Notes and Reports
in
Computer Science and Applied Mathematics

Computer Performance Modeling Handbook

Edited by
STEPHEN S. LAVENBERG

COMPUTER PERFORMANCE MODELING HANDBOOK

Edited by

STEPHEN S. LAVENBERG

IBM Thomas J. Watson Research Center
Yorktown Heights, New York

ACADEMIC PRESS 1983

A Subsidiary of Harcourt Brace Jovanovich, Publishers

New York London
Paris San Diego San Francisco São Paulo Sydney Tokyo Toronto

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1 7DX

Library of Congress Cataloging in Publication Data
Main entry under title:

Computer performance modeling handbook.

(Notes and reports in computer science and applied mathematics ; no. 4)
Includes bibliographical references and index.
1. Electronic digital computers--Evaluation.
2. Digital computer simulation. I. Lavenberg, Stephen S.
II. Series.
QA76.9.E94C66 1983 001.64'028'7 82-8840
ISBN 0-12-438720-9 AACR2

PRINTED IN THE UNITED STATES OF AMERICA

83 84 85 86 9 8 7 6 5 4 3 2 1

Preface

Computer systems have become so complex that intuition alone is not sufficient to predict their performance, and mathematical modeling has come to play an important role. Mathematical models of system performance range from relatively simple ones, whose solution can be obtained analytically, to very complex ones that must be simulated. This *Computer Performance Modeling Handbook* is a state-of-the-art reference manual for computer performance modeling practitioners. It is intended to help practitioners formulate and apply performance models by providing (1) analytical results for a wide variety of performance models, (2) guidance in the simulation of performance models, and (3) numerous application examples. It differs from a textbook in that mathematical derivations are not presented; rather, the emphasis is on easy-to-apply analytical results and simulation procedures. Anyone working in the area of computer performance evaluation should find the handbook indispensable. In addition, it is a useful source of material for performance evaluation courses at the senior or graduate level.

Chapter 1 (''Introduction to Performance Modeling'' by Stephen S. Lavenberg) provides an introduction to the analysis, simulation, and validation of computer performance models. Chapter 2 (''Mathematical Prerequisites'' by Stephen S. Lavenberg) contains basic material on probability, random variables, and the Poisson process to aid a modeler in understanding the assumptions made about a model and the results obtained from the analysis or simulation of the model. Little's formula, an important formula relating performance measures, is also discussed.

The models that play the most important role in performance evaluation are queueing models, in particular, queueing networks. Chapter 3 (''Analytical Results for Queueing Models'' by Stephen S. Lavenberg and Charles H. Sauer) contains the most extensive collection of analytical results for queueing models currently available in the literature. The results are either formulas for performance measures or, as is typically the case for queueing networks, algorithms that can be used to compute performance measures. Over thirty examples are presented. Chapter 4 (''Approximate Analysis of Queueing Networks'' by Stephen S. Lavenberg and Charles H. Sauer) presents approximate analysis methods for queueing networks for which either exact analytical results are not available or, if they are available, the computational expense is prohibitive. This is an area of continuing research activity. However, several methods that have practical uses are presented.

Chapters 5 and 6 deal with statistical aspects of simulation. Chapter 5 (''Generation Methods for Discrete Event Simulation'' by Gerald S. Shedler) describes methods for generation of the random inputs that drive a simulation. General methods for random number generation are discussed and specific algorithms are provided for generation of samples from a variety of distributions. Chapter 6 (''The Statistical Analysis of Simulation Results'' by Peter D. Welch) discusses the statistical analysis of the random outputs produced by a simulation. The purpose of the analysis is to produce an estimate of a performance measure and a meaningful statement about the accuracy of the estimate. The estimation of both transient and steady-state performance measures is considered.

Chapter 7 (''Simulator Design and Programming'' by Harry M. Markowitz) discusses aspects of the designing, coding, and debugging of simulation programs. Simulation programming is illustrated first using SIMSCRIPT, a language specifically designed for simulation programming, and then a general-purpose programming language such as Fortran or PL/I.

Chapter 8 (''Extended Queueing Network Models'' by Charles H. Sauer and Edward A. MacNair) describes a set of powerful modeling elements that can be used to define queueing networks that represent complex system features. The resulting class of queueing networks is far broader than the class of networks considered in Chapters 3 and 4, so typically these networks must be simulated. A series of examples is provided illustrating the usefulness of these networks.

Each chapter is largely self-contained. However, Chapters 1 and 2 contain basic material that is useful for understanding the remaining chapters, and the material on queueing networks in Chapter 3 is useful for understanding Chapter 4.

All of the authors are members of the IBM Research Division, and all except one are at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. Gerald Shedler is at the IBM Research Laboratory in San Jose, California. I am grateful to the Research Division for the time and facilities provided for the preparation of the handbook. In addition, financial support was provided by a grant from the IBM Group Technical Assignment System Structure Technology program. I wish to thank Hisashi Kobayashi for his participation in the conception of the handbook. The inspiration for the handbook was provided by the success of the IBM manual, *Analysis of Some Queueing Models in Real-Time*

Systems, Second Edition, GF20-0007-1, written by Philip H. Seaman and published in 1971. That manual is no longer up-to-date. In particular, it does not cover important results for queueing networks, and it does not contain any material on simulation.

I wish to thank Marylou Dietrich for her careful typing of major portions of the manuscript and for her cheerful cooperation with several authors and their many revisions. The careful typing and other assistance given by Betty A. Smalley during the early stages of preparation of the handbook is also gratefully acknowledged.

Contents

Preface xi

1 Introduction to Performance Modeling

Stephen S. Lavenberg

1.1 The Role of Performance Modeling 1
1.2 Analysis of Performance Models 2
1.3 Simulation of Performance Models 6
1.4 Validation of Performance Models 9
References 10

2 Mathematical Prerequisites

Stephen S. Lavenberg

2.1 Probability and Random Variables 13
2.2 Poisson Process 45
2.3 Little's Formula and Its Applications 49
References 53

3 Analytical Results for Queueing Models

Stephen S. Lavenberg and Charles H. Sauer

3.1 Introduction 56
3.2 Performance Measures 57
3.3 Single Service Center Queueing Models 59
3.4 Introduction to Queueing Networks 105
3.5 Closed Product Form Queueing Networks 119
3.6 Open Product Form Queueing Networks 151
3.7 Mixed Product Form Queueing Networks 159
3.8 Further Results for Product Form Queueing Networks 164
3.9 Sources for Equations 166
References 170

4 Approximate Analysis of Queueing Networks

Stephen S. Lavenberg and Charles H. Sauer

4.1 Introduction 173
4.2 Mean Value Analysis Methods 175
4.3 Decomposition Methods for Networks without Product Form 187
4.4 Extended Service Time Method for Open Queueing Network Model of Disk Systems 208
4.5 Other Methods for Open Networks 217
References 220

5 Generation Methods for Discrete Event Simulation

Gerald S. Shedler

5.1 Uniform Random Numbers 223
5.2 Nonuniform Random Numbers 229
5.3 Discussion 247
5.4 Sample Paths for Stochastic Processes 252
References 264

6 The Statistical Analysis of Simulation Results

Peter D. Welch

6.1 The Random Nature of Simulation Outputs 268
6.2 Estimating Transient Characteristics 280
6.3 Estimating Steady-State Characteristics 288
6.4 Achieving a Desired Accuracy: The Use of Pilot Experiments and Sequential Procedures 320
6.5 Broader Statistical Questions 326
References 328

7 Simulator Design and Programming

Harry M. Markowitz

7.1 Introduction 331
7.2 Simulation Programming 333
7.3 Simulation Programming with a General Purpose Language 348
7.4 Debugging 358
7.5 Selection of a Type 2 Language 360
7.6 Simulator Design 360
7.7 Summary 364
References 364

8 Extended Queueing Network Models

Charles H. Sauer and Edward A. MacNair

8.1 Introduction 367
8.2 Extended Queueing Network Elements 369
8.3 Examples 375
8.4 The Branching Erlang Distribution 390
References 393

Index 395

1 Introduction to Performance Modeling

Stephen S. Lavenberg

1.1 The Role of Performance Modeling 1
1.2 Analysis of Performance Models 2
1.3 Simulation of Performance Models 6
1.4 Validation of Performance Models 9
References 10

1.1 The Role of Performance Modeling

This handbook deals with the use of models in computer performance evaluation and is intended to be a state-of-the-art reference manual for performance modeling practitioners. While the evaluation of a computer system may involve such factors as cost, availability, reliability, serviceability, and security, we shall be concerned only with computer system performance, where performance is measured by such quantities as throughput, utilization, and response time. Performance is one of the key factors that must be taken into account in the design, development, configuration, and tuning of a computer system. Of course the overall system performance is affected by the performance characteristics of the various subsystems and ultimately the individual hardware and software components that constitute the system. Hence, the design of new hardware and software components has system performance implications that should not be ignored.

Once a particular computer system has been built and is running, the performance of the system can be evaluated via measurements, using hardware and/or software monitors, either in a user environment or under controlled

ISBN 0-12-438720-9

benchmark conditions. However, in order to evaluate the performance of a system, subsystem, or component that cannot be measured, for example, during the design and development phases, it is necessary either to make performance predictions based on educated guesses or to use models as an aid in making performance predictions. The interactions in present-day computer systems are so complex that some form of modeling is necessary in order to be able to predict and understand computer system performance.

Performance modeling is widely used, not only during design and development, but also for configuration, tuning, and capacity planning purposes. One of the principal benefits of performance modeling, in addition to the quantitative predictions obtained, is the insight into the structure and behavior of the system that is obtained in the course of developing a model. This is particularly true during system design. The modeler must abstract the essential features of the design and this process leads to increased understanding and may result in the early discovery and correction of design flaws.

Many performance models can be mathematically analyzed to obtain such performance measures as utilizations, throughputs, and average response times. Such models are the subject of Chapters 3 and 4 of the handbook. Other performance models are so complex that they must be simulated in order to estimate performance measures. The simulation of performance models is the subject of Chapters 5–8. It is assumed that the reader has a basic familiarity with the complexities of present-day computer systems and a strong interest in their performance. Mathematical prerequisites are covered in Chapter 2. The remainder of this chapter contains introductory sections on the analysis and simulation of performance models as well as a discussion of model validation.

1.2 Analysis of Performance Models

A computer system can be viewed as a collection of interconnected hardware and software resources that provides service to a community of users. Figure 1.1 shows in a simplified way some of the resources that may comprise a computer system and the flow of work through the system. The work to be performed consists of transaction processing for interactive users at terminals and processing of a batch job stream. The hardware resources shown are terminals, main memory, processors, channels, and I/O devices. The software resources shown are the terminal access method, the batch job entry system, the scheduler (which swaps jobs and transactions in and out of main memory), the dispatcher (which schedules work on the processors), and the I/O supervisor (which schedules I/O requests). Important functions of the software are (1) to allow the hardware to be used in an efficient manner, e.g., by allowing as much sharing of the hardware among the users as possible and (2) to schedule work on the hardware to provide quick response to users, particularly to interactive users. In order to achieve efficient usage of the hardware, several jobs (we use the term jobs to refer to transactions or batch jobs) can simultaneously contend for such resources as main memory, processors, and the I/O subsystem. The figure shows the queues that form for these resources as a result of contention. The time spent waiting in these queues can have a

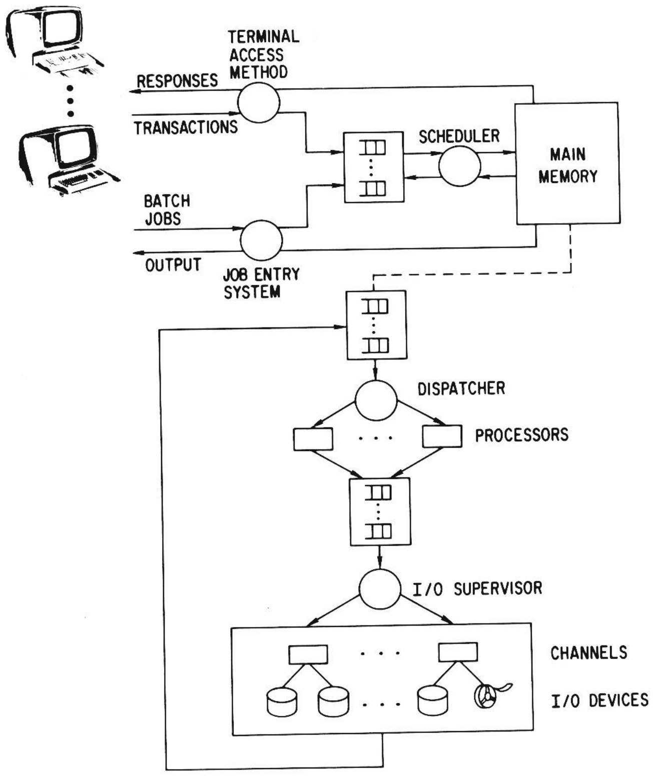

Fig. 1.1 Computer system resources and work flow.

substantial impact on the performance of the system. Perhaps the major reason models are needed in performance evaluation is to predict the effect of contention for resources on performance. The kinds of performance models that play the most important role in this regard are queueing models. The resource requirements of jobs are specified as inputs to a queueing model, and the model explicitly represents the queueing delays that occur. Examples of resource requirements of jobs are the number of instructions to be executed on a processor, the number of bits to be transmitted over a communication line, the amount of main memory required to store a program, the number of instructions executed while holding a serially reusable piece of software, and the length of a record to be read from a disk drive.

Queueing models began to be analyzed around the turn of the century in connection with performance problems in telephone systems. They have since given rise to a mathematical discipline known as queueing theory, a topic in applied probability. Queueing theory employs a variety of mathematical techniques to analyze queueing models. Many queueing models that represent the contention for a single resource have been analyzed. Chapter 3 presents analytical results for single resource queueing models that are useful in predicting the performance of individual components, such as communication lines, processors, and I/O devices.

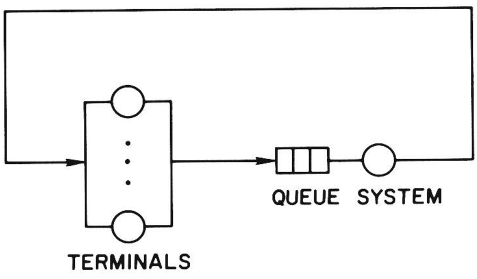

Fig. 1.2 Queueing model with system represented as a single resource.

One of the earliest successful applications of queueing models to problems in computer performance occurred in the late 1960s during the development of IBM's OS/360 Time Sharing Option (TSO). A simple queueing model was used to predict the performance of TSO running a single partition in main memory (see Lassettre and Scherr, 1972). This model is shown in Fig. 1.2 and described in detail in Chapter 3. It is assumed in the model that there are a fixed number of interactive users at terminals. A user thinks, enters input at the terminal, waits for the computer to provide service in response to the input (during which time the keyboard is locked), and receives output at the terminal. This process continues indefinitely. The time it takes to receive output, to think, and to enter input is specified as an input to the model and assumed to be a random variable having a known distribution. The service provided by the system consists of swapping a user's program in and out of main memory and executing the program. The time it takes to perform this service is also specified as an input to the model and assumed to be a random variable having a known distribution. Since there is only a single partition, service for one user cannot go on in parallel with service for another user. Thus, the system is represented as a single resource which can serve at most one user at any one time, and a queue is shown in front of this resource. If there were more than one partition so that service for different users could go on in parallel, a different representation of the system would be required. The model was primarily used to determine the number of interactive users the system could support without exceeding a specified average response time.

In a multiprogramming system several user programs simultaneously reside in main memory so that the processing activity of one program can go on in parallel with the I/O activity of other programs. Furthermore, contention can occur for the processing and I/O resources. What is needed in such a situation is a model that explicitly represents the contention for the multiple resources that comprise a system. Starting in the late 1950s such queueing models, which were developed to represent job shop systems, were analyzed in the operations research literature. Analytical results were obtained for a fairly broad class of models. Independent of this work, multiple resource queueing models were used to predict computer performance as early as the mid-1960s. However, it was not until the 1970s, when the results in the operations research literature began to be applied, that the use of multiple resource queueing models to predict computer performance became widespread. One such

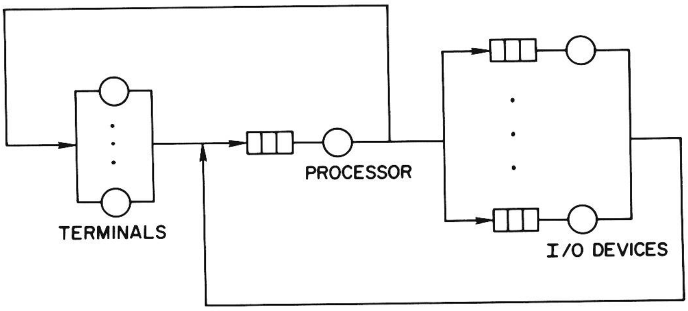

Fig. 1.3 Queueing model with system represented as a network of resources.

multiple resource queueing model is shown in Fig. 1.3. In comparison with Fig. 1.2 the system is no longer represented as a single resource, but as a network of resources, each resource having its own queue of requests. The application of queueing networks to computer performance modeling has led to further progress in developing and analyzing such models. For example, queueing network models that explicitly represent different types of jobs, e.g., batch jobs and TSO transactions, were analyzed. Chapter 3 presents analytical results for a broad class of queueing network models and examples of their application. These models are now widely used in performance evaluation. Several program packages, some of which are commercially available, have been developed to allow users to define and solve queueing network models of computer performance. For a survey of such program packages see Sauer and MacNair (1979).

Although the class of queueing network models that have been exactly analyzed has grown in recent years, many system features cannot be explicitly represented in such models. One important feature that is not represented in the queueing network in Fig. 1.3 is contention for main memory space. The model in Fig. 1.4 differs from that in Fig. 1.3 in that a queue for memory is

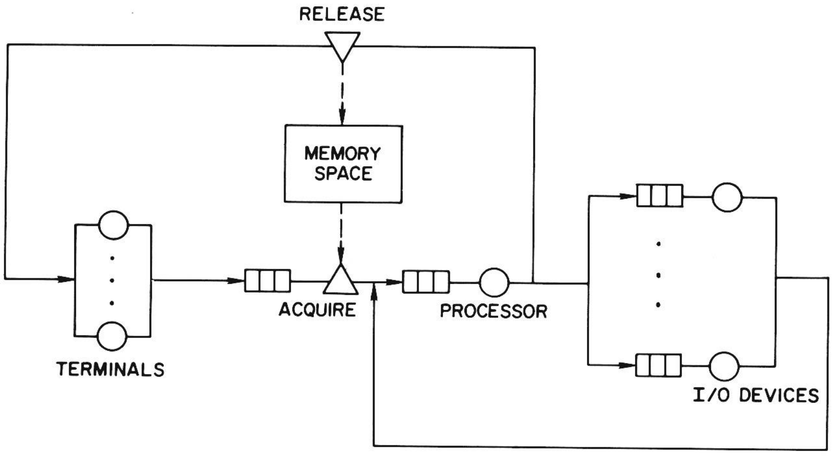

Fig. 1.4 Queueing model with contention for memory space represented.

shown. A job cannot get access to the processing and I/O resources of the system until it has acquired a portion of memory. When the job completes its processing and I/O services, it releases its portion of memory. Queueing networks in which jobs must possess a limited resource (such as memory space) in order to get service from another resource (such as a processor) have not been exactly analyzed. Chapter 4 presents an important technique, based on decomposing a model into a hierarchy of submodels, which can be used to approximately analyze such networks. Other useful approximate analysis techniques, e.g., for networks that explicitly represent processor dispatching based on priorities, are discussed in Chapter 4.

1.3 Simulation of Performance Models

Many performance models are so complex that an exact mathematical analysis is not possible and no reasonable approximate analysis techniques exist. For example, this is usually the case with queueing network models that explicitly represent complex resource scheduling algorithms. In such cases it is necessary to simulate the model. The kind of simulation we refer to is discrete event simulation on a digital computer. Once a model has been formulated a simulation program is written that keeps track of the evolution in time of the model as determined by the occurrence of events at discrete time instants. An event might be the arrival of a transaction or the completion of a service. The simulation program is run in order to obtain estimates of performance measures. Since most performance models incorporate some form of randomness in their inputs, e.g., to determine the times at which jobs arrive or to determine the resource requirements of a job, the outputs produced by such models, e.g., a resource utilization or an average response time, are also random quantities. Thus, the simulation of a model that incorporates randomness has statistical aspects that should not be ignored. Statistical aspects of simulation are discussed in Chapters 5 and 6.

Chapter 5 describes methods for generation of the random inputs that drive the simulation. Typically, the inputs to a model are independent samples from specified distributions. Although simulation programming languages provide some facilities for sampling from common distributions, the methods used do not always represent the state-of-the-art in terms of computational efficiency or appearance of randomness. Chapter 5 provides specific algorithms for generation of independent samples from a number of standard distributions, as well as a discussion of general methods for random number generation and considerations in the use of random number streams. This material will help the practitioner who has to program a procedure to generate samples from a distribution. In addition, Chapter 5 gives some methods for generating event times (e.g., times at which jobs arrive) as inputs to a model when the times between events are not independent samples from the same distribution.

Chapter 6 discusses the statistical analysis of the random outputs produced by the simulation. This is an important aspect of simulation that, if ignored, can result in erroneous conclusions being drawn from a model. The basic problem is that an unknown deterministic quantity in the model is estimated by a random quantity generated by the simulation. The purpose of the statistical analysis is to produce a meaningful statement about the accuracy of the estimate. Both the estimation of transient performance measures and the estimation of steady-state performance measures are considered.

In a trace-driven simulation the inputs that drive the simulation, e.g., the arrival times of jobs and their sequences of resources requirements, are not random quantities obtained by sampling from distributions, but are deterministic quantities usually obtained from tracing the jobs on a running system. Unless randomness is introduced elsewhere in the model a trace-driven simulation will be purely deterministic. Trace-driven simulation is discussed in review articles by Sherman and Browne (1973) and by Sherman (1976).

The design, coding, and debugging of a simulation program is often a very time consuming task. Simulation has the advantage that it can be applied to very detailed performance models, but this is also one of its major pitfalls. Some simulation projects have failed because the model was so detailed that the programming was not completed in a timely manner or because the model required more detailed input data than was available. Thus, it is often good practice to start with a fairly simple model and add detail if the simple model proves to be inadequate. Although it is difficult to say in general how detailed a performance model should be, the level of detail should be chosen based upon the purpose for which the model is intended, the information available about the system design, the kind of input data available, and the time and resources available to code, debug, and run the simulation program. Chapter 7 discusses aspects of the design, coding, and debugging of simulation programs. There are many languages available for simulation programming, and Chapter 7 does not attempt to cover all of these. Instead it illustrates simulation programming, first using SIMSCRIPT, a language specifically designed for simulation programming, and then using a general purpose programming language such as Fortran or PL/I.

GPSS is a widely used simulation language specifically designed to describe the flow of jobs through a system. A model to be simulated using GPSS is described by a block diagram. GPSS is discussed in the books by Fishman and Kobayashi mentioned in the References. The book by Gordon (1975) can serve as a GPSS user's manual. The ease of constructing models using GPSS has led to its wide use. However, the user should be aware that GPSS is considered to have shortcomings with regard to random number generation (e.g., see Section 4.4.9 in the book by Fishman).

As previously discussed, queueing networks can be used to explicitly represent the contention for resources that occurs in computer systems. Chapter 8 describes a set of powerful modeling elements that can be used to define queueing network models that represent many complex system features. The elements have been developed to provide compact representations of system features and to be flexible so that a variety of features can be represented. Furthermore, the elements have pictorial representations so that it is easy to draw a diagram of the network. The class of queueing networks that can be defined by using these elements is far broader than the class of queueing networks that can be mathematically analyzed. While it is possible to simulate these networks using any of the simulation languages mentioned previously, a program package called the RESearch Queueing package (RESQ) has been developed at IBM Research and is specifically designed so that users can easily define these kinds of networks. RESQ builds a simulation program based on the model definition, then carries out the simulation of the model and produces a statistical analysis of the simulation output. The RESQ package also incorporates a program that computes performance measures for those