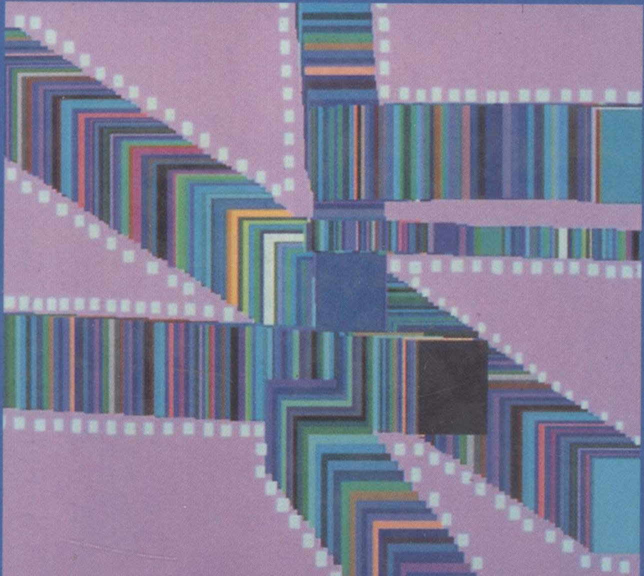
INTERNATIONAL
COMPUTER SCIENCE
SERIES

Performance Measurement of Computer Systems

Phillip McKerrow



Performance Measurement of Computer Systems

Phillip McKerrow
University of Wollongong,

ADDISON-WESLEY
PUBLISHING
COMPANY

Sydney · Wokingham, England · Reading, Massachusetts
Menlo Park, California · New York · Don Mills, Ontario
Amsterdam · Bonn · Singapore · Tokyo · Madrid
Bogota · Santiago · San Juan

Cover design by John Gibbs.
Cover graphic by Laurence M. Gartel.
Printed in Great Britain by T.J. Press (Padstow) Ltd, Cornwall.

First Printed in 1987.

British Library Cataloguing in Publication Data

McKerrow, Phillip
Performance measurement of computer systems.
—(International computer science series).
1. Electronic data processing—Evaluation
I. Title II. Series
004 QA76.9.E94

ISBN 0–201–17436–7

Library of Congress Cataloguing in Publication Data

McKerrow, Phillip, 1949–
Performance measurement of computer systems.

(International computer science series)
Bibliography; p.
Includes index.
1. Electronic digital computers—Evaluation
I. Title.
QA76.9.E94M39 1988 004.2′4 87-19363
ISBN 0–201–17436–7

Performance Measurement of Computer Systems

In memory of Ian Paul and John Mark,
born 31 August 1981, died 1 September 1981

Preface

In this book, I have attempted to combine the results of the last three decades of research in measuring the performance of computer systems into a unified body of knowledge: theory and practice. Unification is based on a formulation of performance measurement.

We can model the code of a computer system with an abstract mathematical object. When this code is executed it becomes an executable object, which we can measure. To ascertain the extent of this executable object is the task of performance measurement. An object (computer system, task, program, procedure) is defined recursively in terms of lower level objects. In the theoretical section of this work, I have defined a set of measures for an executing object. These measures apply at every level of the object hierarchy, have been expressed in mathematical equations, and define a formulation of performance measurement. The measured data can be displayed in graphical form, making evaluation easier.

This formulation provides a general, overall context within which measurement and evaluation can take place. The purpose of measurement is not to collect numbers, but to gain insight into the actions of the object under study. By recording appropriate stimulus information, and by using graphical techniques to analyse the data, we can understand the actions of the object.

The formulation has been validated in a number of ways:

- measurement experiments have been conducted,
- measures proposed by the formulation have been compared to current measurement practice,
- other formulations have been compared to it, and
- corollaries have been hypothesized and tested.

From the results of these validation procedures, I have confirmed a high degree of correlation between the formulation and current practice. On the basis of the formulation, we have designed a hybrid performance analyser, which we have used in performance evaluation, in system optimization, in program execution monitoring, when debugging software, and for finding software related hardware faults. A number of future research areas, which flow out of the formulation, are proposed.

This book commences with an introduction to the field of performance measurement and an overview of various aspects of it. Then, the formulation of performance measurement is described in detail. Other formulations proposed by researchers in the performance evaluation field are discussed, and the underlying conceptual models of program execution are compared. Following this, measurement tools and techniques are reviewed. Next comes the design of a hybrid performance analyser, which is built around a logic state analyser, and is based on a philosophy of hybridization derived from the formulation of performance measurement. Finally, the design of computer systems for performance measurement is discussed. In the last two chapters, the formulation is extended to cover parallel processors, and measurement in a number of other applications.

Case studies are included to illustrate performance measurement methods and software debugging techniques. I have used these case studies to demonstrate the practicality and power of a performance measurement methodology based on the formulation of performance measurement.

Acknowledgements

I wish to thank the following for permission to reproduce material from published sources:

AFIPS Press and T.E. Bell, B.W. Boehm and R.A. Watson (1972) for Figures 3 and 4 from 'Framework and initial phases for computer performance improvement', *FJCC Proceedings,* No. 41, 1141-1154.

Domenico Ferrari for a figure from *Computer Systems Performance Evaluation* © 1978, p. 14. Reprinted by permission of Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Hewlett Packard for a photograph of the HP 1610A logic state analyser and a block diagram; also for photographs of the HP 1630 taken from their brochure no. 5953-9208.

The Institute of Electrical and Electronics Engineers, Inc. for various figures from the following: English, W.R., Engelbart, D.C. and Berman, M.L. (1967) 'Display-Selection Techniques for Text Manipulation', *IEEE Transactions on Human Factors in Electronics,* Vol. 8, No. 1, 5-20; Fromm, H., Hercksen, U., Herzog, U., John, K.H., Klar, R. and Kleinder, W. (1983) 'Experiences with Performance Measurement and Modeling of a Processor Array', *IEEE Transactions on Computers,* Vol. C32, No. 1, January, 15-31; Gehringer, E. F., Jones, A.K. and Segall, Z.Z. (1982) 'The CM* Testbed', *IEEE Computer,* Vol. 15, No. 10, October, 40-53; Segall, Z. *et al.* (1983) 'An Integrated Instrumentation Environment for Multiprocessors', *IEEE Transactions on Computers,* Vol. C32, No. 1, January, 4-14.

Northwest Instrument Systems, Inc. for a photograph of their μANALYST$_R$ connected to an Apple IIe.

Tektronix, Inc. for a photograph of a DAS 9129 logic analyser taken from their brochure no. 57W-5025.

K. Terplan for a table which appeared in 'Network Monitor Survey', *Computer Performance,* Vol. 2, No. 4, pp. 58-173 (Butterworth & Co. (Publishers) Ltd).

I also wish to thank Professor Juris Reinfelds, Professor Geoff Dromey, Richard Miller, Gary Stafford, and Michael Milway for their encouragement and direction during this research. Their comments forced me to clarify my ideas, prompting further avenues of thought and deeper insight.

The logic state analyser used in this research was bought with a grant from the Department of Science and Technology, Australian Research Grants Committee. The Apple microcomputer and other facilities were provided by the Department of Computing Science, University of Wollongong.

During the course of this research I have regularly prayed about problems and meditated upon insights. I have always found God to be one step ahead of me, and ready to give insight and understanding.

Finally I would like to thank my wife and family for the hours of my time they have given up so that I could write this tome. No acknowledgement is complete without heartfelt thanks to the typist, Mrs Lynn Maxwell, who can type faster than I can think, and to Mr John Murray from whose work the line illustrations have been prepared.

This book has been typeset on a Compugraphic phototypesetter using the troff word processor, at the University of Wollongong.

Contents

Chapter 1
Introduction

1.1 Performance measurement

Techniques used to evaluate the performance of computer systems can be grouped into four overlapping areas: measurement of system parameters, evaluation of collected data, modelling of system behaviour, and modifications to improve performance. In this work, I concentrate on the measurement of system parameters. Measurement is discussed in the context of the whole field when other areas of performance evaluation determine and constrain the parameters to be measured.

In the early days of computing, a programmer's main goal was to get a working program with little thought about its efficiency; however, there were some exceptions. Von Neumann (1946) compared the speed with which a number of early computers, including ENIAC, performed multiplications when computing ballistic trajectories. Herbst *et al.* (1955) measured the instruction mix of programs running on the Maniac computer.

In the early sixties, performance measurement was commenced in earnest. As computers became readily available, users sought ways to increase the productivity of both the computer and the programmer and hence to reduce the cost of computing. Computer throughput was increased by using operating systems to handle resource sharing: initially, simple batch systems; more recently, time sharing and multiprogramming. Program development time has been shortened through the use of high-level languages, structured programming, and other software engineering techniques.

Concurrent with these developments, and spurred on by the high cost of computing, has been a desire to evaluate how well systems are performing, and to find ways of improving that performance. During the sixties, performance measurement studies were carried out on many installations. By 1967, the field had grown to the point where Calingaert (1967) was able to publish a survey of the then common techniques, and a few years later Miller (1972) published a bibliography of over 250 papers. The early seventies saw a burst of measurement activity, which diminished to a mere trickle of papers by the mid-seventies as researchers turned to modelling techniques.

Measurement is a fundamental technique in any science (Curtis, 1980). The fact that little work has been reported on the measurement of computer systems in the last few years has been seen by some as an indication that all the work has been done. This is not true - computer performance measurement remains a collection of techniques with no unified body of knowledge. Research effort dwindled, not because all the problems were solved, but because of a number of other factors:

- Measurement ideas were several years ahead of the available technology. It is interesting to read papers from the heyday of measurement, and see the gradual transition from what we have done, to what we are doing, to what we think we might be able to do when we finish developing the tool. Consequently, most of the ideas are not new, but the technology of the early seventies was not cheap enough for the development of powerful, general-purpose tools.
- The complexity of computer systems increased rapidly, making measurement more difficult.
- Researchers were attracted by the mathematical tractability of modelling techniques, particularly analytical queuing models. Modelling provided a rich source of research ideas at a time when measurement was being frustrated by the increasing complexity of computer systems. The lack of tools powerful enough to handle this complexity made measurement too hard.
- The literature of the time consisted of descriptions of measurement techniques and their results. No unified body of knowledge had been established and no theoretical basis for measurement had been developed. Hence, there was no framework within which to tackle the measurement problems posed by the new, more complex systems.

During the last decade, advances in technology have made computing power so cheap that all new test instruments include microprocessors. One new instrument, the logic state analyser, is more powerful than any of the hardware measurement tools of a decade ago. As a result of these advances, technology is no longer a limitation in measurement. The growing use of microcomputers increases dramatically the need for effective performance measurement tools. However, the design of these tools must be grounded in a unified formulation of measurement if lasting results are to be achieved. Such a formulation is developed in the next chapter. In subsequent chapters, current measurement techniques are evaluated in the light of this formulation, and some of the implications of the formulation for future measurement techniques and tools are investigated. The result is a unified body of performance measurement knowledge.

1.2 Measurement categories

To develop a unified formulation of measurement we must gather all the independent measurement categories together under one umbrella. Then common principles can be extracted. The differences between measurement situations are differences in the application of theory and tools, not conceptual differences in either theory or tools. In the following paragraphs, the major applications of performance measurement are briefly discussed. As many of these areas overlap, the discussion is aimed at showing the breadth of performance measurement.

Human engineering is the design of computer systems for use by people. It includes measuring the interactions between the user and the system. Users influence the performance of a system by producing inputs: requests for program execution, data, system commands, new programs, etc. The response of the system to these inputs is important, particularly on an interactive system (Figure 1.1) where the user expects fast response to commands which are input at highly irregular intervals. If the response is too slow, the user gets frustrated and will use another system. If a terminal is poorly designed, people may refuse to use it. Ease of use can be partially evaluated by measuring human and system response times (Figure 1.2).

Lack of feedback to the user may result in the user executing additional commands to check if the previous command worked, significantly increasing the workload. The classic example of this occurred in 1963 when a major American airline was trying

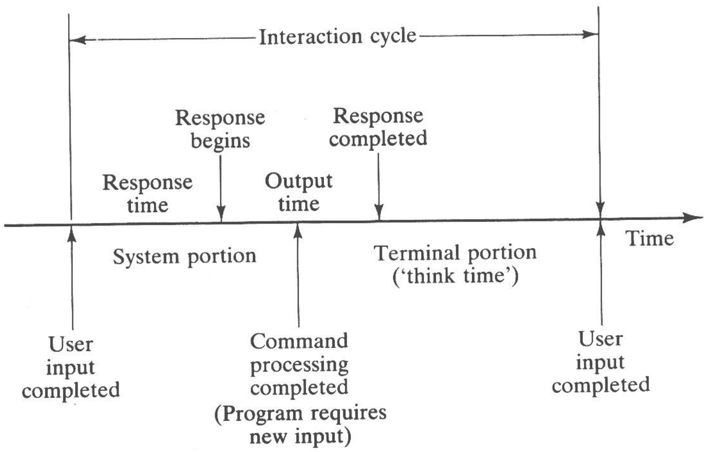


Figure 1.1 User-computer interaction cycle (Ferrari, 1978).

unsuccessfully to go on line nationwide with its new computerized reservation system (Warner, 1974). Everything went well until they tried to bring the last and busiest region, New York City, on line. The system crashed, hopelessly overloaded. When the system was measured, they found that before New York was brought on line the existing load was 90%, not 40% as predicted by simulation. Each operator, after keying in a reservation, would immediately enquire to see if the system had the data. The solution: the ball on the typewriter was wiggled to let the operator know the data was in.

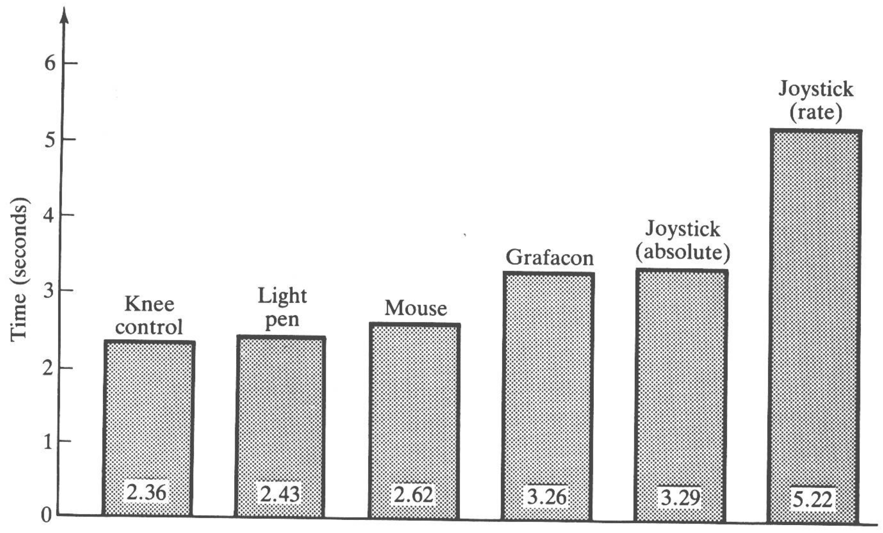


Figure 1.2 Comparison of average times taken by inexperienced users to locate a cursor at a character using a variety of graphics input devices (English *et al.*, 1967; copyright © 1967 IEEE).

Selecting new computing equipment for a company often involves the running of benchmarks on comparative systems in an effort to measure workload characteristics such as: capacity, throughput, batch turnround time, number of interactive users, response time of high usage programs, etc. Benchmarks range from the execution of typical application programs, for example a floating point number cruncher in a scientific application, to complex job control scripts, for example the reproduction of the workload from a typical day on an existing system.

Capacity planning includes measuring how the available resources are used by the system. With this data, management can schedule the workload and plan for growth. Workload can change unpredictably over