May 1991

UILU-ENG-91-2229
CRHC-91-19

Center for Reliable and High-Performance Computing

LANGLEY GRANT
IN-61-CR
-13937
P.49

A USER-ORIENTED SYNTHETIC WORKLOAD GENERATOR

Wei-lun Kao

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: None
a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S): UILU-ENG-91-2229 CRHC-91- 19	5. MONITORING ORGANIZATION REPORT NUMBER(S)
a. NAME OF PERFORMING ORGANIZATION: Coordinated Science Lab University of Illinois; 6b. OFFICE SYMBOL (If applicable): N/A	7a. NAME OF MONITORING ORGANIZATION: NASA
c. ADDRESS (City, State, and ZIP Code): 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code): NASA Langley Research Center Hampton, VA 23665
a. NAME OF FUNDING/SPONSORING ORGANIZATION: NASA; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: NASA NAG-1-613
c. ADDRESS (City, State, and ZIP Code): NASA Langley Research Center Hampton, VA 23665	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

1. TITLE (Include Security Classification)

A User-Oriented Synthetic Workload Generator

2. PERSONAL AUTHOR(S)

Wei-lun Kao

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM ____ TO ____	May 1991	39

6. SUPPLEMENTARY NOTATION

7 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

workload, user-oriented, model, measurement, file system

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A user-oriented synthetic workload generator that simulates users' file access behavior based on real workload characterization is described. The model for this workload generator is (i) user-oriented and job-unspecific, (ii) represents file I/O operations at the system call level, (iii) allows general distributions for the usage measures, and (iv) assumes independence in the file I/O operation stream. The workload generator consists of three parts which handle specification of distributions, creation of an initial file system, and selection and execution of file I/O operations. Experiments on SUN NFS are shown to demonstrate the usage of the workload generator.

0. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

D Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

A USER-ORIENTED SYNTHETIC WORKLOAD GENERATOR

BY

WEI-LUN KAO

B.S., National Taiwan University, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1991

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

MAY 1991

WE HEREBY RECOMMEND THAT THE THESIS BY

WEI-LUN KAO

ENTITLED A USER-ORIENTED SYNTHETIC WORKLOAD GENERATOR

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

[illegible]
Director of Thesis Research

Head of Department

Committee on Final Examination†

Chairperson

† Required for doctor's degree but not for master's.

O-517

ABSTRACT

A user-oriented synthetic workload generator that simulates users' file access behavior based on real workload characterization is described. The model for this workload generator is (i) user-oriented and job-unspecific, (ii) represents file I/O operations at the system call level, (iii) allows general distributions for the usage measures, and (iv) assumes independence in the file I/O operation stream. The workload generator consists of three parts which handle specification of distributions, creation of an initial file system, and selection and execution of file I/O operations. Experiments on SUN NFS are shown to demonstrate the usage of the workload generator.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Professor Ravi Iyer, for his excellent guidance and encouragement. Thanks are also due to Dr. Rene Llames for many useful discussions. I would also like to thanks all my friends, especially those in CRHC for their help. Finally but not least, I would like to thank my parents, brother and sister for their support and encouragement.

This research was performed at the Illinois Computer Laboratory for Aerospace Systems and Software (ICLASS).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Experiments and simulations are useful in designing and comparing computer systems. To generate workload for such experiments or simulations, trace data, benchmarks, and synthetic programs are usually used. Trace data reproduces the actual workload, but provides an inflexible description and requires much memory. Therefore, it may not be suitable for every study. Benchmarks and synthetic programs generate artificial workload, but require careful interpretation of the results.

This thesis introduces a user-oriented synthetic workload generator that simulates typical users accessing files and performing computations. The generator is designed for experiments and simulations related to file systems, such as measuring the performance of a particular file system or comparing several file systems. Because the generator simulates user behavior, it is machine-independent. Therefore, user behavior in a centralized and distributed system, consisting of possible different types of machines, can be simulated. Also, different load intensities (i.e., the number of users using a computer concurrently) can be simulated. The workload generator takes as input a set of distributions of various file usage parameters, rather than only their mean values, and can therefore generate more realistic workload than benchmarks, and more representative measurement or simulation results.

Unlike previous studies, the workload generator models file access operations at the system-call level. The generator reproduces many aspects of resource usage, including file access operation types, number of files, file size, amount access per operation and think time. These characteristics reflect primarily the behavior of applications, rather than of operating systems or machine architectures. Hence, the workload generator provides a portable description of resource usage behavior.

The workload generator consists of three parts. The first part is an interactive graphic interface for users to specify or modify the distributions of various parameters. The distributions may be phase-type exponential or multi-stage gamma distributions and may be specified in the form of tables of probability density or cumulative distribution values. The second part is a file system creator that builds a file system according to user-specified parameters. The third part of the generator simulates a number of users.

The current version of the workload generator is written in C++, uses the X11 window system to display distributions graphically, and assumes a UNIX operating system environment.

The rest of the thesis is organized as follows. The related research and the objective are discussed in Chapter 2. In Chapter 3, the workload model is presented. Chapter 4 describes the design and implementation of the workload generator. Chapter 5 gives several examples for the usage and application of the workload generator. The last chapter summarizes the thesis and suggests future work.

2. BACKGROUND

2.1 Related Research

Experiments and simulations to improve existing systems or to design new systems require workload generators to drive them. Previous research on workload generators can be divided into three types: trace data, benchmarks, and synthetic programs.

Trace data is collected from a system under certain workloads. For instance, Ousterhout, et al., traced the file usage in a UNIX 4.2 BSD file system and used the trace data in experiments on file system cache sizes [ODCH+85]. Hunt, et al., measured the resource usage of batch jobs in a university computer center, grouped the jobs into four clusters, and performed statistical analysis [HDG71]. The advantage of using trace data is that the data represents the workload exactly. However, there are several disadvantages. First, the amount of data is very large. Secondly, it is necessary to remember the initial state of the file system. Third, the trace data, in general, is specific to the particular configuration of the system from which it was collected and cannot therefore be used to simulate a different configuration. Also, it is not usually possible to arbitrarily modify the data to produce other kinds of workloads, such as one representing a different number of users. In other words, the method of using trace data has low flexibility and is restricted to experiments on existing systems.

Benchmarks are programs or jobs which are selected or designed to simulate real workload. For example, Cabrera used compilation and execution of a CPU-bound program, a command "*man man*," and a script as benchmark to measure three computer systems running UNIX [Ser86]. Howard, et al., used a script, consisting of *makedir*, *copy*, *scandir*, *readall* and *make*, to compare Sun NFS with Andrew file system [HKM+88]. Benchmarks are useful in comparing CPU and I/O speed among different computers or systems. However, not all of the users perform the same tasks, so statistical analysis is needed to demonstrate the representativeness of benchmarks. Generally, benchmarks cannot exactly represent the real workload.

Synthetic programs are artificial programs with parameters which can be adjusted such that the behavior similar to that of real workloads is exhibited. For example, Buchholz proposed a synthetic yardstick job which simulated a general file update process [Buc69], and Wood and Forman showed that it was a practical tool [WF71]. However, the job was designed for batch systems and needed modification for interactive systems. Sreenivasan and Kleinman generated workload by using a collection of individual synthetic programs consisting of Buchholz's synthetic programs with six parameters (e.g. the number of master and detail records, the blocksize, and the size of records) [SK74].

Babaoglu modeled virtual memory references and used parameters to produce the reference pattern [Bab81]. He also considered overhead, and tried to minimize it to increase the validity of the synthetic program. Hughes clustered many real jobs into several groups, and used a Markov process to model the workload. The transition matrix was used to produce workloads [Hug84]. Barrington wrote a synthetic workload generator based on a user-oriented analysis of file usage [Bar86]. His analysis considered user files and notes files, but ignored system files, CPU usage and interarrival time of I/O operations. Synthetic programs combine trace data and benchmarks, so it is more flexible and realistic, although certain independence assumptions are made.

2.2 Objective of this Research

In this section, we discuss desirable properties for a workload generator. Domanski presented a comprehensive survey and suggested several criteria: portability, maintainability, adaptability, comprehensibility, and credibility [Dom82]. A good workload should:

- be portable, i.e., it should be machine-independent, and operating system independent;
- be flexible, i.e., it should be able to produce different kinds of workload, from that generated by a single user to hundreds of users;
- consider the variation user behavior; that is, not all the users do the same things;

- include different resource usages, such as CPU, I/O, and so on;
- be amenable to statistical tests of similarity to the real workload; and,
- be easy to maintain.

This thesis describes a user-oriented synthetic workload generator to meet these criteria. Our method analyzes trace data to obtain the distributions of resource usage of users and then uses the distributions during the simulation phase. Thus, the workload generator is based on real workload, can produce single-user as well as multi-user workload, and is capable of representing different user behaviors. It is machine-independent and operating system independent. It is written in C++, and the graphic facility runs on the X11 window system.

3. WORKLOAD MODELING

This chapter describes the workload model used in generating file I/O. In particular, we discuss the various degrees of freedom in building a model and state the reasons for our specific modeling decisions. In general, our decisions were guided by the intended application of the workload generator, namely, to drive experiments and simulations related to file systems. Specifically, the generator should be able to simulate the behavior of different user populations, where a population is characterized by the number of user types and the number of users of each type. The generator should have little or no language or machine dependence.

3.1 Modeling Choices

Workload models can be classified according to

- Granularity (user-oriented or user-unspecific; job-oriented or job-unspecific)
- Level of description (language, kernel or physical level)

- Measure of variability (mean values or distributions)

- Dependence in the operation stream (independent, Markov, or time-series).

The following four sections will discuss each of the above dimensions, and explain the reasons for our design choices.

3.1.1 Granularity

The first decision to be made in designing our workload model concerns the granularity of the description, i.e., whether the model should be user-oriented or user-unspecific (system-wide). A user-oriented model classifies users into several types and provides a description of the behavior of each type (Section 4.2.3). On the other hand, a user-unspecific model describes the workload due to all users. That is, it is concerned only with the aggregate effect of all users. A user-unspecific model is simpler than a user-oriented model because it consists of fewer parameters; a user-oriented model is more flexible because it allows us to describe the behavior of different user populations. We choose a user-oriented model for its flexibility.

Another decision to be made is whether the model should be job-oriented or job-unspecific. A job-oriented model classifies jobs into several groups and provides a description of the behavior of each group. A job-unspecific model describes the workload due to all jobs. A job-oriented model greatly complicates the workload generator and increases its space requirements, but does not provide any more information which would be useful for the generator's intended purpose. Therefore, we chose a job-unspecific model.

3.1.2 Level of description

File I/O can be described at several levels, namely, language, kernel, or physical level.

A language-level characterization describes usage of the various I/O operation available in a particular high-level language. For example, the C language provides such functions as *getc*, *fprintf*, etc. For better performance, some language libraries, such as that for C, maintain a buffer for each file and manage it themselves. A characterization at this level is obviously language-specific.

In operating systems such as UNIX, which prevent users from accessing devices directly but instead provide indirect interfaces, file I/O can also be studied at the kernel level. The interface in UNIX systems appears in the form of system calls, e.g., *open*, *read*, and *ioctl*.

At the lowest, or physical level, system calls to perform file I/O operations appear as disk controller commands. The operations at this level are hardware-specific.

The appropriate choice of level depends on the purpose of the workload generator. For example, we would choose the language level to improve the performance of a particular language, the kernel level to tune an operating system, or the physical level to compare disk drivers. We chose kernel level (or system call level in UNIX systems) as the appropriate level at which to model the workload because our purpose in designing a workload generator is to use it in experiments on file systems.

3.1.3 Measure of variability

Workload can be quantified in terms of total (or mean) values of the usage measures, or in terms of their distributions. If we assume the distributions are exponential, mean values are sufficient to represent behavior, but previous studies have shown that the distributions are not necessarily exponential [DI86, Dev88]. Our model uses distributions to represent the variability in the usage measures, but does not assume the distributions to be exponential. Distributions can be specified in tabular form or in parametric functional form. This generality enables the workload generator to produce more realistic output.

3.1.4 Dependence in the operation stream

A model for the stream of file I/O operations associated with a user can assume that each operation is independent of previous ones. Alternatively, a model can allow an operation to depend on the n previous operations. The dependence is usually expressed in the form of a Markov model (for $n = 1$) or a time-series model (for $n > 1$). For simplicity, we assume independence, subject to obvious logical constraints; for example, an *open* must precede any *read* or *write* . For file I/O operations at the system call level, it is not clear whether the added complexity of modeling dependence in the operation stream is justified. This is an open research question.

3.2 Summary

The decisions made in building the file usage were discussed. The model is user-oriented and job-unspecific, represents file I/O operations at the system call level, allows general distributions for the usage measures, and assumes independence in the I/O operation stream.

4. DESIGN AND IMPLEMENTATION

This chapter describes the design and implementation of our workload generator, and discusses its properties and problems.

4.1 Structure of the Workload Generator

The workload generator simulates users accessing files by generating file I/O operations using specified distributions. The generator can drive a real or simulated file system. When used to drive a real file system, the file I/O operations "generated" are actually executed and, to avoid modifying or destroying existing files, a new file system is created to which file I/O is directed. In this new file system, only those files which may be accessed need to be created. In the remainder of this thesis, we assume that the generator is used to drive a real, rather than simulated, file system.

The block diagram of the workload generator is shown in Figure 4.1. First, file distributions and usage distributions must be specified. These are used to compute tables of cumulative distribution function (CDF) values for use in random number generation.

An interactive graphic interface is provided for input, display, and modification of the distributions. Users can fit a phase-type exponential or multi-stage gamma distribution to an empirical distribution, or supply the probability density function (PDF) values or CDF values directly. Second, a new file system is created. Files are created based on the CDF tables of file distributions computed in the first part. Finally, file I/O operations are executed to simulate users using the computer. The operations are selected based on the CDF tables of usage distributions computed in the first part. The details are described next.

The workload generator is composed of three parts. The Graphic Distribution Specifier (GDS) allows users to input, fit and modify distributions. The File System Creator (FSC) creates a new file system. And the User Simulator (USIM) executes file I/O operations to simulate users.

4.1.1 Graphic Distribution Specifier (GDS)

The GDS is an interactive graphic interface that allows users to input, fit and modify distributions. It uses the X11 window system to display distributions. If the X11 window system is not supported, the GDS can still be used to specify distributions, but no graphical display will be available. Since actual file and usage distributions have been shown to be well approximated by multi-stage gamma distributions [DI86], the GDS supports multi-stage gamma distributions. Fitting of phase-type exponential distributions is also supported because these can represent any type of distribution. Thus, users can use either of these two distribution families to represent the empirical file and usage

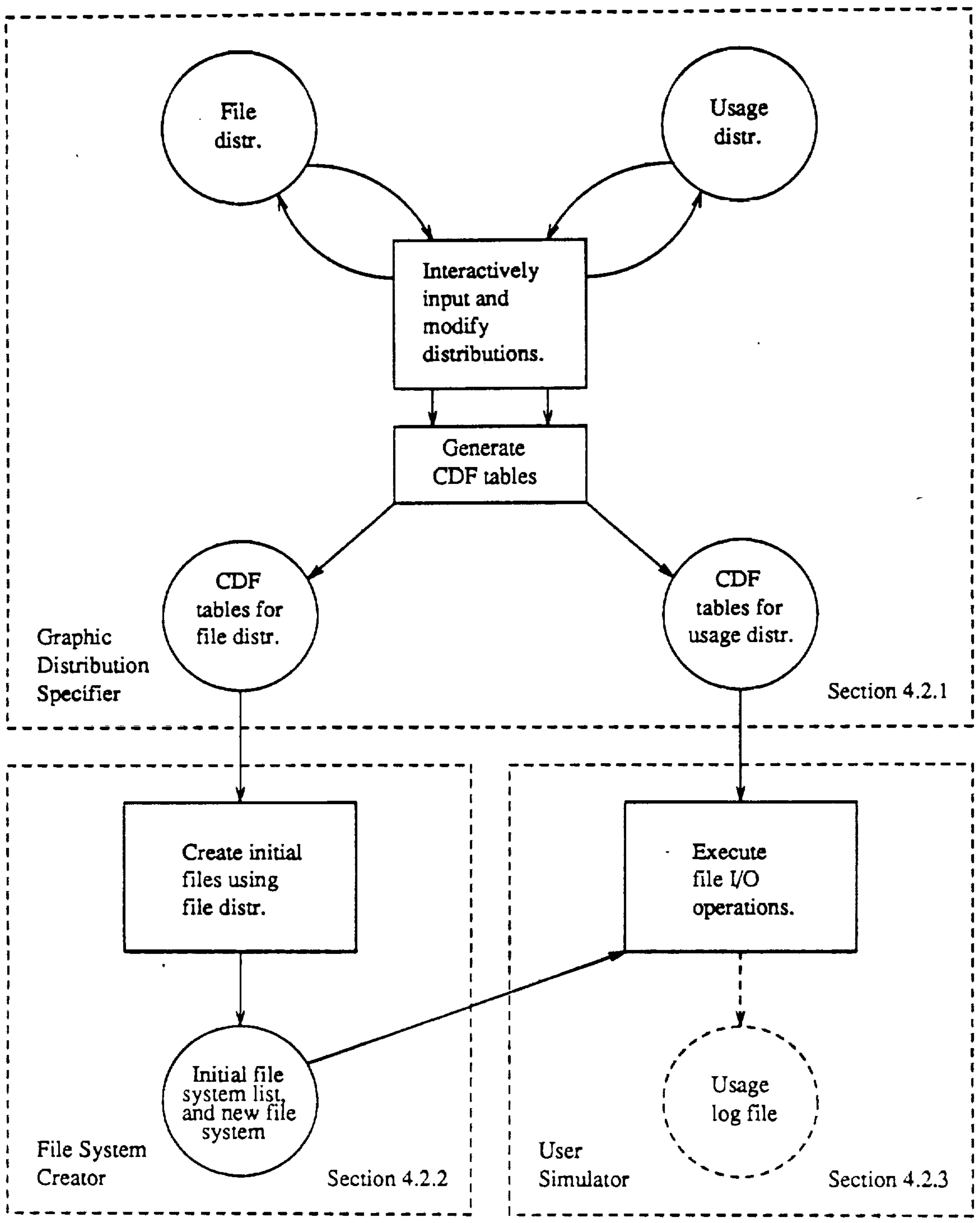

File distr.
Usage distr.
Interactively input and modify distributions.
Generate CDF tables
CDF tables for file distr.
CDF tables for usage distr.
Graphic Distribution Specifier
Section 4.2.1
Create initial files using file distr.
Initial file system list, and new file system
File System Creator
Section 4.2.2
Execute file I/O operations.
Usage log file
User Simulator
Section 4.2.3

Figure 4.1: Block diagram of the synthetic workload generator.

distributions, or supply the PDF or CDF values directly. Finally, the GDS creates CDF tables for the FSC and the USIM. To compute CDF values from PDF values, Sympson's method for numerical integration is used.

4.1.2 File System Creator (FSC)

The FSC builds a new file system according to the file distributions for each file category. Note that many files are not referenced. For the file distributions, we only need to consider those files which were accessed during the measurement. By this, we greatly reduce the size of the new file system.

We classify files into two basic types: system files and user files. Directories are treated as special files. However, users can define other types of files for their particular file system. Each type of file is associated with a size distribution.

In the new file system, we create a directory for system files, and several directories, one for each virtual user. Files in the system directory and a user's directory are created according to the file distributions.

4.1.3 User Simulator (USIM)

The USIM simulates workload on a terminal or workstation, i.e., a series of users logging in and using the computer. In particular, users' file accessing behavior is simulated. The USIM takes as input the following specifications: the number of users, the number of user types, the fraction of all users belonging to each user type, and for each combination of user type and file type, distributions of number of files accessed, file size and size accessed per operation.

Based on these specifications, the USIM repeatedly randomly selects a file access operation to be performed, the file on which to perform the operation, the amount of this file to access, and the time delay to the next operation.

4.2 Properties and Problems

In this section, we point out some properties and problems in the design of our workload generator.

- The workload generator creates a new file system, which requires disk space. This is done to avoid changing the state of the existing file system.
- Disk I/O due to swapping is not considered.
- The amount of memory required to store the CDF values for the distributions is the product of the number of user types, number of file types, and the number of sample values per distribution. This amount can quickly become prohibitively large for more than a small number of user types and file types.
- Variation in the behavior of a user over time is not considered. That is, the distributions are not time-dependent.
- Only sequential file access is simulated. This is not unrealistic considering that previous work has shown that in a university laboratory environment, for a majority

(86%) of the files accessed, the contents are either equally accessed or accessed at most once [DI86]. However, in other environments, such as a commercial database system, nonsequential (or random) file access may be the predominant behavior.

- The new file system is assumed to reside completely within a single machine. A distributed file system cannot be currently created automatically. Users have to specify the locations of the files for a distributed file system environment.

- While the workload generator is portable across different computer systems, the file and usage distributions obtained from one system are not necessarily representative of other systems.

4.3 Summary

The workload generator consists of three parts, namely, the Graphic Distribution Specifier (GDS), the File System Creator (FSC) and the User Simulator (USIM). The GDS is an interactive graphic interface for input, display and modification of distributions. The FSC creates a new file system to avoid changing the state of existing file systems. The USIM simulates users by executing file I/O operations. Some properties and problems in the workload generator were discussed. In Section 6.2, we discuss some future work which addresses some of these problems.

5. USAGE AND APPLICATION

To illustrate the usage and application of the Workload Generator, several examples are shown in this chapter. Section 5.1 demonstrates the use of the Workload Generator. Section 5.2 shows the measurement of Sun NFS using the Workload Generator as load control. Section 5.3 suggests a procedure of comparing different file systems.

5.1 Usage of the Workload Generator

The Workload Generator consists of three part. The Graphic Distribution Specifier (GDS) is an interactive graphic interface which allows users to specify arbitrary distributions and to display their density functions interactively. To fit an empirical distribution, users can specify phase-type exponential or multi-stage gamma distributions. A phase-type exponential probability density function is

$$f(x) = \sum_{i=1}^{N} w_i \, exp(\theta_i, \; x - s_i)$$

where w_i is the weight, s_i is the offset of the ith phase, N is the number of phases, the w_i's sum to unity, and

$$exp(\theta, y) = \frac{1}{\theta} e^{-\frac{y}{\theta}} \qquad 0 \leq y < \infty$$

A multi-stage gamma probability function is

$$f(x) = \sum_{i=1}^{N} w_i \, g(\alpha_i, \theta_i, x - s_i)$$

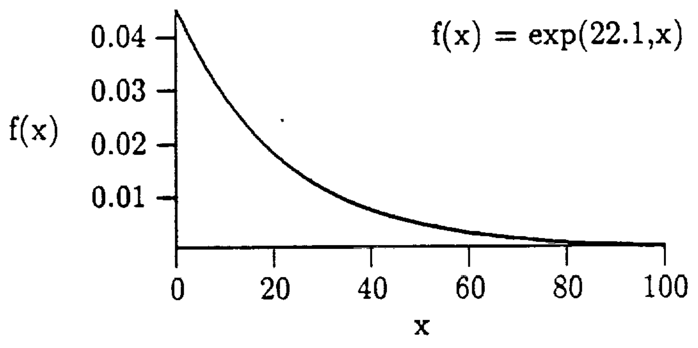
where w_i is the weight, s_i is the offset of the ith phase, N is the number of phases, the w_i's sum to unity, and

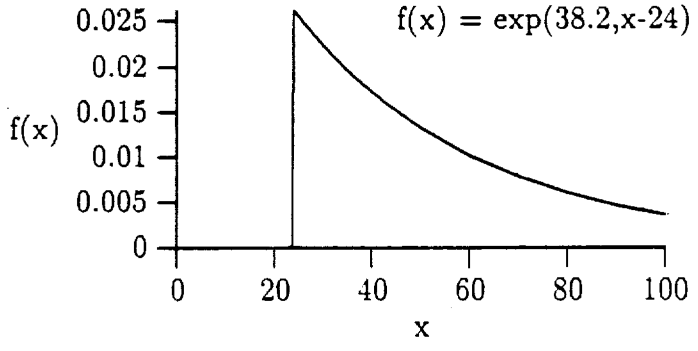
$$g(\alpha, \theta, y) = \frac{1}{\Gamma(\alpha)\theta^{\alpha}} y^{\alpha-1} e^{-\frac{y}{\theta}} \qquad 0 \leq y < \infty$$

Figures 5.1 and 5.2 show examples of phase-type exponential and multi-stage gamma distributions.

In the rest of this section, we would like to use an example which takes data in [DI86] and [Dev88] to show how to use the workload generator and visualize the results.

Before executing the User Simulator (USIM), a new file system must be created. An example characterization of the new file system is shown in Table 5.1. Since only the mean values of the characterizing measures are specified, as opposed to their distributions, it is necessary to assume some form of distribution for the measures to generate files for the new file system. We assume that the measures are exponentially distributed. These exponential distributions are specified to the GDS to created tables of cdf values. The File System Creator is then involved and supplied with these tables to generate a new file system.





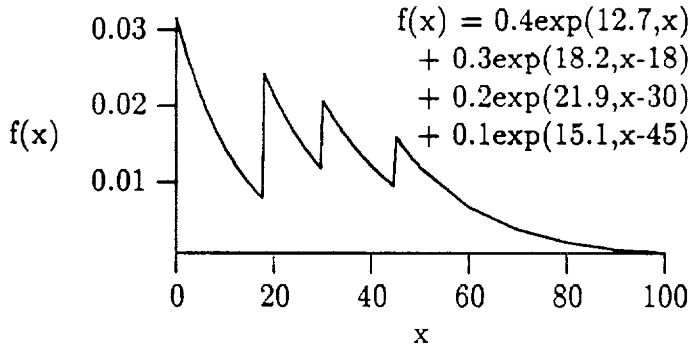


Figure 5.1: Examples of phase-type exponential distributions.

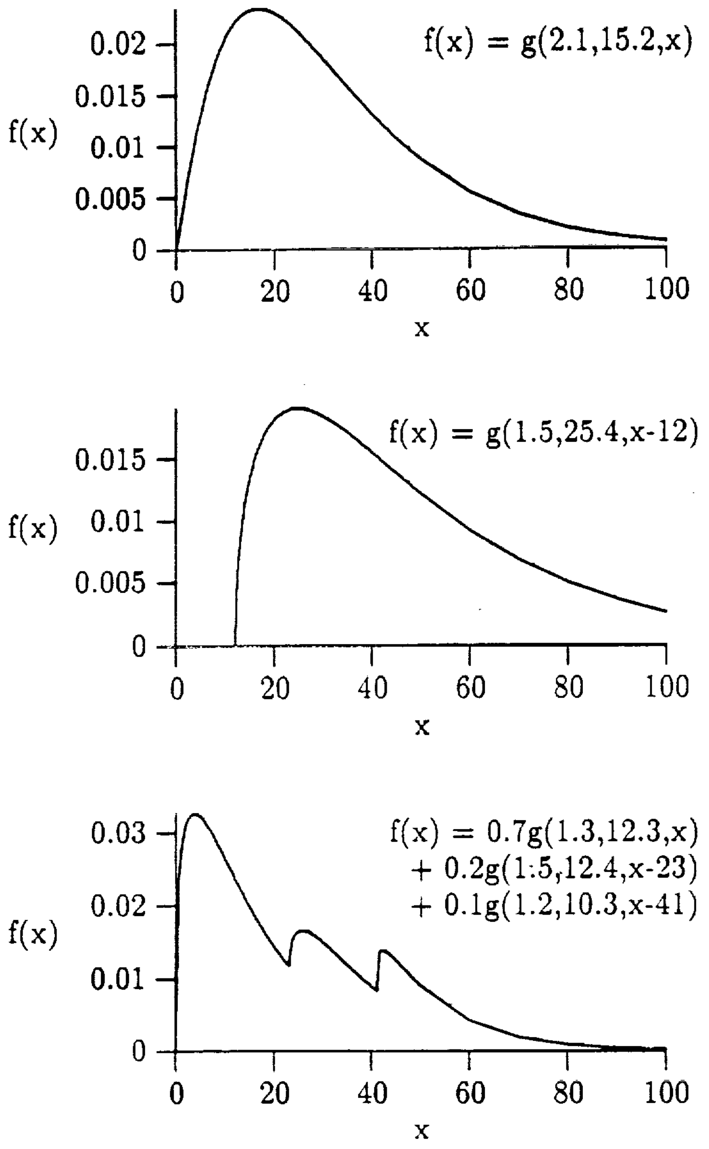


Figure 5.2: Examples of multi-stage gamma distributions.

Table 5.1: File characterization by file category.

file category			file size	percent of files in category
file type	owner	type of use		
DIR	USER	RDONLY	714	7.7
	OTHER	RDONLY	779	3.4
REG	USER	RDONLY	5794	21.8
		NEW	11164	9.7
		RD-WRT	17431	4.6
		TEMP	12431	38.2
	NOTES	RDONLY	31347	6.4
		RD-WRT	18771	3.2
	OTHER	RDONLY	15072	5.0

To run the USIM, the usage distributions must be specified. The file usage characterization of typical users is shown in Table 5.2. As in the characterization of the new file system, the usage measures are specified in terms of mean values only; the measures are assumed to be exponentially distributed. For the access sizes of file access system calls, we assume they are exponentially distributed with a mean of 1024 bytes. Think time (inter-I/O-request time) is also assumed to be exponentially distributed with a mean of 5,000 microseconds. These exponential distributions are specified to the GDS to create tables of CDF values. The USIM is then involved and supplied with these tables to simulate the file accessing behavior of users.

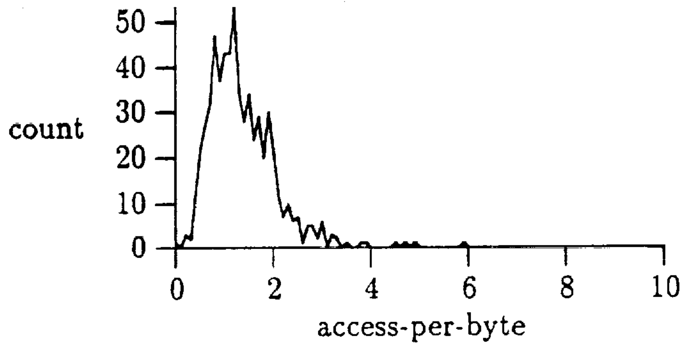
After simulating 600 login sessions, the system-wide file usage distributions are shown in Figures 5.3–5.5. These graphs show the distributions of average access-per-byte, average file size and average number of files referenced as in [DI86] and [Dev88]. For each

Table 5.2: User characterization by file category.

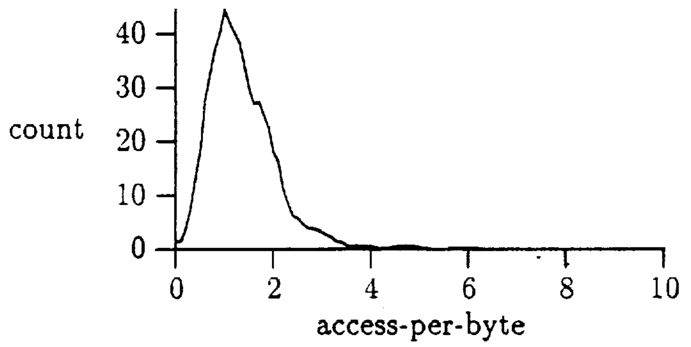
file category			characterizing measures			percent of users
file type	owner	type of use	accesses	file size	files	accessing category
DIR	USER	RDONLY	3.28	808	2.9	69
	OTHER	RDONLY	2.28	1198	2.5	70
REG	USER	RDONLY	1.42	2608	6.0	100
		NEW	2.36	11438	4.0	40
		RD-WRT	3.50	19860	2.2	46
		TEMP	2.00	9233	9.7	59
	NOTES	RDONLY	0.75	53965	11.3	53
		RD-WRT	1.77	20383	5.7	38
	OTHER	RDONLY	2.11	13578	3.1	55

graph, we show the empirical distributions before and after smoothing. There is also a program, Usage Analyzer, for users to analyze the results and display them graphically.

In evaluating the performance of file systems, one popular performance index is response time. In this example, the response time of each file I/O system call was measured by getting the difference of before and after calling a system call. The example simulation in this chapter was performed on a SUN 3/50 workstation with a local disk, but all the files accessed were stored in a SUN 4/490 file server. The network file system used was the SUN Network File System (NFS). The mean and standard deviation of the access size and response time for file-access-related system calls are shown in Table 5.3. If we change number of users or usage distribution of users, we can obtain different response times, and then compare the results to tune the file system or to decide to change file systems.

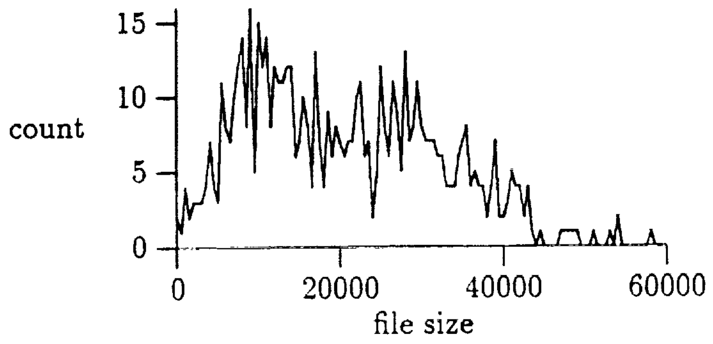


(a) Before smoothing

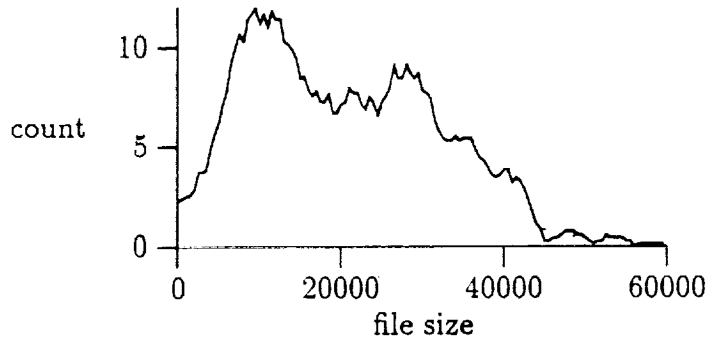


(b) After smoothing

Figure 5.3: Average access-per-byte.

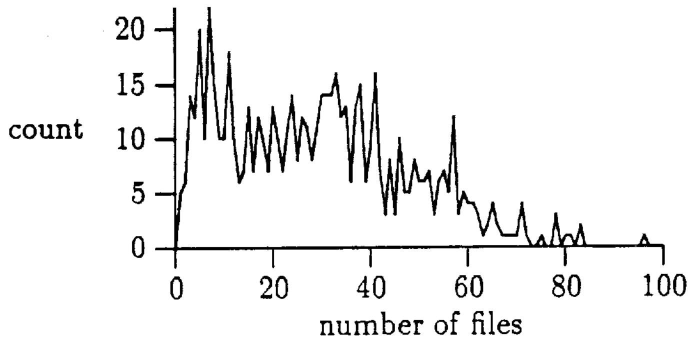


(a) Before smoothing

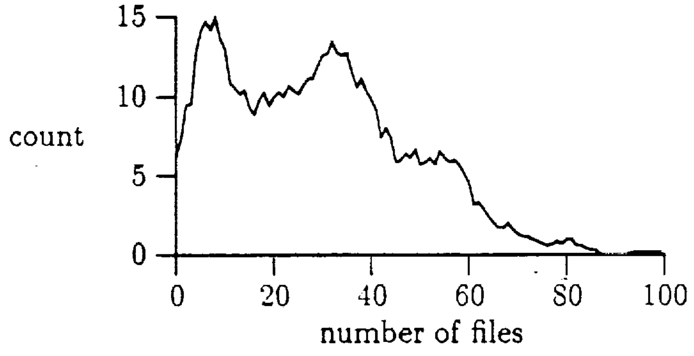


(b) After smoothing

Figure 5.4: Average file size (bytes).



(a) Before smoothing



(b) After smoothing

Figure 5.5: Average number of files referenced.

Table 5.3: Mean and standard deviation of access size (byte) and response time (microseconds) of file access system calls.

number of users using computer	access size mean(std)	response time mean(std)
1	946.71(956.76)	1284.83(4201.52)
2	936.06(945.16)	1716.26(7026.62)
3	932.80(946.87)	2120.99(13308.12)
4	956.12(965.49)	2447.55(16834.38)
5	947.98(948.53)	2960.32(16197.86)
6	928.66(935.09)	3494.30(30059.28)

5.2 Measuring the SUN NFS

To understand the effects of the different number of users and the different usage distributions, we use the file distribution and usage distribution in the previous section and change some usage distributions to determine the effects on response time.

To understand the effects of the different number of users with different think times, a series of experiments were performed, and the response times were measured. In these experiments, three types of users were simulated and they are determined by think time (Table 5.4). The computer was used by one user, two users, up to six users simultaneously. The results are shown in Figures 5.6–5.11. On these figures, each response time is the mean value during 50 login sessions. Figure 5.6 shows the response time under extremely heavy I/O users. From the curve, we can find that the response time has a linear relation to the number of users. This linear relationship is because all the users compete for resources all the time. Figures 5.7–5.11 show the response times from simulating different populations of users, with each population composed of a specified proportion

Table 5.4: Types of users simulated in experiments.

user type	think time
extremely heavy I/O	0
heavy I/O	5000
light I/O	20000

of heavy and light I/O users. We simulated populations with 0%, 20%, 50%, 80%, and 100% heavy I/O users. The slopes in these figures are not as large as that in Figure 5.6 because the competition for resources is not as heavy. One interesting observation is that the average response times in these figures are similar; that means a 5000-microsecond think time is not much different from a 20000-microsecond think time. This phenomenon may be due to the large standard deviation in response time (Table 5.3).

To know the effects of the different access sizes of file I/O system calls, a series of experiments were performed, and the response time was measured under different access sizes, from a mean of 128 bytes to 2048 bytes. The load in these experiment was an extremely heavy I/O user. The results are shown on Figure 5.12 and the response time is also the mean value during 50 login sessions. The results suggest that it is better to have large access sizes for file I/O system calls, which is why most language libraries want to keep a buffer for each file and manage it themselves. However, this mechanism requires extra space for these extra buffers.

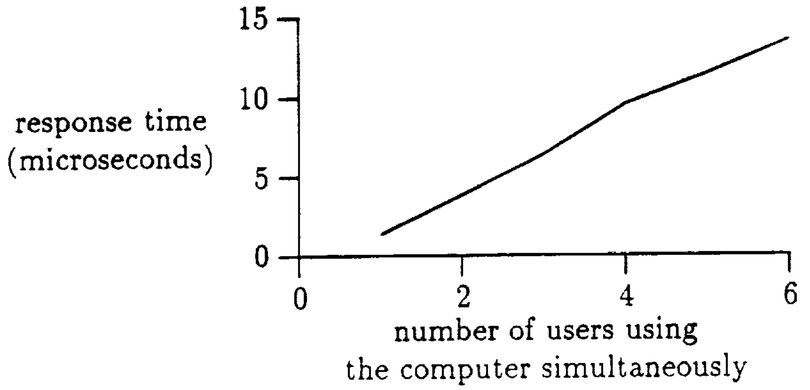


Figure 5.6: Average response time per byte under all extremely heavy I/O users.

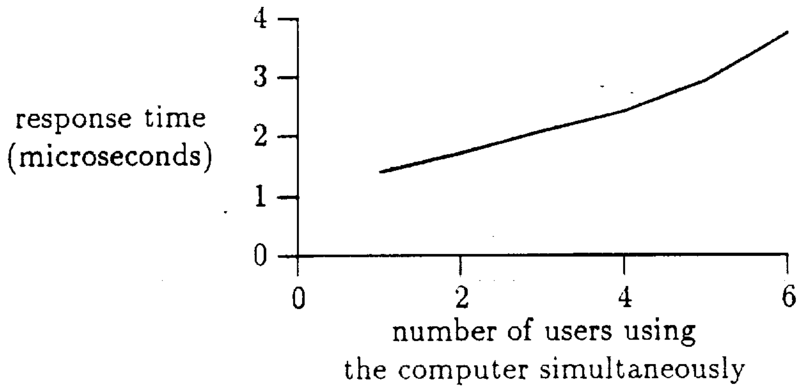


Figure 5.7: Average response time per byte under 100% heavy I/O users.

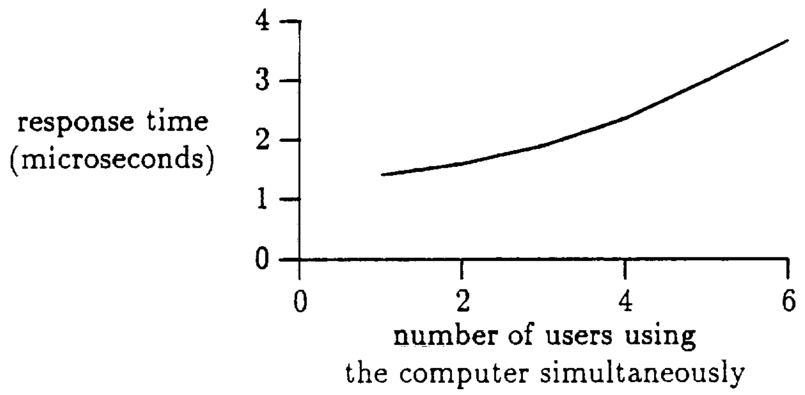


Figure 5.8: Average response time per byte under 80% heavy and 20% light I/O users.

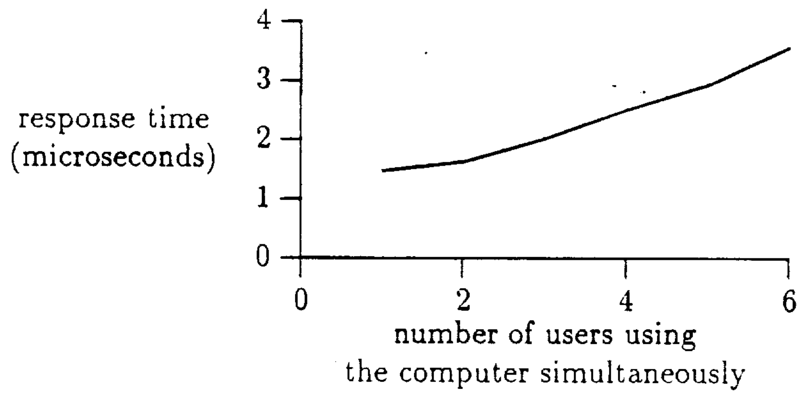


Figure 5.9: Average response time per byte under 50% heavy and 50% light I/O users.

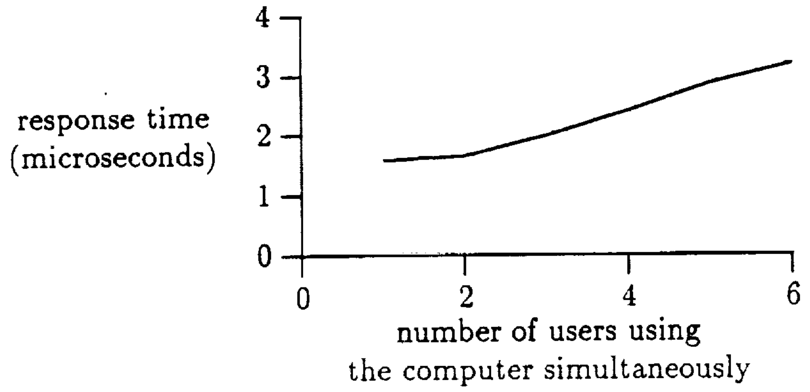


Figure 5.10: Average response time per byte under 20% heavy and 80% light I/O users.

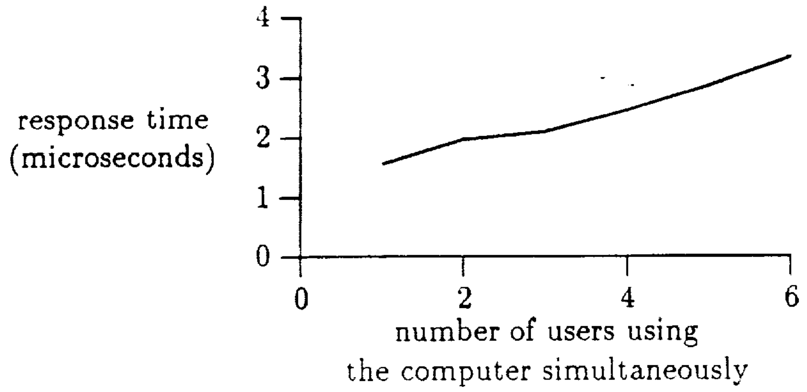


Figure 5.11: Average response time per byte under 100% light I/O users.

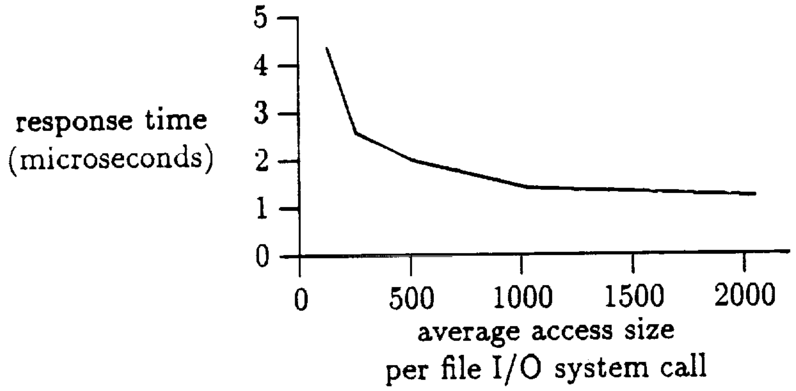


Figure 5.12: Average access time per byte under different access sizes of file I/O system calls.

5.3 Comparing Different File Systems

The previous section discussed measuring a file system. To compare two or more different file systems, we need to do a similar measurement for each file system and compare the results by different workload environments. One file system may be better under some particular environment, and others may be superior under different environments. To evaluate which file system is the best, we need to determine the environment first. For example, a laboratory may want to install a new computer system and faces several choices for the file system in addition to specifying the hardware and operating system. Then, they need to compare the performance of candidate file systems. The existing comparison and information about the performance of them may not be useful for the laboratory due to different workloads, such as number of users, type of users, etc. In this case, benchmarks are too artificial, and real data will be useless if they want to change the number of users in the laboratory. The Workload Generator is based on the existing file usage and it is user-oriented, so it is useful when the number of users changes but the usage of the computer system keeps the same; that is, the types of users keep the same. Therefore, this workload generator can be used to compare the file systems based on their particular file usage, and then the best choice can be made.

A procedure for using our simulation tool in a file system comparison study is as follows.

1. Measure the detailed usage distributions by modifying the kernel.
2. Execute the GDS with the distribution obtained from the previous step and generate all the required CDF tables.
3. Execute the FSC to build an artificial file system for the USIM.
4. Set up the computer system with one candidate file system, then execute the USIM as many copies as the number of users in the lab. Measure response times and average file I/O speed by a method similar to the one used in step 1.
5. Change the file system to another candidate, and keep the rest the same. Repeat the previous step to measure this file system. Repeat this step for all the rest of the candidate file systems.
6. Compare the results.

If there are several choices for hardware or operating systems, then all combinations should be considered and more experiments are needed.

5.4 Summary

The Workload Generator is designed for experiments and simulations related to file systems, such as measuring the performance of a particular file system or comparing several file systems. This chapter first mentioned about the usage of the Workload Generator, then used the Workload Generator to measure the performance of the SUN NFS, and then suggested a procedure to compare different file systems and demonstrates an example of application.

6. CONCLUSION

6.1 Summary

In this thesis, we developed a user-oriented synthetic workload generator based on a user-oriented model. The model was job-unspecific, represented file I/O operations at the system call level, allowed general distributions for the usage measures, and assumed independence in the I/O operation stream. The workload generator was logically partitioned into three parts which handle specification of distributions, creation of an initial file system, and selection and execution of file I/O operations.

6.2 Future Work

In Section 4.2, we discussed some problems with this type of workload generators. Some of these cannot be easily solved. Some possible improvements are as follows.

- The file types could include indexed files and direct-access files. To support simulation of accessing of such files, the location of data in a file should also be logged during measurement.

- To simulate time-varying user behavior, such as transitions between CPU-bound and I/O-bound phases, a Markov process model can be used. For example, from a previous study [CS85], we know that the distribution of inter-login times varies depending on time of day. A more realistic model would therefore allow for time-dependent distributions.

- Our assumption of independence in the file operation stream needs to be examined in greater detail.

- Considering the increasing popularity of window systems, the implications for user behavior should be considered. Under a window system, a user may have several simultaneous logins and may run several commands simultaneously (perhaps background jobs). Currently, our analysis associates a user with a single interactive session initiated by a login and terminated by a logout. The analysis must be extended to account for simultaneous login sessions initiated by a single user.

REFERENCES

[Bab81] O. Babaoglu. On construction synthetic programs for virtual memory environments. In D. Ferrari and M. Spadoni, editors, *Experimental Computer Performance Evaluation*, pages 195–204. North-Holland, 1981.

[Bar86] T. Barrington. A synthetic workload generator based on the user-oriented analysis of file usage. EE 441 project report, University of Illinois at Urbana-Champaign, 1986.

[Buc69] W. Buchholz. A synthetic job for measuring system performance. *IBM Syst. J.*, 8(4):309–318, 1969.

[CS85] M. Calzarossa and G. Serazzi. A characterization of the variation in time of workload arrival patterns. *IEEE Tran. on Computers*, C-34(2):156–162, February 1985.

[Dev88] M. V. Devarakonda. *File Usage Analysis and Resource Usage Prediction: A Measurement-Based Study.* Ph.D. thesis, University of Illinois at Urbana-Champaign, 1988.

[DI86] M. V. Devarakonda and R. K. Iyer. A user-oriented analysis of file usage in UNIX. In *Proc. COMSAC-86*, pages 21–27, 1986.

[Dom82] B. Domanski. Load driving a system. *Computer Performance*, 3(4):195–200, 1982.

[HDG71] E. Hunt, G. Diehr, and D. Garnatz. Who are the users? An analysis of computer use in a university computer center. In *AFIPS Conf. Proc. SJCC*, volume 38, pages 231–238, 1971.

[HKM+88] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and performance in a distributed file system. *ACM Tran. Computer Systems*, 6(1):51–81, February 1988.

[Hug84] H. D. Hughes. Generating a drive workload from clustered data. *Computer Performance*, 5(1):31–37, 1984.

[ODCH+85] J. Ousterhout, D. Da Costa, D. Harrison, J. A. Kunze, M. Kupfer, and J. G. Thompson. A trace-driven analysis of the UNIX 4.2 bsd file system. In *Proc. of the 10th ACM Symposium on Operating System Principles*, pages 15–24, 1985.

[Ser86] G. Serazzi. *Workload characterization of computer systems and computer networks.* North-Holland, 1986.

[SK74] K Sreenivasan and A. J. Kleinman. On the construction of a representative synthetic workload. *CACM*, 17(3):127–133, March 1974.

[WF71] D. C. Wood and E. H. Forman. Throughput measurement using a synthetic job stream. In *AFIPS Conf. Proc. FJCC*, volume 39, pages 51–56, 1971.