WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

The Two Sides of Time Sharing

Martin Greenberger

Sloan School of Management and Project MAC
Massachusetts Institute of Technology
127-65

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

The Two Sides of Time Sharing

Martin Greenberger

Sloan School of Management and Project MAC
Massachusetts Institute of Technology
127-65

Based on material presented to the IFIP Congress 65.

Work reported herein was supported (in part) by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01). Reproduction in whole or in part is permitted for any purpose of the United States Government.

Development of the OPS system was a cooperative undertaking with Anthony Gorry, Malcolm Jones, James Morris, David Ness, Mayer Wantman, and Stephen Whitelaw.

1. Introduction

There is more than one way to bisect the new subject of time sharing into contrasting aspects. For example, we could view the *good* and the *bad* of it, a dichotomy that is guaranteed to provoke lively discussions. But these discussions, although healthy, are premature. They are a little like the Harvard sweatshirts we see on some three-year olds. They focus attention on important issues and goals, but come too early in the developmental process to be very meaningful.

My bisection is not going to be into the *good* and the *bad*, but rather into the *system* and the *user*, both of whom we consider good for now. The distinction between system and user is reflected in the double-edged acronym of M.I.T.'s Project MAC: *Multi-Access Computer* refers to the physical tool or system, whereas *Machine-Aided Cognition* expresses the hopes of the user. The distinction is also present in the phrase *on-line real-time*. *Real-time* depicts the performance of the processor, while *on-line* describes the status of the user. The distinction may even be found within the scheduling algorithm of an on-line operation, which happens to be the subject of a recent operations research paper (11).

Speaking of operations research, there was a very fine article by Georges Brigham published about 10 years ago in the Journal of the Operations Research Society of America (2). It won the Lanchester Prize for the best O/R paper of

the year. It dealt with a congestion problem in an aircraft factory, and it contained an interesting queueing analysis of a tool counter. The analysis viewed the service operation at the tool counter mathematically, first from the side of the tool clerk or server, then from the side of the mechanic or user. This approach gives two mathematical problems that are completely equivalent. We might call them duals. Unless we change the statement of service objectives in passing from one side of the counter to the other, both problems yield identical solutions.

The point is, of course, that the service objectives normally <u>are</u> different on the two sides of the counter. The server and the user look at the operation through different glasses.

Behind the tool counter the clerk moves as quickly as possible. He strives for efficiency and dispatch to prevent congestion on the other side. In his haste, he may fail to notice which customer has been waiting the longest. This oversight does not affect average customer wait, so long as no one starts a brawl from impatience. The clerk may also show a slight preference for customers who wish the fewest tools, a bias that actually reduces average wait. During free periods, the clerk nibbles away at <u>background</u> tasks which he tries to get done without neglecting the more important service operation in the <u>foreground</u>.

In front of the counter, the mechanic is eager to get back to his job. His foreman may be marking time, and a

costly machine may be sitting idle in need of repairs. He is miffed at having to wait longer than the next fellow. He is upset by having to accept a Stillson instead of a single-head wrench, or a bolt that turns out to be the wrong size and requires him to return to the counter. If the mechanic had his way, all the tools he needs would be available from one counter, without waiting, error, replacement, discussion, or red tape.

The sets of objectives of user and system are not necessarily contradictory. On the contrary, the server's objectives usually are fashioned so as to accommodate the user, but typically the <u>average</u> user or the aggregate of users, rather than the <u>individual</u> user.

Now let's talk about computers. We really did not need the tool-counter example, since the same two-sided phenomenon is familiar in the computer field. In a conventional computer center the operations manager strives for greater equipment efficiency through batch processing and a closed shop, while the programmer longs for faster response time and closer touch with his run. This state of affairs has lead to conflict, and this conflict has been the single most important motive for the development of time sharing. Time sharing is definitely a concession to the user and a recognition of his point of view.

2. Time-sharing Supervisors

The first order of business in making time sharing work on a meaningful scale was to solve the technical problems occasioned by this new mode of operation. Supervisors were built to:

a. Accept input continuously from the remote terminals of a number of simultaneous users,

b. Parcel out successive pieces of the computer's time to users equitably,

c. Schedule this time so that the computer seemed immediately responsive to most requests,

d. Protect the programs of each user from damage by an errant neighbor,

e. Manage input-output and queues of interrupts,

f. Transfer programs flexibly between primary and secondary storage,

g. Maintain data files,

h. Provide error detection and security from unauthorized use, and

i. Allow batch processing to continue in the background, concurrently with operation of the remote terminals in the foreground.

Programming the supervisors to solve these technical problems were major undertakings, partly because the job had not been done before, and partly because the computers available at the time were not designed to do the job.

Emphasis was placed on the server side of the operation - getting the processor to time share.

A few time sharing supervisors with limited horizons did cross over to the user's side (15,18), while the bigger and broader supervisors (3,17) provided a framework and the means for others to do so. CTSS, the time sharing supervisor at Project MAC, is structured as a modular set of subroutines that encourages further evolution (4). Users may add to CTSS by writing and compiling programs in a variety of languages, including FORTRAN, MAD, ALGOL, and FAP. There are flexible facilities for editing. From this base, systems can be hand tailored for particular applications, such as engineering design (16), stress analysis (1), and symbol manipulation (21). Each of these systems is in a sense a fulfillment of CTSS in a special user's domain.

3. User Systems

The development of time sharing supervisors has been an approach to man-machine operation from the server's side of the counter. In contrast, certain systems have approached from the user's side (5,13,20). These systems were designed to facilitate man-machine interaction in a specific problem context. Sharing time among users was not a prerequisite or initial concern, although it makes good sense as an afterthought, and some of these systems have

since moved in that direction. Sketchpad, for example, which is one of the earliest and most vivid of the interactive user systems, is being implemented in an extended 3-dimensional form on the Project MAC computer.

The full power of an on-line system is attained when the user is able to shift flexibly and easily between a wide variety of different activities without incurring high set-up costs or requiring awkward adjustments. The raw computer can do many wondrous things, as we all know, but this versatility does not benefit most users unless it is placed at their fingertips. The average user should not have to carry his water to the faucet. The system should take care of this for him.

Specifically, it should be convenient for the on-line user to regulate echo checking, make calculations, display intermediate results, obtain helpful guidelines, and view key variables. He should have the ability to move easily between program modification and execution without the need for recompilation. He should be able to execute a subroutine by simply referring to it symbolically, alter its arguments without fuss, compound it with other subroutines, and treat the compound as he treats the parts. These features are important to him whether he is solving a mathematical problem, building a simulation model, designing a man-machine decision procedure, constructing a real-time operation, or simply doing some data analysis. And if he is ping-ponging among several such activities, the value of

these features, when provided by a single system, is multiplied.

The OPS system at Project MAC was developed to provide these features (8,9,10). It is an on-line system that is multi-purpose to the individual user, just as CTSS is multi-purpose to the community of users. The OPS system uses CTSS and is a natural extension of it.

4. An Illustration

Suppose we have the following problem to solve: our single processor time shares N terminals, using a round-robin scheduling procedure with a quantum of one second; we want to know how long a one-second request has to wait, on the average, for values of N ranging from 1 to 50; we also want to know how much batch processing can be accomplished in the background during periods when none of the N terminals is requesting service.

For N equal to the number of terminals currently connected, we can answer these questions by making the time-sharing operation introspective; that is, by having it gather statistics about itself. But we cannot use this device in general, unless we are willing to subject the operation to a stream of perturbations in N. The public relations side of such an approach to the problem gives even the most callous administrator substantial reason to pause.

Simulation is a more feasible tack, as any good student of the art will hasten to tell us. Changing N in a simulation does not disturb the clientele, and can be accomplished with great dispatch.

A still more elegant solution to the problem is to develop a queueing model with N as parameter. By virtue of suitable simplifying assumptions, the model becomes a set of difference equations. A compact algorithm can be programmed to solve the equations recursively.

The benefits and drawbacks of empirical data gathering vs. simulation vs. mathematical analysis are well documented. What we would really like to be able to do is a little of all three, back and forth, until our gradually increasing comprehension of the problem becomes the desired solution.

This iterative process can expand into a series of runs over several weeks or even months in a traditional production-oriented batch setting. Consider the debugging sequence required to program the algorithm; add to this the statistical analyses necessary for estimating relationships and reducing data to classical probability distributions, combine all that with repeated complications of the simulation and contemplation of its results; and you see how this process can be very costly in time. In addition, the process requires a flexible simulation system, like Simscript, good statistical subroutines, like Chi-square and multiple regression, debugging aids, like traces and dumps,

a desk calculator, graphical displays, and so on.

In a time-sharing environment, we can hope for a better match of computer with the creative or problem-solving process. Indeed, this has been one of the main motivations in the development of time sharing. We can hope for an on-line system that provides us with the variety of facilities we need, and allows us to switch back and forth between debugging, calculating, modeling, simulating, modifying, displays, and analysis, with minimal effort and expense.

The OPS system has these features, and brings us closer to the day when the total process that we have been describing might occupy an able researcher no longer than one interesting afternoon at his terminal.

5. The OPS System

The OPS system is a multi-purpose modular apparatus with on-line facilities for: symbolic matrix and vector calculations; statistical analyses such as multiple regressions; FORTRAN and MAD type programming with instant execution without need for compilation; incremental modeling of computer simulations; and graduated levels of user interaction. Switching among facilities is simple and entails minimal set-up costs, giving great flexibility of use. Subroutines are added or subtracted with ease, allowing the user to adapt the system to his own

specifications. Information about the system is available on line.

The OPS system has recently been made a publicly available command at Project MAC. In giving the command, the user can specify operators stored in his personal file, and these will be loaded with the standard operators of the OPS system. Operators are simply precompiled BSS subroutines. They can be loaded at load time, or dynamically any time thereafter.

By means of the OPS system, the user can:

1. Call operators by name from the console;

2. Compound operators into MAD or FORTRAN type programs that may themselves be called like operators;

3. Execute (or test) while compounding;

4. Compound (or remember) while executing;

5. Edit compound operators and their parameters either from the console or from a compound operator;

6. Refer symbolically to common storage by cell or array name;

7. Dynamically modify the mapping and contents of common storage at execution time;

8. Add operators without limit and bring them into core as needed;

9. Intensify or reduce the level of user interaction by means of switch settings.

6. Standard Operators

Among the standard operators that come with the system are:

1. SET, which is like the general assignment (=) statement of FORTRAN and MAD, except that its symbols can denote complete matrices and vectors, as well as elements. Thus, executing the operator

```
SET D = A + B * LOG.(C)
```

may cause a 20 x 20 matrix A to be added to the product of a 20 x 20 scalar matrix (whose elements are all equal to the log of the constant C). The scalar matrix need not be stored explicitly. SET also provides general matrix multiplication and transposition of matrices. Elements of matrices are referred to by the customary parenthesis notation. Dimensions may be symbolic, as in the following example:

```
SET D(I,J) = A(I,C)/SQRT.(B(C,J))
```

2. IF and IFB (pronounced IF-B), which are similar to the MAD conditional statement, and cause a branch depending upon the truth value of a Boolean proposition.

3. REPEAT and GOTO, which allow for looping, indexing, and unconditional transfers.

4. TYPE, which permits symbolic entry or print out of unformated information to or from common storage.

5. SAVEC and LOADC, (pronounced SAVE-C and LOAD-C), which save and retrieve information between common storage and disc.

6. SAVES, LOADS, ENTERS, ERASES, RESETS, and PRINTS (prounouncd SAVE-S etc.), which save, retrieve, modify, and print the symbol table.

7. SAVEK, LOADK, EDITK, PRINTK, ENTERK, ERASEK, NAMEK, CALLK, RETRNK, and TAKEP, which save, retrieve, edit, print, delete, initiate, name, rename, execute, and terminate compound operators, and distribute their parameters.

8. TEXT, which prints out prespecified textual information during execution.

9. FIT, which performs a multiple linear regression of a dependent variable on a set of independent variables. For example

```
FIT Y TO X1 X2 X3
```

fits the observations stored in the vector Y to those stored in vectors X1, X2, and X3. A vector of weights may be specified, and a vector containing the residuals of fit may be created.

10. SCHED, RSCHED, DELAY, CANCEL, LOCAL, CALLAK, CALLBK, and RETRNA, which form a package of simulation operators for on-line modeling using an agenda that schedules actions and events.

11. FORMAT, INPUT, and OUTPUT, which make possible arbitrarily formated reading and printing in the sense of FORTRAN.

12. VIEW, which gives a snapshot of key system variables for error checking.

13. ERROR, which provides diagnostic information on common user mistakes.

14. CTSS, which allows execution of any CTSS command from within the OPS system.

15. GUIDE, which provides descriptive information on the OPS system.

16. DRAW, which furnishes a random draw from the exponential, normal, or rectangular probability distribution.

7. The Future

Future time sharing supervisors will run on computer systems having many processors and many active memory modules, flexibly interconnected. These systems will correct a number of current technical deficiencies in time sharing operation. They will provide for the dynamic allocation of storage through page turning, so that a user will not have to specify or load all his programming and data requirements when he begins to interact with the computer; and they will permit the sharing by different users, not only of processors and storage, but also program packages assembled as pure procedures in symbolically identified segments of memory.

These advances are on the server's side of the operation. There is still much work to be done there, and it will benefit the user directly. Progress will also come on the user's side in the form of more natural languages, a much wider range of terminal equipment, improvements in graphical and audio input-output facilities, and bigger and better OPS-type systems.

The situation today in computation is about what it was after the turn of the century in the distribution of electricity (12). The first electric service in New York City illuminated the early Edison light bulbs, and did little else, except run an occasional elevator, toaster, or iron. There was plenty of skepticism about the benefits of distributing electricity in those days. Each electric light installation was required by law to be accompanied by a gas light alongside to cover what was at first considered to be the very substantial likelihood of failure. As it turned out, very few of the gas lights ever had to be used.

Much progress has been made in the design of distribution lines and turbines since then, but the system developments are dwarfed by what has happened on the user's side--household appliances, radio, television, hi-fi, communications, heating systems, air conditioning, ... , computers.

Today there is great intellectual challenge in designing better computer system configurations, as well as the programming concepts and software required to make the

systems work as intended (and, historically, even better than intended). Many of the finest minds in the computer field are dedicated to the task. At Project MAC, system programming receives top priority, and the twofold MAC objective is temporarily dominated by a concerted effort to foster and implement the utility concept.

> The broad system goal of Project MAC may be regarded as the development and operation of a community utility that is capable of supplying computer power to each customer where, when, and in the amount needed. (7)

Work also is in progress at Project MAC on the user's side of the counter, and OPS is only one of a large number of examples. In coming years we may expect facilities for users to expand dramatically in magnitude and importance, just as applications of electricity have expanded during the past several decades. More and more scientific brainpower and creative energy will shift in this direction.

The shift has just begun. Only after it has been underway for some time will it really be meaningful to sit down and talk about the <u>good</u> and the <u>bad</u> of time sharing.

8. References

1. Biggs, J. M. and Logcher, R. D. Stress: A Problem-oriented Language for Structural Engineering. Technical Report MAC-TR-6, M.I.T., 1964.

2. Brigham, G. "On a Congestion Problem in an Aircraft Factory", Operations Research, (November 1955).

3. Corbato, F. J. et. al. The Compatible Time-Sharing System. M.I.T. Press, Cambridge, 1963.

4. Corbato, F. J. and Glaser, E. "Introduction to Time Sharing," Datamation, (November 1964).

5. Culler, G. J. and Fried, B. D. The STL On-line Computer. TRW/Space Technology Laboratories, Redondo Beach, California, 1964.

6. Dennis, J. B. "A Multiuser Computation Facility for Education and Research", Comm. of the ACM 7, (September 1964).

7. Fano, R. M. "The MAC System: A Progress Report", IEEE Spectrum, (January 1965).

8. Greenberger, Martin. The OPS-1 Manual. Technical Report MAC-TR-8, M.I.T., 1964.

9. Greenberger, Martin. A New Methodology for Computer Simulation. Technical Report MAC-TR-13, M.I.T., 1964.

10. Greenberger, M., Jones, M., Morris, J., and Ness, D. "Trial Launching of the OPS-3 System", MAC-M-233, M.I.T., April 1965.

11. Greenberger, Martin. "Priority Scheduling of Time-Shared Computer Systems", Bulletin of ORSA, Vol. 13, Supplement 1, Spring 1965 (Abstract).

12. Greenberger, Martin. "The Computers of Tomorrow", The Atlantic Monthly, (May 1964).

13. Hellerman, H. "Experimental Personalized Array Translator System", Comm. of the ACM 7 (July 1964).

14. McCarthy, J. "Time-Sharing Computer Systems", Management and the Computer of the Future, (M. Greenberger, Ed.), M.I.T. Press, Cambridge, 1962.

15. Morrisey, J. J. "The Quicktran System", Datamation, (November 1964).

16. Ross, T. D. and Feldman, C. G. Verbal and Graphical Language for the AED System: A Progress Report. Technical Report MAC-TR-4, M.I.T., 1964.

17. Schwartz, J. "A General-Purpose Time-Sharing System", AFIPS Proceedings 25, (1964) p. 397.

18. Shaw, J. D. "The JOSS System", Datamation, (November 1964).

19. Stotz, R. "Man-Machine Console Facilities for Computer-Aided Design", AFIPS Conference Proceedings 23, (1963).

20. Sutherland, I. E. Sketchpad: A Man-Machine Graphical Communication System. Technical Report 296, Lincoln Laboratory, January 30, 1963.

21. Weizenbaum, J. OPL-1: An Open-Ended Programming System within CTSS. Technical Report MAC-TR-7, M.I.T., 1964.