IONS

HIS

MENT

E

BLE

LA-UR-81-3502

MASTER

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-31

LA-UR--81-3502

PERC 004357

TITLE PERFORMANCE MODELING IN THE DESIGN PROCESS

AUTHOR(S) William Alexander
Richard Brice

SUBMITTED TO NCC '82
Houston, Texas
June, 1982

DISCLAIMER

By acceptance of this article the publisher recognizes that the U.S. Government retains a nonexclusive royalty-free license to publish or reproduce the published form of this contribution or to allow others to do so for U.S. Government purposes. The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

PERFORMANCE MODELING IN THE DESIGN PROCESS

ABSTRACT

Performance modeling and analysis of computer systems is often ignored during the project design phase in favor of other techniques collectively known as structured design or software engineering. We describe benefits that can result from including performance analysis as an integral part of the design process. Several different goals, timeframes, and roles played by performance analysis during system design are illustrated by three case studies of current projects at Los Alamos.

KEY WORDS: Performance modeling, prototypes, simulation, system design

Los Alamos National Laboratory
P. O. Box 1663, Mail Stop 294
Los Alamos, NM 87545
(505) 667-7676

PERFORMANCE MODELING IN THE DESIGN PROCESS

by William Alexander

Richard Brice

Los Alamos National Laboratory

Los Alamos, New Mexico

INTRODUCTION

In the past decade, since the ideas collectively referred to as "Structured Programming" and "Structured Design" have gained acceptance, the conventional wisdom has become that one should ignore performance considerations in the design phase of a computer system project in favor of modifiability, maintainability, and correctness. This doctrine assumes that one can always "tune" the system to meet performance criteria after it is built.¹ If one is astute enough in design choices so that the first running version of the system comes within, say, an order of magnitude of the performance goals, then these goals often can be met by relatively simple modifications; in this case the attention paid during design to understandable structure and modifiability will be rewarded. But performance is a result of interactions among many elements of software, hardware, and environment, and sometimes these interactions are counterintuitive. It is not uncommon for systems to

W. Alexander
MODELING

fall so far short of performance goals that only fundamental, and therefore very expensive, changes will serve.

It is our experience that the chances of such catastrophies can be reduced by applying modeling techniques during the system design process in such a way that none of the advantages of structured design need be sacrificed.

The models we refer to in this paper may be analytic or simulation; in some cases the simpler methods of operational analysis are adequate.² The point is that sufficient data should be collected, and sufficient analysis done on it, to give some assurance that the performance goals of the system being designed will be met. The kinds of data that must be gathered include hardware and software characteristics of the components of the system, measures of the behavior of the environment in which the system will run, including workload and competing systems, and even the ways in which the system being designed will alter the existing environment. Because most design projects have deadlines, it is much more likely that sufficient performance analysis will be included if the installation already has models of the proposed system's environment, or at least has data collection facilities installed.

Differences in performance goals as well as in other design objectives imply that modeling will assume different

W. Alexander
MODELING

roles, occupy different timeframes, and require different information from one design project to the next. Three design projects with which we have been involved at Los Alamos illustrate some of these points.

COMPUTING AT LOS ALAMOS

At Los Alamos, the Integrated Computing Network (ICN) allows all validated computer users at the Laboratory access to almost any of the machines or services of the Central Computing Facility (CCF). Figure 1 is a schematic diagram of the ICN. (Dotted portions indicate future plans.) At the "front end" of the network (the right side of the diagram) over 1350 terminals and remote entry stations are concentrated in stages to front-end switches (the SYNCs) so that traffic can be routed between any terminal and any worker computer. The worker computers include four Cray-1s, four CDC 7600s, two CDC Cyber-73s, and a CDC 6600. Each of the worker computers is connected to the File Transport (FT) switches and so to the "back end" of the network (left side of the diagram). The FTs are the means by which workers can send files to each other and to the special service nodes in the network. The special services provided at present by the network include an output station (PAGES) to which are attached a wide variety of printing and graphics devices, a mass storage and archive facility (CFS),³ a gateway that handles file traffic between workers and computers outside

W. Alexander MODELING

the ICN, and an integrated performance monitoring and batch job control station (FOCUS).

Although all types of computing are done at Los Alamos, most of the CPU hours on the large workers are spent executing large, long-running scientific programs. Many of these produce graphics output. Some users have a need to run programs larger and longer than even our present worker computers can handle.

THREE CASE STUDIES IN DESIGN

The distributed interactive graphics project

The goal of the distributed interactive graphics project is to improve user productivity by improving the performance of an existing interactive graphics system that runs on a large scientific computer (a Cray-1 or a CDC 7600).⁴ It is hoped that system responsiveness can be improved by adding an intelligent terminal or a larger minicomputer as a front end, and by distributing the software between the two computers. The front end is intended to handle graphics-terminal or device interactions and drive the graphics screen. Design issues include the choice of minicomputer, the hardware and software constituting the link between the two computers, the distribution of the graphics software, and whether the distribution should be static or dynamic.

W. Alexander
MODELING

A simple model of this system might include CPU and memory on the two computers, and simple links between the two computers and between the front end and the terminal. Input to this simple model would include speed and size of the CPUs and memories, and bandwidth of each link. We would also need the CPU burst sizes and distributions and the frequency and size of communications over each link for a given distribution of the software. All of this information will probably be known or can be obtained by the system designers. This model can only give an order of magnitude estimate of performance and tell the designers whether a component of the system is an obvious bottleneck.

A more realistic model of this system would incorporate other information not so readily available to the application designers. The link between the two computers envisioned by the designers is actually the back end of the network depicted in Figure 1. Contention for network services, overhead in communication protocols, error rates, and buffer space within the network are all likely to reduce the effective bandwidth of the communications links. Contention for CPU and memory resources on the two computers will also alter the communication bandwidth and the rate at which the distributed application can execute. As the distribution of software between the two computers changes, the competition this system introduces into each of the two

W. Alexander
MODELING

computer systems will change also. Quantitative knowledge about the effects of these factors will be available only if there has been an ongoing data collection effort on the mainframes and the network. There probably would not have been time or manpower available to obtain the information for the design effort if it were not already being collected.

One way to get around the problem of missing information is to actually distribute a prototype application and measure the effective communication bandwidth and rates of CPU service. The designers did this, using a DEC VAX 11/780 as the front end. The initial performance results were discouraging. The designers concluded that the chosen division of software between the two computers was wrong. They also observed that the number of ways of splitting the application was too large for exhaustive trial and error. Yet the frequency and size of communications between each pair of modules was not known, and the model needed this information to predict performance for a given split of the software. The designers therefore invested the time to build distributing tools that automated the code conversion process, allowing them to move modules from one machine to another more easily. This effort turned their prototype into a flexible data collection tool for

W. Alexander
MODELING

performance analysis, as well as providing the designers with useful insights.

The prototype multiprocessor project

The prototype multiprocessor project has as a short term goal the production of a tool for evaluating various approaches to parallelizing certain classes of numerical computations.⁵ A longer term goal is to determine the properties of applications that are candidates for multiprocessing and to determine the properties of a multiprocessor that will efficiently execute the applications at Los Alamos. The prototype is explicitly intended to serve as a data collection device in designing or specifying an eventual production multiprocessor to meet production performance goals. The prototype is required to be as flexible as possible to allow hardware simulation of a variety of architectures, such as banyan, perfect shuffle, star, ring, and hypercube networks. Its flexibility and self-measurement abilities are more important than its absolute performance.

Clearly, however, the prototype must perform well enough to allow timely and reliable execution of all test cases of interest. Thus performance analysis and modeling are appropriate during its design to evaluate choices of CPUs, memories, and hardware and software communication

W. Alexander
MODELING

mechanisms. The data required to support this first-phase performance analysis include the characteristics of the hardware components under consideration and the overhead costs of the communication protocols, all of which should be easily obtainable. We also need computational and communication characteristics of the application programs resulting from a given parallelization; here previous execution measurements and analysis of the individual application programs would be extremely helpful.

This prototype will not be installed in our ICN, although some provision will probably be made for downloading programs to it. Thus the hardware will run in isolation, and we need not take environment into account when modeling its performance. In another sense, the existing large numerical programs that will be run on the prototype constitute its environment, and information about their current performance characteristics will aid performance analysis on this prototype just as information about the network and operating systems did for the distributed graphics project.

Los Alamos has a long history of using supercomputers for scientific computation. The results of this experience have always been made available to vendors; by explaining the computing needs of scientists and by specifying the performance goals which each new generation of

W. Alexander
MODELING

supercomputers must meet, the Laboratory has contributed to their design. In the second phase of the multiprocessor project, performance modeling can be helpful in "scaling up" performance projections from the prototype to various production alternatives. Not all components will be speeded up by the same amount in going from prototype to production version, and some communication functions that were being simulated in software may be implemented in hardware. The data for this model will come, we hope, from the prototype.

The network switch project

The goal of the network switch project is to design a network switch to replace the SEL 32/77 minicomputers that act as file transports in the back end of the ICN.⁶ The new switch will have more ports so it can be connected to more network nodes than each SEL can, and it will have hardware support for error detection so that the network can provide more reliability without software overhead. The performance of the current FTs is quite satisfactory, so improving performance is not a primary motivation for this project. Building a prototype of the proposed new switch is unnecessary because the present FTs serve quite well in that role.

It is desirable that the new FTs meet performance needs imposed by ever increasing message traffic rates in the back end of the network as far into the future as possible. A

W. Alexander
MODELING

relatively simple model of the proposed switch and its environment was constructed to investigate its performance under loads in excess of those observed at present. The data needed for this model included characteristics of the CPU, buffer memory, channels chosen, and current back-end network message rates. Once again, the fact that these message rates were already being collected made the modeling effort more practicable.

Merely increasing the present message rates with the same distribution of large versus small messages and with the same set of nodes currently comprising the network constituted a reasonable extrapolation of future workload for the FTs for two or three years. But predicting workload growth in any computing system more than a few years in the future is practically impossible because there are certain to be qualitative changes in the structure of the system as well as in the way people use it. We have discussed this problem in.⁷ Our approach was to vary all the workload parameters over a wide range of values. In this way we were able to predict what loads the proposed system would handle and the characteristics of loads that may cause its performance to deteriorate.

W. Alexander
MODELING

MODELS VS. PROTOTYPES AS PERFORMANCE ANALYSIS TOOLS

In these three examples, we have seen both models and prototypes play differing roles in the design process. Obviously, prototypes serve many useful purposes in design, but we are interested here only in their uses for performance prediction and their relationship to models. Models and prototypes have different strengths and weaknesses as performance prediction tools.

Prototypes can provide "order of magnitude" performance information, especially if they can be installed in the actual environment in which the production system being designed is to run. But as performance predictors they have three major drawbacks.

- (1) It is difficult to extrapolate measured service rates of a prototype to the production system because it is not possible to predict accurately how the different behavior of the production system will interact with its environment. The production system will presumably have different memory size, computational needs, communication behavior, and interaction rates than the prototype, and these will affect the environment as well as being affected by it. Models can incorporate the environment.

W. Alexander
MODELING

(2) Although prototypes usually can be used to predict the effect on performance of simple changes in the system, they cannot do the same for changes in the environment. For example, we know that the distributed graphics system will run faster if we are allowed to make one simple change in the scheduling algorithm of the operating system used on the large scientific computers, but this fact could be learned only from a model that incorporated the operating system.

(3) Prototypes usually cannot be made as flexible as models; hence, prototypes cannot be used to predict the effect of fundamental design changes. This point requires further discussion.

Flexibility can sometimes be achieved in prototypes, but usually at higher cost than in models. Prototypes are not always inflexible, but it is unusual to spend so much time and effort in the computer system design process on a prototype. The multiprocessor prototype was consciously designed as a hardware simulator of a variety of multiprocessor architectures, and this capability makes it very like a model. One can imagine a continuum characterized by increasing cost and complexity as one moves from operational analysis through models and prototypes to the actual production system. The design process is

W. Alexander
MODELING

characterized by making choices among alternatives; typically one can try different alternatives more quickly and much more cheaply with a model than by actually implementing them. There is a subjective cost/benefit function that applies to the choice between trusting the results of a model and implementing a prototype. The cost of implementing a prototype is usually more easily justified in extreme situations, such as designing with new technologies (including new software technologies) or in completely unfamiliar situations.

Modeling as part of the design process produces benefits besides performance prediction. Because modeling is a relatively quick method of "implementing" a design, issues that normally come up only during implementation sometimes arise much earlier. Ambiguities in the specifications may be noticed early, and if these would have necessitated redesign, time and money can be saved. Beyond merely predicting performance, modeling can specify the performance levels that individual components will have to achieve in order for the whole system to meet its performance goals. Modeling can also help explore the behavior of the system under a variety of workloads or other external conditions.

W. Alexander
MODELING

Although modeling is an art requiring some expertise, it is not nearly so difficult as it used to be. A number of commercial packages and languages are available to support analytic and simulation modeling of computer systems.^{8,9} It is not unreasonable to develop models and in-house modeling capabilities with the aid of these tools, and the investment can pay repeated dividends.

There are, of course, difficulties in integrating modeling into the design process. It is particularly difficult to analyze the performance of software that has not been written. Although the work of Smith and Browne¹⁰⁻¹⁴ is promising, much remains to be done. We do not imply that performance modeling as part of design always works or can answer all questions, only that it has proven useful in our experience.

CONCLUSIONS

Here, in capsule form, are some lessons we have learned trying to integrate performance modeling into the design process.

Performance modeling should play a central role in system design; ignore it at your peril.

The role of performance modeling is not the same in all design projects. Clearly specify your performance goals

W. Alexander
MODELING

and what factors will affect performance; then try to model those factors.

Obtaining the data for the models can be a major problem; ongoing measurement projects are always worthwhile.

Prototypes can be valuable data gathering tools if they are instrumented for this purpose.

Anticipate the effect of environment on the system you are designing, and the effects of the system on the environment.

Include the performance analyst on the design team from the beginning; if he is perceived as an "outsider," he is more likely to be ignored, especially if decisions have already been made.

W. Alexander
MODELING

REFERENCES

1. Yourdan, E., and L. Constantine. *Structured Design*. Englewood Cliffs, New Jersey: Prentice-Hall, 1979, pp. 290-291.
2. Denning, P. J., and J. P. Buzen. "The Operational Analysis of Queueing Network Models." Computing Surveys, 10 (1978), 3, pp. 225-262.
3. Blood, M., R. Christman, and B. Collins. "Experience with the LASL Common File System." Digest of Papers from Fourth IEEE Symposium on Mass Storage Systems, April 1980, pp. 51-54.
4. Hamlin, G., and J. E. George. "Experiences with Distributing Graphics Software Between Processors." To be published.
5. Trujillo, V. A. "Multiprocessor System Description." Los Alamos National Laboratory memorandum to Bill Buzbee, June 26, 1981.
6. Tolmie, D., R. Christman, T. Klingner, C. Stallings, and R. Jurgens. "Network Switch Functional Design Requirements." Los Alamos National Laboratory memorandum to Robert Ewald, March 31, 1981.

W. Alexander
MODELING

7. Alexander, W., and R. Brice. "Long-Range Prediction of Network Traffic." Proc. CPEUG81, November 1981.
8. ASPOL Reference Manual. Minneapolis: Control Data Corporation, Pub. No. 17314200-B, 1975.
9. The Performance Analyst's Workbench System Users Manual. Austin: Information Research Associates, 1981.
10. Browne, J. C. "Designing Systems for Performance." Proc. ACM/SIGMETRICS Conference on Measurement and Modeling of Computer Systems, September 1981, pp. 1.
11. Smith, C. U., and J. C. Browne. "Performance Specifications and Analysis of Software Designs." Proc. Conference on Simulation, Measurement and Modeling of Computer Systems, August 1979.
12. Smith, C. U., and J. C. Browne. "Modeling Software Systems for Performance Predictions." Proc. CMG X, 1979.
13. Smith, C. U., and J. C. Browne. "Aspects of Software Design Analysis: Concurrency and Blocking." Proc. Performance 80, May 1980.

W. Alexander
MODELING

14. Smith, C. U. "The Prediction and Evaluation of the Performance of Software From Extended Design Specification." PhD Dissertation, Dept. of Computer Science, University of Texas, August 1980.

Figure 1. LOS ALAMOS INTEGRATED COMPUTING NETWORK