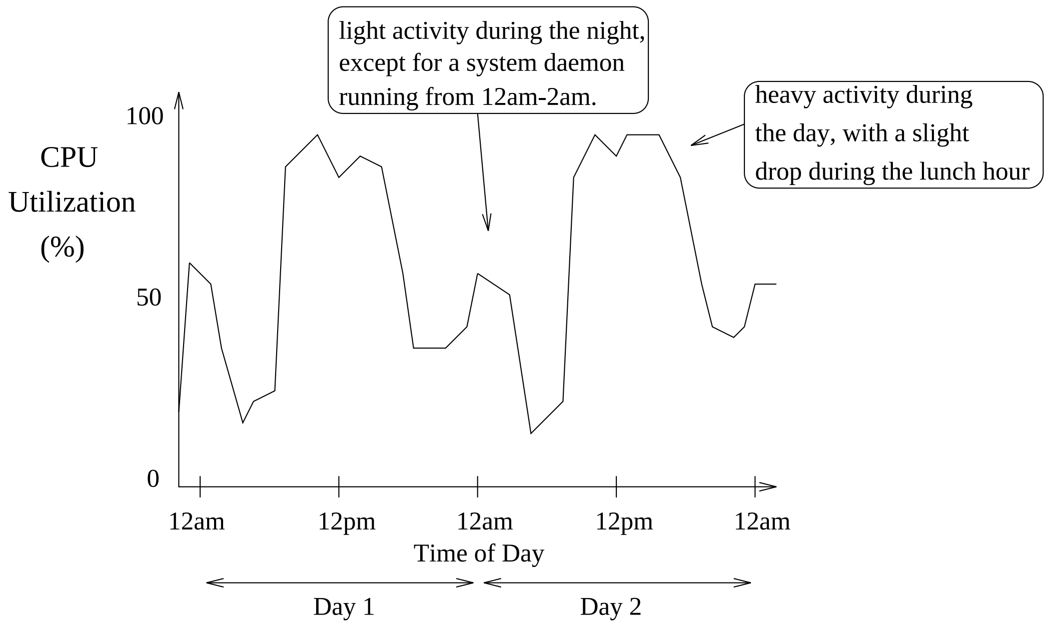

Figure 2.2: Typical CPU Utilization Fluctuation over a period of Two Days

large and cumbersome to manage. The most crucial elements (parameters) of the workload must be identified and a model constructed that includes these key elements. The initial workload characterization should identify potential bottlenecks in the system. It should also be used to identify natural partitions in the data. Focusing on these smaller components of the workload will make the system model much more manageable.

The preliminary analysis should be used to identify and refine the interval that will be used for the detailed study. Often workloads will exhibit bursty periods of intense CPU, I/O, or network activity. These fluctuations may correspond to normal variations in user or system activity, or they may correspond to increased system load at the onset of a course assignment or software release deadline. Figure 2.2 shows normal fluctuations that might be expected in the system's CPU utilization over a period of two days. Depending on the intended purpose of the study, the analyst must decide if these fluctuations are significant and if they should be included in the study. Probabilistic models and time series analysis may be used to aid in the selection of the duration period of the analysis.

(4) Model Design

Step 4a: Analysis of parameters

Once the set of parameters to be used to characterize the basic components in the workload has been determined in the preliminary analysis, a method to derive the values of parameters for the representative components must be identified. There are usually two types of parameters, which are considered separately: *single-value* and *sequence-type* parameters. Single-value parameters represent global values such as CPU time, the number of I/O operations on disk, or the memory space demand. Sequence-type parameters represent the dynamics of the execution of the components, such as sequences of I/O or CPU bursts.

Usually distributions of each parameter are studied to gain more insight into the data. The probability density function (pdf) and the cumulative distribution function (cdf) of each parameter are commonly examined to determine the skewness or modality of the data.[2] If the data are extremely skewed, a decision may be made to remove the outliers, as they may contort the classification methods. Depending on the goals of the study, the extracted outliers may have to be incorporated back into the model using an alternative means.

Some of the extraction techniques that will be outlined in Step 4b require *normalization*: the parameters must be within a uniform range. In this case, a *scaling* technique is applied to the parameters to convert them to homogeneous units. If there are n parameters and m components, one commonly used normalization formula is

$$x'_{ij} = \frac{x_{ij} - \overline{x}_j}{\sigma_j}, (j = 1, \ldots, n; i = 1, \ldots, m), \tag{2.1}$$

where x'_{ij} is the normalized value of the j^{th} parameter for the i^{th} component, and $\overline{x}_j$ and σ_j are the mean and standard deviation of the j^{th} parameter. This transformation results in a mean of zero and a standard deviation of one for each parameter in the new data set.

Step 4b: Extraction of representative values

If there are n parameters that are chosen to characterize the components of the workload in the preliminary analysis, then an n-tuple vector is used to represent each component of the workload. Since the amount of data is often immense, a statistical technique that identifies groups of components with similar characteristics should be applied. In studies that have two or fewer parameters, regression, correlation analysis, or goodness-of-fit distribution tests are sufficient. Due to the complex nature of most systems, however, generally more than two parameters are needed to characterize the workload. In this case, a multidimensional method such as cluster analysis must be employed. Ferrari *et al.* [FSZ83] list the following

[2]The probability density function is the function $f(x)$ that is the fraction of all processes observed that require x units of the resource. The cumulative distribution function is the function $F(x)$ that is the fraction of all processes observed that require $\leq x$ units of the resource.

different statistical methods that can be used with multidimensional data:

1. Sampling of single parameter distributions
2. Sampling of workload components
3. Clustering algorithms
4. Reduction of joint probability distribution of parameters
5. Markov models
6. Principal component analysis

Method 1 samples the probability distributions of each parameter to derive the parameters (features) for the model. Method 2 uses direct sampling from the actual workload. Method 3, cluster analysis, groups similar components of the workload into classes, and then representative components are extracted from each class and assigned to the single-value parameters in the model. This method is discussed in more detail in Section 2.5. Method 4 is based on matching the joint probability of the actual workload and the model workload. Method 5 is most frequently used when the dynamic properties of the model are to be recapitulated. Method 6 can be used to reduce the number of parameters in the model. It examines the correlations among the parameters and attempts to discover a reduced feature set such that the projection of the data onto this lower dimensional space accounts for a significant proportion of the total variance of the original data set.

Once statistical methods have been used to partition the data, extraction from the workload's representative clusters is necessary. If a large number of components is extracted, this is likely to increase the model's representativeness, while at the same time decreasing its compactness. Depending on the statistical method and criterion used, the extraction may take different forms. For cluster analysis, the centre of mass of a cluster (see Equation 2.4) is often used to represent it.

Step 4c: Assignment to components

There are two general classes of workload models: *natural* models and *synthetic* models. Natural models may be a live load or trace from a system, whereas synthetic models do not contain any direct samples from a real workload. Natural models are usually quite easy to develop compared to synthetic models; however, they are very inflexible and system-dependent. Usually they represent only a short interval of the real workload, due to the overwhelming amount of data that would be needed to represent longer periods of time. Synthetic models are most commonly in the form of parametric programs that attempt to reproduce the resource usage observed in the real workload. Synthetic models are much more flexible and compact than natural models.

In this step, the representative values must be transformed into executable components. Depending on the class of workload model to be used, this step may be performed differently. If a natural model is to be used, the components of the actual model whose parameter values are closest to the values of the representative components must be identified. If a synthetic model is to be used, the parameter values of the representative components must be assigned to the control parameters, and the model must be calibrated by modifying the appropriate calibration parameter (Step 5a will explain the calibration process).

Step 4d: Reconstruction of mixes

The mix of the significant components in the model is reconstructed in this step. For the model to be representative of the actual system, situations that occur in the original system should also happen in the workload model. Since periods of intense activity and specific mixes of components may have an effect on the critical resources in the system, they should be captured in the model.

(5) Model Validation

For a workload model to be useful, it must be representative of the workload that it attempts to model. The definition of representativeness will depend upon the modelling approach chosen in Step 1b. Two different types of workload model validations, *blackbox* and *whitebox* validation, are outlined in [Bod90]. These methods may be used in isolation, but for best results, they should be used to supplement each other.

The more traditional blackbox model validation is shown in Figure 2.3. This validation is used when the performance indices determine the representativeness of the model. The actual workload W is measured from the actual system S and is used to derive the workload and device parameters that comprise the model workload W'. The model workload W' is used as input to the workload system model (simulator or other performance calculator) and produces the performance indices P_i'. The actual system performance indices P_i are measured from the actual system S and are compared to the model performance indices P_i'.

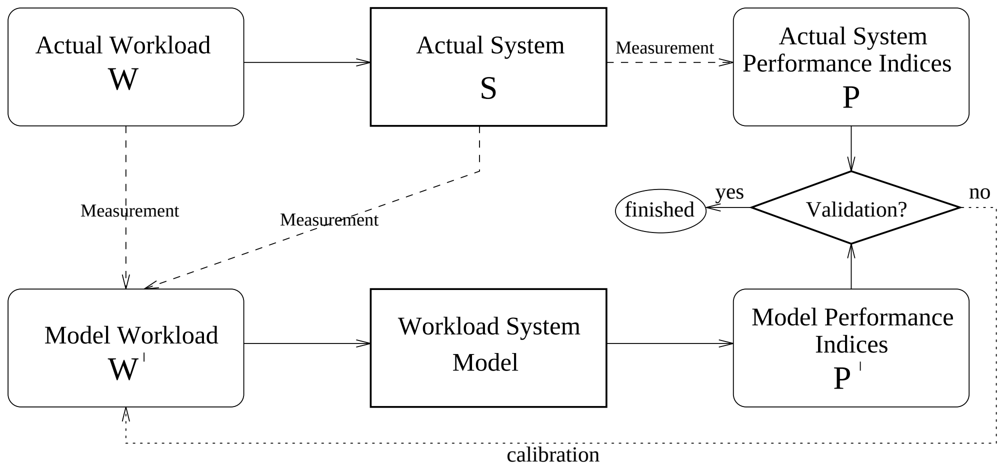

Figure 2.3: Blackbox Model Validation Scheme

In general, the model is said to validate if

$$\sum_{i=1}^{n} \frac{|P_i - P_i'|}{P_i} \leq e, \tag{2.2}$$

where e defines the degree of accuracy required. If the desired accuracy is not obtained, then model calibration may be performed by adjusting the parameters in the workload model W' and rerunning the workload model to produce new model performance indices P_i'.

The whitebox model validation that is shown in Figure 2.4 is a direct comparison between the actual workload W and the model workload W'. This comparison is not always very straightforward, especially when a large number of parameters comprise the model workload. The significant workload components WC_i identified by the preliminary workload characterization in Step 1d should at least be validated. This type of validation scheme is particularly applicable to a functionally-oriented workload modelling approach.

(6) Model Usage

The evaluation of a computer system may be conducted for a variety of purposes, including capacity-planning, performance analysis, or sizing and selection of a new computer system. The approaches that are taken to perform this evaluation are commonly classified into the following three types: (1) analytic models, (2) simulation models, and (3) empirical models.

An *analytic model*, which is an analytical representation of a mathematical model, is the most desirable method, as it can evaluate the system performance with minimal efforts and costs over a wide range of system parameters and configurations.

As most complex, real-world systems with stochastic elements cannot be accurately described by an analytic model, a *simulation* is often the only type of investigation possible.

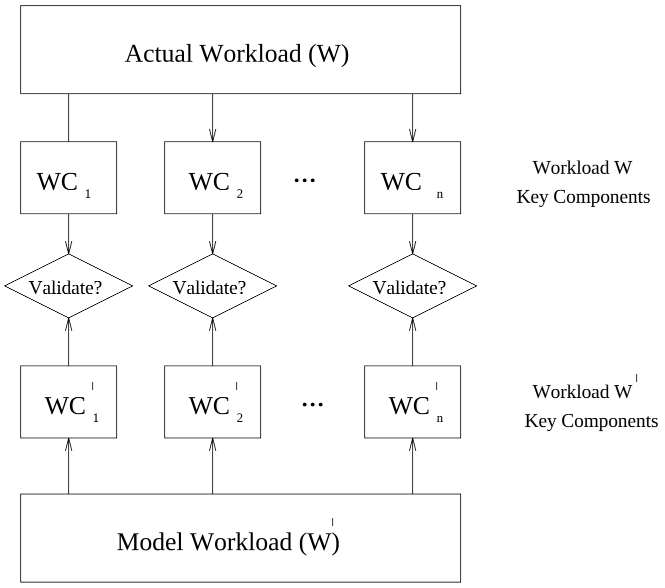

Figure 2.4: Whitebox Model Validation Scheme

A simulation model represents the behaviour of a system as it evolves over time, under stimuli which represent the system environment. Simulations are desirable because they allow us to estimate the performance of a system under the actual system conditions or under a projected set of conditions.

An *empirical model* is a statistical characterization of system performance based on data measured from the system itself. As measured data are also needed for the other approaches, this technique may confirm confidence in the model.

Ferrari [Fer78] suggests that attacking a study with various approaches, techniques, and tools may give the evaluator the most understanding of the system.

2.2 Interactive Systems

The earliest studies in the area of workload characterization and modelling were for batch systems ([SK74], [HPLG79], [HDG71], [Ser81b], [Fer72], [AMB76]). In these batch systems, the workload was submitted as a number of individual jobs, which were processed one at a time, without any user interaction. These early studies used classification techniques such as cluster analysis to identify the homogeneous components of the workload. A model for the system was then produced using selective representative components that were identified by the workload characterization. In general, the implementation of executable models for batch systems requires much less information than that needed for interactive systems.

2.2.1 Static Studies

There was gradually a progression towards interactive systems with a more user-friendly user interface. As time-sharing systems with interactive users began to replace batch systems, a myriad of studies for interactive systems began to emerge. Most studies were for *static workload models* ([SK74], [AM78], [Fer81], [HS82], [BCMS82]), in which the time-varying characteristics of the system were not taken into account. For these studies it was assumed that the system was operating at a steady-state equilibrium during the data collection interval. Many of the techniques that are used in workload characterization and construction of static models for centralized systems can also be applied to distributed systems.

In [AMB76] an initial observation of the data was made by plotting histograms for individual features in the data set. Highly skewed histograms were replotted using a logarithmic scale, to provide more information about the distributions. They also displayed a two-dimensional scatter plot, but noted that due to the multimodal nature of the data (the data contained 17 features), higher dimensional techniques such as clustering were required. They clustered on the job features and used the means of the features in each cluster and Kiviat graphs to access the appropriateness of the clusters chosen. This examination revealed that the clusters had similar characteristics, such as the types of users and tasks placed in each cluster, even though only the resource requirements of jobs were used in the cluster analysis.

[HS82] designed a benchmark for a research VAX/VMS environment. The workload characterization problem was approached in a much different manner than in [AMB76]. Their goal was to develop a benchmark by identifying a set of programs that were a representative sample of system activity. They studied the types of programming done by the eight most active users in the system, interviewed users in the system, and monitored the system to determine a representative benchmark script.

[Ser81b] proposed a functional and resource-oriented procedure for workload modelling. The basic workload component chosen for the analysis was the program-step that consisted of six different features: disk I/O, lines printed, CPU, memory, work files, and tapes. They applied two different cluster analysis methods and principal component analysis to examine the statistical properties of program-steps. The two clustering methods produced nine different representative clusters each. Examination of the program-steps that were placed in each cluster showed functional similarities within the clusters indicating favourable partitioning by the clustering methods. Principal component analysis, which is commonly used to reduce the size of a feature set, was also applied to the six features. The principal component analysis of the six features was in agreement with the functional characterization determined by the clustering methods.

After workload characterization has been used to identify the key components, the model must then be constructed. A general method for the construction of a representative synthetic workload is described in [SK74]. In this study, a synthetic program is used to generate a synthetic workload consisting of a subset of the jobs in the actual workload. The joint probability density function of the actual workload was used to generate the synthetic workload. The CPU time and the number of I/O activities were the two features that were chosen to quantitatively represent the workload.

Cluster Analysis is one of the most commonly used techniques for static workload model construction. [Chu79] used cluster analysis to identify a number of workload classes in a centralized IBM computer system at the University of Toronto. The classes derived from the cluster analysis and the installation-defined classes were compared by using each as input to a multiclass queueing network model of a computer system. Utilization and throughput of the queueing network model were examined to make this comparison. The clustering classes were better than the installation-defined classes in terms of their representation of the resource usage patterns and their validation of the queueing network model.

As cluster analysis is the component extraction technique that we will use in our study, a further discussion of cluster analysis and several studies that have benefitted from its use will be presented in Section 2.5.

2.2.2 Dynamic Studies

A model that is only based on the static characteristics of the workload will not be able to capture the variation in time and the dynamic behaviour of the components. In systems that have a lot of workload fluctuation, a dynamic model may be required to provide an accurate representation of the workload.

There have been many studies that examine dynamic model components ([CS94], [Hug84], [RK85], [CS85], [CIS86], [Har82], [Har83], [CS86], [CMT90], [Fer84], [Ser81a]). These studies focus on such dynamic aspects as the job mix, or use Markov chains or time series in the design of dynamic system workload models. The most commonly used models for obtaining concise representations of the workload's dynamic characteristics are stochastic processes, numerical fitting, and graph based techniques.

Although the model designed in this thesis is not dynamic, we do briefly examine some of the dynamic components of the workload of our study system (see Section A7.1).

2.3 Distributed Systems

Designing models for distributed systems is much more difficult than for centralized systems. This is mainly due to the large number of input workload parameters that are typical of distributed systems. The timing between the many components that are found in distributed systems also adds to this complexity.

In [Fer86], Ferrari outlined a number of problems that he felt may have a negative impact on the performance of distributed systems. It is the complexity of distributed systems that compounds these problems. Ferrari argued that the network itself is an obvious potential bottleneck and that the networking software is likely to cause transmission slowdown. Insufficient processing power or I/O bandwidth could also prove to be potential bottlenecks. The final potential bottleneck disclosed is related to suboptimal placement of files. Ferrari concluded this paper by stating the importance of workload characterization and measurement in distributed systems:

> "An absolutely essential requirement in all of the performance improvement efforts alluded to above is the measurement and characterization of the actual or expected distributed system workload. Our ignorance about the properties of the workloads of distributed installations for the various application contexts of interest is the single most important reason for our unsatisfactory understanding of distributed systems performance."

One topic that has received a lot of attention in distributed systems is the possibility of load sharing (or load balancing). Load sharing is an effective way to share computing resources in a distributed computing environment. The distribution of work among the various hosts in the system is evened out so as to improve the overall performance of the system. The primary objective of load sharing is to transfer jobs on heavily loaded machines to machines in the network that are not as loaded. This transfer may be performed when the job is initially placed or after it has accumulated some run time (migration). Workload studies by [LO86] and [Cab86] have proven fruitful in predetermining the potential of load sharing for distributed systems.

[LO86] analyzed the behaviour of UNIX processes, to predict the possible benefits of dynamic load balancing techniques. The study commenced with a detailed analysis of the resource usage of the UNIX processes. They found that processes were either CPU hogs, disk hogs, or ordinary (small in both CPU and disk usage) processes, and that a small percentage of the largest processes accounted for a large fraction of all CPU time used. They used their observed process characterization to drive a number of simulations for different dynamic assignment heuristics. Their three primary findings were:

1. For processes requiring large amounts of CPU, either initial placement or migration alone provides significant reduction in response ratios.

2. Initial placement and process migration provide the best improvement when used together, particularly as more processors are used.

3. Simple heuristics for initial placement and process migration can be used to drastically improve CPU-intensive processes, without hindering ordinary processes.

[Cab86] examined several different workloads and hypothesised about their influence on load balancing strategies. While this study was not specifically implemented on a distributed system, the idea of studying the workload to predict the potential of load sharing in this environment is relevant to our study. In this study, process data were collected from different installations of three different types of processors in multi-user systems. Similar to [LO86], the data were found to contain a large percentage of processes with very short lifetimes. They conclude that general purpose load balancing strategies should be based on process migration. They suggest implementing a mechanism that would detect and migrate long-lived processes, and they caution against implementing load balancing at the command interpreter level.

Recently there has been a proliferation of multi-window user workload environment studies ([ACRS94], [ASR94], [RJH95]). The environment in these studies is similar to the system we examine in this thesis. Behaviour that is typical of a user who has several windows (activities) running in parallel from a workstation is shown in Figure 2.5. They note that parallelism is introduced in a user's behaviour because a user runs jobs simultaneously from different windows. Although the jobs are running simultaneously, it is questionable whether

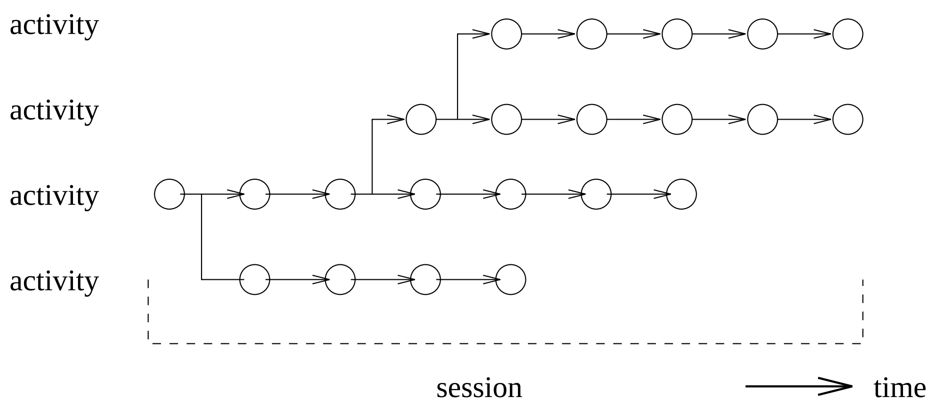

Figure 2.5: A User Session with Multiple Sequential Command Streams

the combined arrivals of all jobs for a user will illustrate these parallel properties, as it is only possible for a single user to give one job at a time.

Although there exist workload characterization studies for distributed systems, there is a lack of studies that use this type of characterization to synthesise a full-scale distributed system workload model. Many studies focus the characterization on an isolated task or function of the distributed system. In general, it is the scale, complexity, and vast amount of data that complicate the task of designing models for distributed systems. This difficulty may be the reason why few studies can be found in this area of research.

2.3.1 File System Studies

Although the data collected for our study were not detailed enough to study file usage and individual requests to the file server, understanding file usage should really be an intricate part of any distributed file system study. By understanding file size distributions, file request patterns, and locality of reference, we can achieve improved performance and availability in a distributed file system, as well as possibly providing insight into appropriate file caching and placement policies ([Sat81], [Sat90], [BR90], [Bla91], [Bla92], [GZS94]).

The five file system studies that we will review in this section have some general similarities. All of these studies demonstrate how a clear understanding of a system can be used to influence decisions for system design. In our workload characterization of the CDF system in Chapter 4, we will also examine the current state of the system to determine potential areas for system improvement.

[OdCH$^+$85] provided an extensive study of the UNIX 4.2 BSD file system, with the goal of collecting information that could aid in the design of a shared file system for a network of personal workstations. Data were collected at the logical level by continuous monitoring of three different systems, and analyzed to provide insight for a hypothetical network file system. The first significant finding was that on average each user does not use much file data, suggesting that network bandwidth is unlikely to be the bottleneck in network file systems. Secondly, most data are modified within a few minutes of being created, indicating a potential for large disk block caches. The third finding was that a significant reduction in disk I/O could be obtained when very large disk caches were used.

This study was repeated six years later and reported in [BHK$^+$91]. Dramatic changes in the speed of computers and the introduction of new network-based operating systems motivated the revisitation of this study. Many findings were quite similar to the findings of the 1985 study, but there were a couple of substantial differences revealed in the follow-up study. First, they found that file throughput had increased by a factor of 20 and had become more bursty. Also, the size of large files had increased by an order of magnitude. Their measurements of the Sprite file system caches showed an increased read hit ratio, suggesting that writes may eventually dominate file system performance, and thus new approaches may be necessary.

[BR90] examined file access behaviour in several commercial production VAX/VMS environments, using detailed I/O traces. Their hope was that this study would be useful for making design decisions for distributed file systems. They found that the average number of control operations per file was relatively small for most files, and that a small percentage of the active files accounted for a large percentage of the control operations. Most files were kept open for relatively short periods of time, and those that were reopened were done so not long after being closed. This type of information can be useful in making decisions about whether a write-back or a write-through caching policy should be used for the clients.

The Andrew File System (AFS) is a heterogeneous, large-scale, campus-wide distributed computing environment, which has been developed for the students, faculty, and staff at CMU ([How88], [Kaz88], [HKM$^+$88]). This is the system on which our original hierarchical load sharing simulator was to be based (see Section 1.3.1). An earlier paper [SHN$^+$85] focused on the design considerations that were made in the construction of the system's prototype. The primary goal was to support sharing of information between workstations. Other factors that influenced the design of this distributed system were: scalability, location transparency, heterogeneity, user mobility, and compatibility with existing operating system interfaces.

A synthetic model for a distributed system file server in a UNIX/NFS environment was designed in [Bod90]. The input workload to the file server was measured during the peak hours of a six-week period. This input was in the form of NFS file or directory operations that originated at the client workstations and were relayed over the Ethernet to the file server. Workload characterization was used in the preliminary analysis to determine the factors to be used in the workload model. The four key factors identified were: frequency distribution of file server requests, request interarrival distributions, file referencing behaviour, and distribution of sizes of read and write requests. A synthetic, externally-driven model was decided upon, with accuracy, flexibility, and reproducibility being of primary concern for the model. Rather than using simple statistics to represent the key components, representative distributions are used to generate the inputs to the synthetic model. This approach will also be used by the model designed in this thesis.

2.3.2 Network Studies

Local area networks (LANs) have been developing quickly, and heterogeneous local networks with a high degree of connectivity are now blossoming everywhere ([Tan88], [Kei89], [Sta94]). The scale of these systems is also expanding, producing a real need to develop techniques for modelling and simulating these interconnected networks ([CST88], [GT88], [FV90], [ES92]). There have been many studies which specifically focus on the network of a distributed system ([SH80], [Gon85], [Haw86], [CHKM88], [Bux89], [Kei89], [Gus90], [RVH94], [Sta94]). As in the file system studies, these studies carefully characterize the workload; however, in this case the workload is viewed from the network perspective.

Conventional workload characterization for LANs usually takes the form of measurements of the traffic that has been transmitted by the various stations in the network. This traffic analysis usually studies the packet size distributions and the rate of packet generation on the Ethernet. Such a characterization could prove insightful for capacity and configuration planning studies or for the examination of the efficiency of other protocols. It may also be used to determine if the current placement of components, such as those used for storage and processing, yields sufficient performance. Manufacturers and designers of the LAN components can use workload characterization to assess various design alternatives. [Haw86] summarised the design alternatives as follows:

- selection of technologies with different speed, cost, and error characteristics.
- imposing topology restrictions.
- specifying different parameters of media access control algorithms.
- choosing different partitioning of functions between adapters and machines.
- designing different approaches for movement of data between adapters and controllers.
- using alternative higher layer protocols and parameters within them.

Network workload characterization can also focus on the components or resources within a LAN. Such an approach is of particular interest for assessing design alternatives. [Haw86] listed the following categories of components that may be used in an analysis:

- technology (baseband, broadband, fiber optics, topologies, etc.).
- Data Link access methods (CSMA/CD, tokens, etc.).
- devices (adapters, machines, peripherals, etc.).
- protocols (higher level, network and intra-system).
- applications (traffic characteristics and resource demands).

Most early LANs used Ethernet (running at 3 or 10 mega bits per second (Mbps)) as the transmission medium. These typically used either the standardized token ring or token bus topology that are shown in Figure 2.6. The 100 Mbps fiber optic ring, called the Fiber Distributed Data Interface (FDDI), is also popular due to its astonishing speed. Access to the shared Ethernet medium is typically controlled by means of a protocol called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). This is the type of access protocol that is used in the CDF computing environment studied in this thesis. In this protocol, hosts wishing to send on the network “listen” to the broadcast channel until it becomes free, at which time the host can broadcast its message. To ensure reliable transmissions, acknowledgements are generated by higher level protocols. A *collision* occurs when two hosts attempt to transmit on the Ethernet at the same time. In this case, retransmission at a later time will be necessary.

An early study by Shoch *et al.* [SH80] examined the measured performance of the CSMA/CD protocol on a 2.94 Mbps Ethernet local network. Their measurements were performed by using a series of specialized test and monitoring programs. They captured the data passively by examining the state of the cable at individual stations. Their study

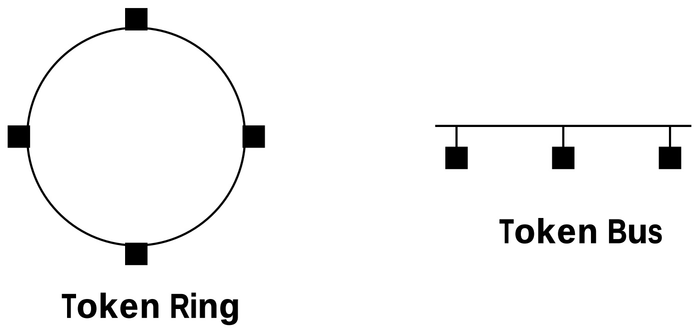

Figure 2.6: Ethernet Topologies

showed that the error rates were very low and that few packets were lost. Under normal load there were very few collisions, meaning that the access time for stations needing to retransmit was almost zero. Under an artificially-generated extremely heavy load, there were more collisions, but the resolution mechanisms still performed well and the channel utilization remained very high (almost 98%). They found that even under excessive load, the Ethernet channel did not become unstable. These observations led them to conclude that the Ethernet system is a particularly attractive choice as an architecture for local communication.

In recent studies by Raghavan *et al.* ([RVH94], [RVH95]), the networkload methodology was proposed for describing the workload that is generated by all of the clients of a network. Their hierarchical design of a networkload model, which is shown in Figure 2.7, identifies three levels of load generation patterns. The levels in the hierarchy, from top to bottom, are: sessions, commands, and client-server requests. At the session level the workload is a sequence of sessions, such as user login sessions. Session duration times and session interarrival times are parameters that could be used to represent the workload at this level. At the command level, the workload is the sequence of commands or events that make up a session. If the number of distinct commands is too large, various classes of commands that load the server in different ways could be used to represent the workload at this level. File size or the number of file requests would be typical parameters at this level of description. All of the load that results from the clients on the network is represented at the request level. Packet monitoring tools could be used to collect this type of information. Parameters such as the number of requests generated by the clients per time unit and the time taken by the server to satisfy requests generated by the clients are needed for characterization at this level. The hierarchical approach of network modelling that is examined in this paper will also be used to design our model in Chapter 6.

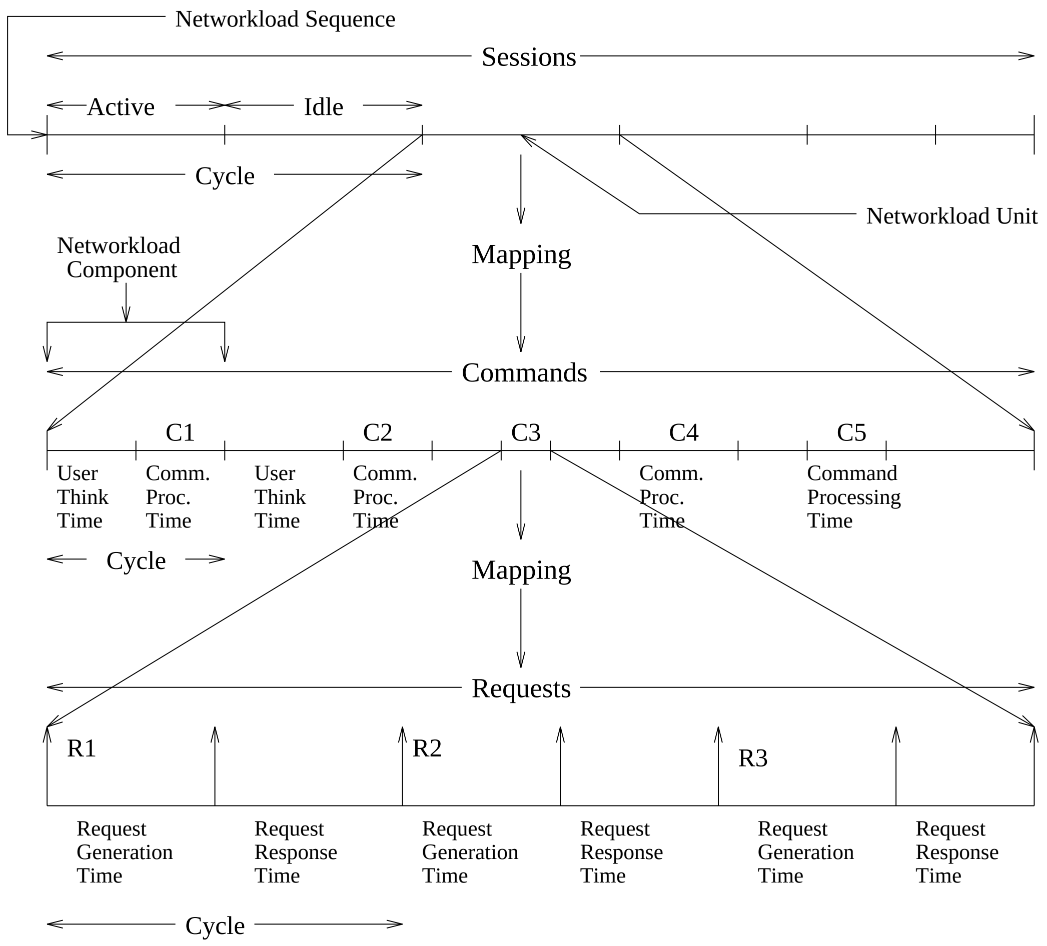

Figure 2.7: Hierarchical Structure of the Networkload Model

Although packet size distributions and generation rates can be successfully used to characterize and model network systems, they may not provide the required level of detail for some workload characterization studies. With the exception of the networkload model, most models really do not give a complete picture of the actual workload that is generated from the different workstations or computers in the network. In a sense, they concentrate on the effect, rather than the cause [Noe82]. In the next section, we examine some user classification studies that approach the workload in terms of the effect that individual users have on a computer system.

2.4 Classification of Users

An increasingly diverse user population has prompted research in the area of user classification. Depending on the system, the types of users may include students, teaching faculty, researchers, software producers, and administrators. Each of these groups of users may generate a significantly different workload. It is useful to study the behaviour of users to better understand the impact that certain types of users place on the system. If it is desirable to add a certain number of users of a known type to the system in a capacity planning study, a separate definition for each user type would be needed.

If similar user subsets can be identified and characterized, they can be used as the baseline for a system model. As explained in Figure 2.1 earlier in this chapter, the characterization of the workload may be done at different levels in the hierarchy. Classification of users in terms of the commands that users enter, sequences of commands, or the number of commands per session are considered to be at the functional level of workload characterization [Noe82]. Compared to characterization at the physical level, workload characterization based on this type of user classification is not as system-dependent. It is also possible to classify users at the physical level by considering the amount or rate of system resources that are consumed by commands issued by users. This type of approach is similar to the approach that is taken to classify users in this thesis.

There are a number of studies that focus on the importance of classifying and understanding the behaviour of users in a system. Studying user behaviour can be quite complicated, as there is a degree of unpredictability involved whenever classification of human behaviour is involved. The earlier studies concentrated on the commonly accepted "List-Mods-Run" user-behaviour model that was used to prepare computer programs. The most commonly used user classification technique, however, is cluster analysis. Cluster analysis has consistently given positive results in the area of user classification, and thus was chosen as the user classification technique to be used in this thesis.

In [Sub92], a user community using AFS file servers was studied. They broke the user workload down into groups of users that exhibited similar request patterns, and used these user classes in the design of an AFS file server model. Server usage data were collected on a per user basis and cluster analysis based on user type was employed. They showed that clusters were formed for distinct groups of file referencing patterns and file read/write size distributions. On one of the servers, clusters were evident for the number of disk I/O operations, mean response times, and the mean percent of calls requiring disk I/O for each of the user types. The users were divided into four groups that included technical writers, system administrators, students, and developers. A synthetic workload model was made based on the workload description for each user type. The number of users of each type were made configurable parameters to determine the performance of the server under different user configurations.

Noethe [Noe82] developed a different method for studying and describing user-behaviour characteristics in a system command language environment. Six groups of commands (general file, specific file, edit, compile, run and other) were initially identified. For each user session, the commands were associated with a six-component vector, based on how often commands from each of the six groups were used. Sessions with similar "edit" command percentages were grouped together. Then the sessions were subdivided such that those with a similar percentage of "compile" commands were grouped together. Finally, sessions were regrouped according to the percentages of "run," "general file," "specific file," and "other" commands. Since all groupings of the commands were made in one-dimensional space, determining the appropriate subdivision was straightforward. Although this method gave good results in this particular study, clearly it would not scale if the amount of data and number of parameters were significantly larger. It is also possible that the number of groups that would result from this method may be more than the number that is actually required to distinguish between classes of users. As no attempt is made to minimize the number of user classes, it is unlikely that this method would consistently result in a compact model.

The purpose of [TBMS90] was to produce accurate and comprehensive estimates of the user workload that can be used with computer network performance analysis tools. In this study, fourteen different generic user types were identified as the basic workload component of interconnected local networks. Nine different types of computer applications that were commonly used by these users were identified. Unlike most network workload studies that examine the network packets with no concern for from where or by whom they were generated, this study is more interested in identifying the impact that specific types of users and their most commonly used applications have on the system. The user workload is estimated (in packets per second) for each group of users based on their known connection time and packet generation rate. This workload is then used with a highly interactive network modelling tool called the "NetMod Network Designer" [BSST90].

The occasional user who performs activities that are non-characteristic of his or her user group may have a significant influence on the workload that has been deemed representative of that user group. It may be advisable to perform an initial workload analysis to identify these unusual users. If typical user behaviour is of concern to the study, it may be appropriate to remove these outlying users before the estimates of the workload in each user group are calculated.

The argument in [Blu94] was that most user classification studies focus on relatively homogeneous user populations, thus providing little insight into the range of diversity of user classes. Blumson examined the user behaviour of a very large distributed file system that was accessible to every member of the University of Michigan campus. They measured user activity based on file reads (number and size), file writes (number and size), directory reads (number and size), as well as the number of times that file status is read or written. File system operations were collected from the system over a period of several weeks. They initially divided the users based on their known role in the university and studied the means and standard deviations of these groups. This classification was destitute, as many groups had a lot of small users, thus detracting from the classification. They rectified this problem by adding a subcategory called "casual users." They provided a table of their final user classifications, which do show clear differences among the various categories of users.

2.5 Cluster Analysis

Cluster analysis is a multivariate statistical tool that can be used to identify patterns or groupings within a set of data. It was first used in the fields of biology and zoology as a means of classifying plants and animals of various species. It is a way of placing objects into groups suggested by the data, such that objects placed in a particular group are more similar to each other in some sense, than to objects placed in other groups. Figure 2.8 shows how three distinct clusters are selected from a set of data. There are many good general references devoted the subject of cluster analysis ([And73], [SS73], [Har75], [Spä80], [Spä85], [Sil86], [KR90], [Eve93], [Boc94]).

2.5.1 General Introduction

In cluster analysis, a way of determining the "distance" or "similarity" between pairs of objects must be defined. Generally the objects being clustered are thought of as points in an n-dimensional space, where n is the number of variables that are used to determine the distance between pairs of objects. The Euclidean distance between two points in space is a commonly used similarity measure. Two objects that have a small Euclidean distance

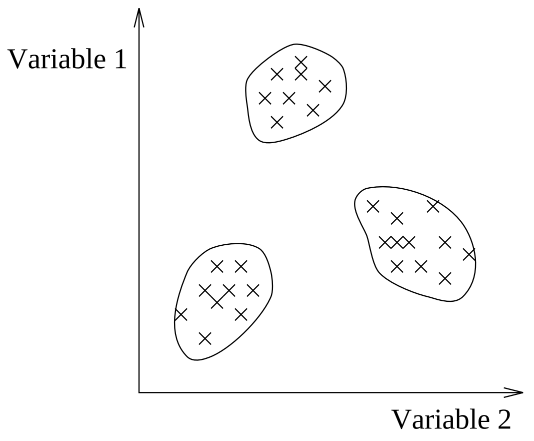

Figure 2.8: Three Distinct Clusters

between them are considered to be "similar" and are likely to be placed within the same cluster or group. The Euclidean distance D in R_n between two objects, f_a and f_b, is calculated as

$$D(f_a, f_b) = \sqrt{\sum_{i=1}^{n} (f_{a_i} - f_{b_i})^2}. \tag{2.3}$$

A criterion must be specified that can be used to determine which objects are assigned to which clusters. A particular clustering method will allow the user to state the similarity measure that will be used to determine the distance between two objects, and the criterion for determining the distance between an object and a cluster. There are many different clustering methods available, each with varying approaches to these issues.

Clustering methods are generally classified as either *hierarchical* or *nonhierarchical*. Nonhierarchical methods include *iterative* methods such as the k-means method. These types of methods start with an initial subdivision of k classes. The selected criterion is used to iteratively improve the selection of the classes by shifting the elements of one cluster into another. The *centres of mass* (or cluster centroids) of the clusters are recomputed after each new cluster assignment. Using the notation from Equation 2.3, the centre of mass C is defined as $C = (c_1, c_2, \ldots, c_n)$, where $c_i = \frac{1}{k}\sum_{j=1}^{k} f_{j_i}$. k is the number of objects in the cluster, and n is the number of dimensions.

Hierarchical methods include such methods as minimal spanning tree, single-linkage, and Ward's method [Eve93]. Each observation begins as a cluster itself, and then the next "closest" cluster is merged in each step, until there is only one cluster left. A tree can be

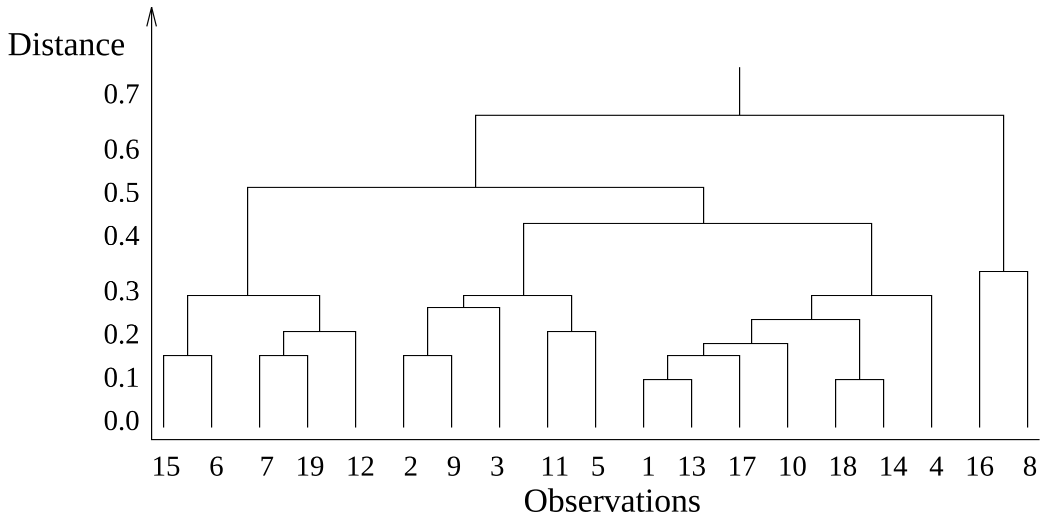

Figure 2.9: Dendogram Showing how the Merging of Observations Forms a Tree

constructed to represent this merging of points. A particular level in the tree corresponds to the partitioning of the data for a certain number of clusters. Dendograms like the one shown in Figure 2.9 are used to graphically display the tree. The individual observations are listed along the horizontal axis, while the distance between pairs of observations is shown on the vertical axis.

It should be noted that often the parameters that are used in the clustering process are not homogeneous. They must be scaled to homogeneous units so that it is meaningful to compare the variables with each other. Chapter 3 in [And73] discusses some of the many normalization techniques that may be applied. Several papers ([AMB76], [BCMS82], [Art86], [FGK88], [MC88], [DI89], [Raa93]) discuss how to deal with the issue of scaling data prior to cluster analysis. The cluster analysis method to be used and the type of data to be analyzed will determine which scaling method is most suitable.

Although cluster analysis is a widely accepted method for finding groupings in data, it is not without fault. The most common difficulty cited is determining the number of clusters in a data set. Several techniques have been identified that attempt to predict the number of clusters in a sample. [Mil80] and [MC85] examine numerous criteria for determining the number of clusters in data sets known to have different types of clusters. The two graphs in Figure 2.10 show a data set for which two different sets of clusters have been chosen. In the first graph five clusters are identified, whereas in the second graph, three clusters are identified. It is important to note that the three-cluster and five-cluster configurations are both good choices for this particular data set. The intended use of the clusters should be used to make decisions about the number of clusters to be chosen.

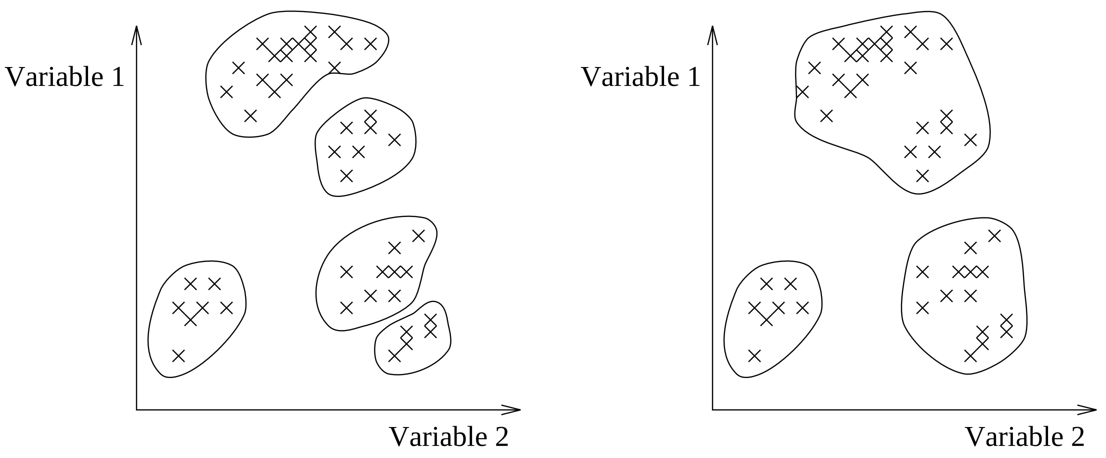

Figure 2.10: Different Numbers of Clusters for the Same Data Set

There is a plethora of clustering methods available. This complicates the matter of choosing the method that is most appropriate for the study at hand. Different clustering methods are usually biased in different ways, depending on the nearness criterion used. Some methods are biased towards finding equal-sized clusters, others towards finding clusters with similar variance, and still others towards finding elongated or spherical clusters. Figure 2.11 shows how different methods may choose different clusters for the same set of data. It is sometimes helpful to visualize your data to gain insight into which method might be most appropriate. The analyst should experiment with a wide range of method types, and choose the one that seems most appropriate for her or his particular set of data.

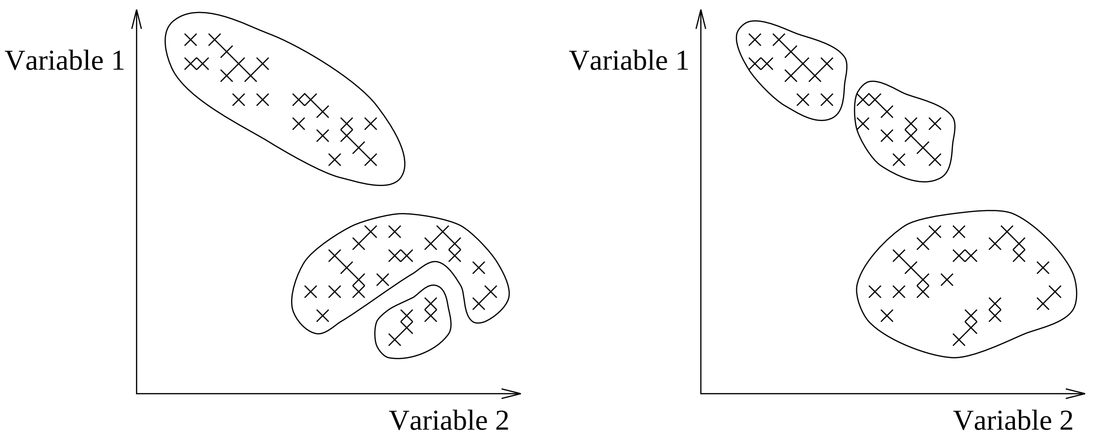

Figure 2.11: Alternative Three-Cluster Classifications for the Same Data Set

2.5.2 Workload Characterization and Modelling

Cluster analysis has been highly utilized in workload characterization and model design for computer systems ([Chu79], [AMB76], [Ara78], [DI89], [CS94], [SC84], [Art86], [Hug83], [Raa93]). It is the ability of cluster analysis to extract meaning from multiply-dimensioned data that makes it such a favourable choice in workload characterization. In workload studies, often massive amounts of highly-dimensional data are collected, making it impossible to decipher the data using only manual methods. Cluster analysis is a practical tool that can be used to determine similar components in the data for inclusion in a workload model, and for the design of compact system models.

In the case of a workload study, the intended purpose of the model should be used to guide decisions, such as choosing a clustering method and determining the number of representative clusters. If a compact system model is required or if it is desirable to reduce the number of parameters in a model, then a smaller number of clusters will have to be chosen, even though the objects in the clusters may be more dispersed about the cluster centres. The decision about which clustering method to use should be guided by the workload characterization of the data and the known biases of the various clustering methods. Once the analyst understands the type of data that he or she is dealing with, this should simplify the problem of choosing the appropriate clustering method.

Many of the studies mentioned earlier in this chapter have used cluster analysis. Cluster analysis has been used in workload characterization and designing models for batch systems ([HDG71], [Fer72], [AMB76], [Ser81b]), as well as in the design of static models for interactive systems ([Ara78], [AM78], [Chu79], [Fer81], [BCMS82]). Many dynamics studies have also used cluster analysis ([Har82], [Har83], [Hug84], [CIS86], [CF86]).

[BCMS82] built a general package that could be used to implement a static workload model. The input required by the package is the process accounting records that are generated by a system. The classification of the workload components is attained by a clustering algorithm. Since it is hard to know which clustering algorithm is best a priori, the popular k-means algorithm is used by default in the package. This choice may or may not be desirable, depending on the type of data to be analyzed. The standardized Euclidean distance is used to determine the distance between objects. The overall mean square ratio and the ratio between the inter-cluster variances are used to determine the number of clusters to be used in the model. Although an automated model generator that is based on cluster analysis may give good results for some data sets, it is questionable whether the generalization that has been applied to its clustering algorithms will make it transferable to a diverse set of computer systems.

In model design, cluster analysis is often used to determine a number of representative clusters, and then the centroids of these clusters are used to represent the clusters (or classes) in the model. As the characteristics of the clusters may be dispersed about the cluster centroids, this approach may not always produce the most representative model. Calzarossa and Ferrari [CF86] used cluster centroids to examine the sensitivity of cluster selection by taking an experimental approach. They grouped the workload data into command classes with similar resource usage characteristics and used these classes as input to the simulation of a single-server model. The paper concludes by stating that under the assumptions outlined in the paper, clustering techniques for executable workload model design work well.

We conclude our discussion of cluster analysis by listing three issues that motivated the use of cluster analysis in this thesis. These issues were lacking in the cluster analysis literature.

1. Cluster analysis is performed on "live" data, which according to [Sco82], is an area of cluster analysis that has had limited research. The ability of cluster analysis to procure meaning from a raw set of data is explored. Different clustering methods are compared to determine which method is most appropriate for the data set at hand.

2. In this thesis, a class-based model that is developed using cluster analysis is used to develop a distributed system model. As cluster analysis has more commonly been used to study centralized systems, its application in this area furthers the research for distributed systems.

3. The problem of how many representative clusters should be chosen is too often glossed over in the literature. This thesis inspects this problem, and gives somes guidelines that can be used to determine appropriate numbers of clusters to select.

2.6 Focus of this Thesis

This thesis is concerned with studying the workload of an academic computer system, and using this information to design a model for a distributed file system, while taking its compactness into consideration. The workload is characterized to provide a general idea of how the system is operating. The study applies some of the techniques that were first used in the analysis of centralized systems to build a model for a distributed system. It considers the dynamic properties of the workload, but does not concentrate on these aspects in the model design. It uses cluster analysis to classify both users and commands. The classes are studied in detail and then blended into a synthetic workload model of the distributed file system that would be suitable for a simulation study.

The studies that have been examined in this chapter are by no means an exhaustive search of all of the literature in the area of workload characterization and modelling. These studies have been chosen because of their relevance to this particular study and comprehensiveness. We have attempted to give a broad overview of this area and an indication of where our study fits in.