# Performance Measures for Online Systems

John Ulmschneider and Patrick Mullin

## System performance: an overview

Librarians assess a library automation system by many parameters, such as the richness of its functionality, the ease of use of its interface, and its overall purchase and operating cost. One of the most important criteria is a system's *performance.* It is not uncommon for librarians to praise or condemn a system based on performance alone. But what do library managers mean when they speak of "system performance"? The "performance" of a computer application system can mean different things to different observers.[1,2] At one extreme, many librarians treat the functionality of the applications software as the main criterion of performance: what does the application software actually do, and how well or thoroughly does it do those things? At the other extreme, the computer industry has developed a number of performance measures for computer systems that distinguish the computing hardware's capabilities from the way application software uses those capabilities. Hardware evaluations center on such parameters as central processing unit (CPU) speed, data retrieval and transfer speed from disks, memory architecture, and the like. Software evaluations assess many aspects of the application's operations to build a final picture of its performance: the application's use of processing resources, disk storage and retrieval demands, instruction mix, response time, memory requirements, and other parameters (Figure 1).

FIGURE 1.

**Hardware evaluation compared with software evaluation**

| Hardware evaluation | Software evaluation |
|---|---|
| CPU speed in million instructions per second (MIPS) | response time to interactive users |
| memory speed, caching | subroutine speed for boolean combinations |
| disk seek and read time | disk storage demands for data and work space |
| number and speed of communications channels | number of concurrent users or terminals supported |

**John E. Ulmschneider is the Assistant Director for Library Systems for the North Carolina State University Libraries. Patrick J. Mullin is Systems Librarian at the University of North Carolina at Chapel Hill and Interim Director of the Triangle Research Libraries Network.**

Performance measures for evaluating library automation systems include something from both ends of the spectrum. In general, library managers are not concerned with the capabilities of the hardware platform used for a system; they are concerned only with the way the application software performs for the user. Librarians also sharply distinguish responsiveness for interactive operations, where users query the application system in real time, from batch operations, where a series of programs is executed automatically by the computer. The performance evaluation of a library application system is assessed through three parameters concerned primarily with interactive operations:

*Response time:* how quickly a computer system delivers a response to a user query in an interactive environment;

*Application efficiency:* what computing resources (processor cycles, memory, disk space) are required by software to deliver an adequate response time;

*Capacity:* the volume or amount of work a system can perform with a given amount of hardware resources, for instance, the number of concurrent searches it can perform.

The relationship between these performance parameters is not straightforward. For instance, suppose an application system is very efficient on machine resources, with clever and tight code that minimizes the use of memory. Such an application might squeeze the most from the machine resources available to it, but might deliver poor response time because it does not use enough memory to speed up, sort, and merge operations. Or suppose an application delivers very fast response time, but requires enormous machine resources to do so. Such an application likely will be too expensive to maintain.

Because the relationships between performance parameters are complex, assessments of library systems must include data on all three performance variables. Library managers should require library application programs to meet certain minimum standards. For instance, the system ought not to require a supercomputer to perform boolean searches and should take less than five minutes to respond to interactive queries. Managers do not, however, expect them to show ideal scores in all three areas.

Each of the three performance measures lends itself to wide discrepancies in definition and application. An "efficient" program can end up using considerably greater memory resources than an inefficient program if it seeks to minimize the use of slow mechanical devices(e.g., disk drives, tapes) by storing volumes of data in main memory for instant access. On the other hand, a system providing high capacity might do so only under ideal conditions, for instance, when all the online catalog queries are known item searches. Because of these discrepancies, vendors and buyers of library systems should define exactly the nature of performance parameters expected of a system. In general, efficiency and capacity in purchase contracts are largely system-dependent measures, and standards for their performance pertain only to particular hardware-software combinations. Librarians have reached a general consensus, however, on response time: interactive queries should average no more than three to five seconds from transmission of a query to receiving an answer.

**Response time**

Of the three parameters, response time is both the most widely applied and least understood measure. For most library managers, re-

**. . . response time is both the most widely applied and least understood measure.**

sponse time generally means the time between transmission of a query to the system (by pressing the return key) and the time when characters first appear on the screen in response to the query. Many factors in the application system affect response time, among them the speed of disk drives and how much the software uses them, the memory available of the application software, and the number of concurrent users on the system. Most measurements of response time, however, include three distinct components:

(1) *Transmission time:* the time required by the transmission channel to deliver queries from the terminal to the computer, and data from the computer to the terminal;

(2) *Application response time:* The time required by the application after receiving a query to process the query and to begin transmitting a response to the terminal; and

(3) *Display time:* The time required by the terminal to display the entire reply from the computer.

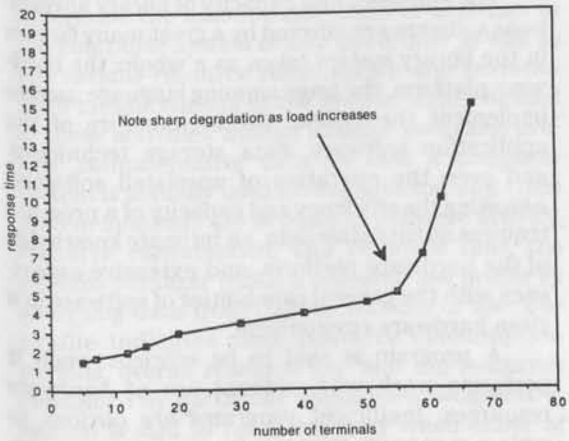
Response time measures usually do not distinguish the contribution of each element to the overall response time, even though users attribute the entire response time solely to the application software. Under normal circumstances, components 1 and 3 make a negligible contribution to response time, in the range of milliseconds. In special circumstances, however, their contribution may be significant. For instance, in local area networks, propagation of queries and responses through several miles of cables, translators, bridges, and routers can introduce significant delays. Modem connections may also introduce considerable delay. Even directly wired connections can slow down response time if the line speed is low or the output device is a printer. Because of these factors, vendors of application systems usually agree to meet response time criteria only in the context of control over the entire hardware plant, and they specify dedicated terminals using the fastest and most secure communications possible.

How is response time actually measured? Three approaches are possible. In the *stopwatch method*, one evaluator enters a query to an online catalog while a second evaluator times the query with a stopwatch. The second evaluator starts the timer at the instant the return key is pressed and stops the timer at the instant the reply begins to appear on the screen. Generally, the evaluators employ a carefully designed script that exercises most of the searching functions of the system in simple and more complex searches. The response time is averaged over all the searches and over a number of sessions. By using a number of terminals and users simultaneously, the evaluators can mimic a real online environment with multiple simultaneous searchers, subjecting the application system to a *stress test* or *benchmark test*. In its simplest form, a stress test measures the responsiveness of a computer system as more and more of its functions are used simultaneously. Stress tests usually identify a peak load, or number of concurrent users, beyond which performance becomes unacceptable. (See Figure 2.)

The stopwatch method is simple to implement, cheap, flexible, and expandable. It is also, by and large, a reliable method if done carefully. Nonetheless, human reaction time, communication time, and other variables may affect the final results.

FIGURE 2.

**Typical response time under load revealed by stress test**



In the *simulation method*, a desktop computer or a simulation program on the library system's computer is equipped with a search script, similar to that used in the stopwatch method, and is connected to the library application program. The desktop computer or the simulation program transmits queries and receives replies from the library application program, and measures very precisely the time between transmitting a query and receiving a reply.

The simulation method requires modest technical expertise to implement. It retains all the advantages of the stopwatch method while eliminating variables introduced by human participants. Just as with the stopwatch method, evaluators can establish a bank of computers, or a number of simulation programs, to execute simultaneously the prepared search scripts in order to subject a computer system to a stress test.

The *system monitor method* uses software on the computer system itself to record response time data on devices and software supported by the system. Most general-purpose hardware platforms for library automation systems provide system software to record statistics on the performance of a program in a number of areas: how much memory it uses, how often it accesses disk drives, how quickly it answers requests from terminals, and how much data it sends to them. Mainframe computers have used such programs for years to generate billing data, and they have tuned them to a high degree of accuracy and comprehensiveness.

Both the stopwatch method and the simulation method rely on searching scripts as models of anticipated user behavior to gauge actual system performance. The design of such scripts seldom reflects user reality. (Recent attempts to base scripts on statistical and qualitative evaluation of transaction logs, which are verbatim records of every query to a library automation system and every response of the system to the user, are improving the design of such scripts.) Instead, the scripts are thorough exercises of every aspect of an application system's functionality, with a mix of commands that cover every possible function in the system. The application system's response time to such a mixture certainly reveals its response in carrying out specific functions, but may not reflect its actual responsiveness in operational use.

The system monitor approach, in contrast, is a strictly empirical one. Rather than develop a model of user behavior, it exhaustively records an application's responses to actual users and search loads. Since it is capable of analyzing the data from hundreds of thousands of commands entered over extended periods of time, it also draws upon a much larger universe of experience that any model can construct. As a result, its measurements provide a more accurate and complete picture of response time than user models do and are free of biases resulting from a poorly designed mix or scheduling of test queries.

The system monitor method has two additional advantages. First, it is not implemented as a special test requiring staff participation, special test computers, or special software. Monitor software runs as part of the normal operating environment and generates reports on terminal activity and response time as part of the daily activity log of the system. Second, it provides a much more detailed and comprehensive picture of an application's performance, including data on its use of machine resources as well as response time. Data provided by system monitor programs bear importantly on understanding an application's efficiency and throughput, for instance.

System monitors measure response time at the point where the communications system connects to the computer, so that communication delays are not included in the response times. Managers can use this data to evaluate software performance independent of the communications plant and to help attribute response time problems to either software or communications. On the other hand, the actual response time experienced by a user is the most visible indication of an online system's performance. In general, library managers supplement system monitor reports with periodic monitoring of actual user response time, including stopwatch measurements when necessary.

Because of their inherent limitations, response time measurements that require search models are best limited to acceptance testing: the final tests of functionality and performance before a library accepts a vendor's system and pays for it. Managers may use them to strike periodic benchmarks, but they should recognize that the models do not usually reflect the actual use or response time of the system. System monitors should be used for pre-purchase tests by obtaining data from operational sites; such data may point to performance problems before acceptance. The data may prove particularly useful if the desired system is installed at a site closely matching the profile of the purchasing site, with user populations similar in size, interests, and activity, and identical hardware resources. Even under these circumstances, system monitor results should not be the basis of final acceptance for payment; it is simply too easy to overlook differences between one installation and another. After installation, however, library managers should receive regular system monitor data that reports actual performance of the software: response time, computer resource use, and the like.

Numerous observers have raised two particular concerns with respect to measuring response time in library systems.[3,4] First, online catalog searches vary widely in the amount of work they require of a program. Many searches are direct, known-item searches, where the program need only retrieve single records. Other searches may require locating, performing combinatorial operations with, and retrieving large sets of records. Second, the definition of a "search" is open to debate. Is a search concluded only when the user locates the information required? Or should library managers consider a search equivalent to a transaction, defined as a single interaction between user and computer?

Methods that rely on models address these concerns by using search scripts that exercise most of the functionality of the application software. The scripts include searches that require considerable processing as well as known-item searches, and usually provide for multi-step searches (e.g., perusing an index list, selecting a retrieval set, and then narrowing the set to find the desired item). The system monitor method, on the other hand, cannot distinguish difficult from simple searches; it measures the response time for individual transactions, regardless of their type. System monitor methods compensate for this limitation by processing a very large transaction volume, which ultimately produces a statistically valid judgment of normal response time.

## Efficiency and Capacity

The efficiency and capacity of library applications software are affected by a great many factors in the library system taken as a whole: the hardware platform, the programming language used to implement the system, the architecture of the application software, data storage techniques, and even the operation of unrelated software. Assessing the efficiency and capacity of a program requires quantitative data, an intimate knowledge of the hardware platform, and extensive experience with the general capabilities of software in a given hardware environment.

A program is said to be efficient when it performs work with optimal use of hardware resources. Inefficient programs are obvious to system managers; they require prodigious resources to perform simple tasks. Efficient programs are not so easily pinpointed. Generally only close examination of the actual code or architecture of an efficient program reveals areas for improvement (or admiration). For example, a programmer can improve the efficiency of a program by decreasing disk drive access, memory resource use, or CPU time to perform a given task. Efficiency judgments extend to suites of applications programs as well as to single programs, since library applications often consist of a number of programs performing different tasks in concert. The overall architecture of a system can be considered efficient or inefficient, depending on how it uses system resources.

Efficiency bears directly on capacity. Capacity measures the amount of work a computer system can perform given a certain mix of machine resources and programs. Capacity relates to the computer system as a whole, not just to a given applications program, since both available hardware resources and a program's use of them determines the amount of work possible. Efficient programs make better use of hardware resources. A computer running efficient programs can perform more work in any given machine configuration than one with inefficient ones. For instance, efficient programs may permit the system to handle up to twenty concurrent users, while inefficient programs may reduce this capacity to only ten or twelve. The types of work performed on a computer system also affect its capacity. Certain users or activities require more hardware resources than others and can affect overall capacity significantly. For example, catalogers and other technical support users editing the database usually require a great deal more CPU support, disk access, and the like than someone merely searching the catalog.

The first step in measuring capacity is to determine the amount and kinds of activities in the computer system at any given time as well as the various resource consumption and performance measurements of the system while engaged in those activities. System monitor programs provide comprehensive data on how a computer system is actually used throughout the day. Once system monitors are in place to measure activity, resource consumption, and response time, the systems manager builds a resource use profile by analyzing data from days or months of use. The profile indicates peak resource consumption periods, overall resource use, and the resources consumed by particular application programs. A system is said to reach capacity when either of two events occurs:

(1) Consumption of hardware resources reaches defined maximum limits. The defined maximum resource use of hardware platforms, beyond which additional resources are recommended, varies from manufacturer to manufacturer. Most manufacturers consider a CPU saturated, for instance, at about eighty-five percent average use. Disk storage reaches a maximum when growth space is not sufficient for short-term growth.

(2) Response time for online users degrades below a defined maximum. When response time degrades above an average of five seconds for most transactions, for instance, the computer system no longer has capacity for additional users.

Systems managers and librarians employ resource use profiles precisely to avoid reaching capacity on a computer system. By monitoring system resource consumption through frequent profiles, managers can model future system demand and project resource requirements necessary to maintain adequate response time, disk storage, and other resources.

## A Management Example: Performance Measures at TRLN

The Triangle Research Libraries Network (TRLN) is a cooperative library automation project of Duke University in Durham, North Carolina State University (NCSU) in Raleigh, and the University of North Carolina at Chapel Hill (UNC-CH). TRLN has focused on the Bibliographic Information System (BIS) as the core and first module of an integrated library system. The circulation control module is currently undergoing beta testing at NCSU. A vendor-supplied acquisitions and serials control system will be implemented by all three institutions. The TRLN system is a distributed system. Tandem computers located on each campus support the catalog for that campus. TRLN's unique software allows library patrons to search any one of the catalogs in the network or to search multiple catalogs simultaneously, dis-

FIGURE 3.

**Summary Terminal Use And Response Time Report By Terminal**

DATE OF THIS REPORT: 03/06/90 RUN TIME: 04:10:56 AM

| TERMINAL-NAME | | REP-DATE | TOTTRAN | AVERAGE RESPONSE |
|---|---|---|---|---|
| $ATP0 | #VAX1 | 03/05/90 | 219.00 | 4.56 |
| $ATP0 | #VAX2 | 03/05/90 | 674.00 | 4.88 |
| $ATP0 | #VAX3 | 03/05/90 | 92.00 | 2.67 |
| $ATP0 | #VAX4 | 03/05/90 | 666.00 | 3.65 |
| $ATP1 | #DCA1 | 03/05/90 | 1089.00 | 4.20 |
| $ATP1 | #DCA2 | 03/05/90 | 768.00 | 3.64 |
| $ATP1 | #VAX5 | 03/05/90 | 290.00 | 4.00 |
| $ATP1 | #VAX6 | 03/05/90 | 481.00 | 3.60 |
| $ATP2 | #DCA3 | 03/05/90 | 623.00 | 3.83 |
| $ATP2 | #DCA4 | 03/05/90 | 961.00 | 4.19 |
| $ATP2 | #DCA5 | 03/05/90 | 164.00 | 3.30 |
| $ATP2 | #DCA6 | 03/05/90 | 534.00 | 3.60 |
| $ATP3 | #DCA10 | 03/05/90 | 423.00 | 3.60 |
| $ATP3 | #DCA7 | 03/05/90 | 1089.00 | 3.84 |
| $ATP3 | #DCA8 | 03/05/90 | 717.00 | 4.53 |
| $ATP3 | #DCA9 | 03/05/90 | 263.00 | 4.11 |
| $ATP4 | #DCA11 | 03/05/90 | 773.00 | 4.69 |
| $ATP4 | #DCA12 | 03/05/90 | 848.00 | 4.45 |
| $BSCTR33 | #BASS1 | 03/05/90 | 307.00 | 4.38 |
| $BSCTR33 | #BASS3 | 03/05/90 | 76.00 | 8.07 |
| $BSCTR33 | #CIRC1 | 03/05/90 | 65.00 | 3.24 |
| $BSCTR33 | #HUM1 | '03/05/90 | 367.00 | 3.51 |
| $BSCTR33 | #HUM2 | 03/05/90 | 100.00 | 3.72 |
| $BSCTR33 | #HUM3 | 03/05/90 | 170.00 | 4.43 |
| $BSCTR34 | #CHEM1 | 03/05/90 | 126.00 | 4.18 |
| $BSCTR36 | #PUBB1 | 03/05/90 | 786.00 | 3.67 |
| $BSCTR36 | #PUBB10 | 03/05/90 | 811.00 | 3.59 |
| $BSCTR36 | #PUBB11 | 03/05/90 | 786.00 | 3.71 |
| $BSCTR36 | #PUBB12 | 03/05/90 | 520.00 | 4.30 |
| $BSCTR36 | #PUBB2 | 03/05/90 | 302.00 | 4.76 |
| $BSCTR36 | #PUBB3 | 03/05/90 | 529.00 | 4.02 |
| $BSCTR36 | #PUBB4 | 03/05/90 | 333.00 | 3.83 |
| $BSCTR36 | #PUBB5 | 03/05/90 | 1286.00 | 3.84 |
| $BSCTR36 | #PUBB6 | 03/05/90 | 861.00 | 4.07 |

Response time average per transaction for all terminals: 3.80 seconds

Total number of transactions for all terminals: 32,644.00

NOTE: A transaction is equal to reading a command and outputting a response to the command.

playing the results as a merged retrieval set.

The three TRLN universities use two system monitor tools available on their systems to generate and analyze performance data. The system resource and performance monitor software MEASURE, available as part of the Tandem operating system software, collects detailed data on terminal response time, application resource use, and other items of interest (e.g., communication line activity, disk drive accesses). ENLIGHTEN, a third-party product from Software Professionals, Inc., can be used with MEASURE-created files to construct graphic representations of the data either online dynamically or in print format.

The TRLN libraries use MEASURE to collect response time data, to analyze software efficiency and pinpoint areas for improvement, and for capacity modeling and projection. Capacity modeling and efficiency analysis requires the collection and analysis of enormous volumes of data, usually on a great many hardware and software parameters simultaneously. Because of the volume, this kind of data is collected only periodically, and then through well-defined samples of system activity throughout the day (see below). On the other hand, data on the number of transactions on the system and the average response time for those transactions by port or terminal (Figure 3) and by time of day (Figure 4) is monitored constantly. The transaction response time reported by MEASURE is not the user-apparent response time. MEASURE calculates only the response time from the moment a command is received by the Tandem system to the moment a response is sent from the Tandem to the user device. It does not include communication time or display time.

This basic transaction and response time information is used in a variety of ways at the three universities: to prepare reports and track trends; to justify, plan, and budget equipment purchases; and to analyze the workload on and balance of the composite system. Each of the three universities reports the average number of daily transactions on its system and the average response time in the monthly *TRLN Project Status Report.* Despite different hardware configurations, the data provides some indication of the relative use of the three Tandem-based systems. For instance, in the fall of 1989, each TRLN institution experienced a sharp increase in the level of transactions, some by nearly forty percent. Other statistics, collected within the libraries, corroborated this increased use of library services. Circulation, for instance, increased nearly thirty percent at NCSU.

FIGURE 4.

**Summary Terminal Use And Response Time Report By Time of Day**

DATE OF THIS REPORT: 03/06/90 RUN TIME: 04:11:54 AM

| FROM-TIME | TO-TIME | AVERAGE RESPONSE | TOTAL TRANSACTIONS |
|---|---|---|---|
| 08:00:00 AM | 08:30:00 AM | 2.38 | 413.00 |
| 08:30:00 AM | 09:00:00 AM | 2.74 | 784.00 |
| 09:00:00 AM | 09:30:00 AM | 2.94 | 764.00 |
| 09:30:00 AM | 10:00:00 AM | 3.13 | 834.00 |
| 10:00:00 AM | 10:30:00 AM | 3.73 | 1143.00 |
| 10:30:00 AM | 11:00:00 AM | 4.29 | 1528.00 |
| 11:00:00 AM | 11:30:00 AM | 5.05 | 1665.00 |
| 11:30:00 AM | 12:00:00 PM | 7.15 | 1777.00 |
| 12:00:00 PM | 12:30:00 PM | 4.80 | 1649.00 |
| 12:30:00 PM | 01:00:00 PM | 3.71 | 1112.00 |
| 01:00:00 PM | 01:30:00 PM | 3.75 | 1127.00 |
| 01:30:00 PM | 02:00:00 PM | 3.83 | 1264.00 |
| 02:00:00 PM | 02:30:00 PM | 4.58 | 1580.00 |
| 02:30:00 PM | 03:00:00 PM | 4.27 | 1390.00 |
| 03:00:00 PM | 03:30:00 PM | 3.81 | 1192.00 |
| 03:30:00 PM | 04:00:00 PM | 3.98 | 1430.00 |
| 04:00:00 PM | 04:30:00 PM | 4.07 | 1227.00 |
| 04:30:00 PM | 05:00:00 PM | 4.18 | 1383.00 |
| 05:00:00 PM | 05:30:00 PM | 2.96 | 950.00 |
| 05:30:00 PM | 06:00:00 PM | 2.93 | 788.00 |
| 06:00:00 PM | 06:30:00 PM | 3.07 | 860.00 |
| 06:30:00 PM | 07:00:00 PM | 2.73 | 586.00 |
| 07:00:00 PM | 07:30:00 PM | 2.93 | 582.00 |
| 07:30:00 PM | 08:00:00 PM | 3.51 | 1034.00 |
| 08:00:00 PM | 08:30:00 PM | 3.41 | 1171.00 |
| 08:30:00 PM | 09:00:00 PM | 3.75 | 978.00 |
| 09:00:00 PM | 09:30:00 PM | 3.27 | 842.00 |
| 09:30:00 PM | 10:00:00 PM | 2.62 | 552.00 |
| 10:00:00 PM | 10:30:00 PM | 2.61 | 832.00 |
| 10:30:00 PM | 11:00:00 PM | 2.96 | 452.00 |
| 11:00:00 PM | 11:30:00 PM | 1.92 | 202.00 |

Response time average per transaction for all terminals: 3.80 seconds

Total number of transactions for all terminals: 32,644.00

NOTE: A transaction is equal to reading a command and outputting a response to the command.

On each campus, this basic transaction and response time information is reported to the library administration, library staff, and library users (e.g., Figure 5). It can be used to demonstrate progress or to warn of potential problems. In 1987, for instance, TRLN began to re-examine its software programs, rewriting many of them to increase the efficiency of the system. The resulting thirty-five percent increase in efficiency provided sufficient processing reserve to absorb the sharp increase in transaction levels in the fall of 1989 and still maintain "acceptable" response time. On the campus of UNC-CH, the data has been used to monitor the need for additional terminals in the House Undergraduate Library based upon the average number of transactions per day per port or terminal. As a result, in the past two years, the number of available terminals in that location has been doubled.

The records of terminal activity in a particular area also can be used to question the need for a terminal in areas of light or low use. For instance, at UNC-CH, a terminal in one department generated only ninety-nine commands in a two-week period during February 1990. On the basis of this data alone, it would seem that a terminal in this area was not justified. Such data should mandate a review of justifications for maintaining a terminal in little-used locations.

The daily statistics can be used to schedule batch jobs which contend with online functions for resources. Through a semester, TRLN staff monitor busy times and busy days of the week. As might be expected, activity declines sharply late Friday afternoon. Tuesdays, however, are as busy as or busier than Mondays. TRLN staff generally schedule extensive processing runs during low-use periods.

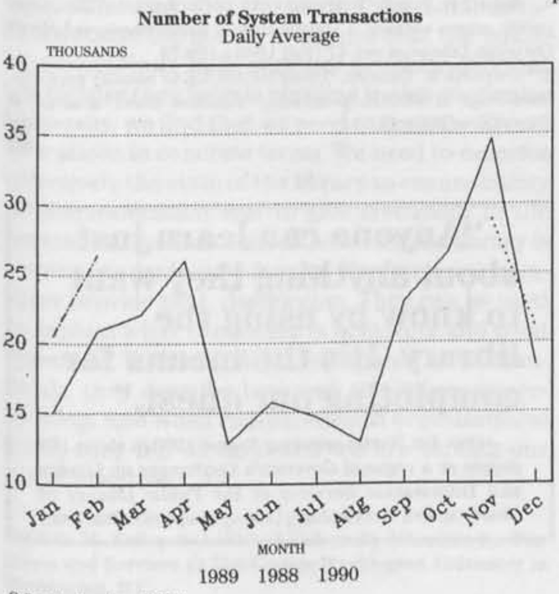
Terminal activity levels also can help identify physical conditions that lead to heavy use of terminals. In the cluster area of UNC-CH's Davis Library, for instance, one terminal is more heavily used than any other. Two characteristics distinguish this terminal: (1) it has more room for users to set materials down on either side of the terminal than do other terminals in the cluster, and (2) there is ample "personal" space because it is separated from other terminals, so that no other terminals (and hence no other users) are close by.

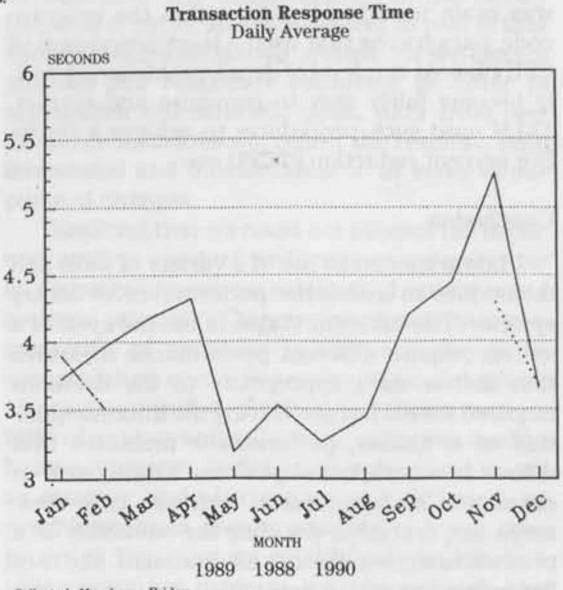
In planning and budgeting for the normal growth of systems, the pattern of current use can be called upon to project future needs. The increase in transaction levels needs to be closely monitored to determine the need for additional terminals and the need for additional processor capacity. This growth in transaction levels, coupled with the increase in data base coverage through retrospective conversion and new services such as the implementation of the TRLN Circulation Control Subsystem, must all be factored into planning the annual budget allocations and biennial budget proposals.

Dial access is one area where this data should be carefully monitored. A frequent question about remote access is: how much is enough? The simplest answer is that there is no single answer; it is always changing. The question should be how to monitor its use and to plan for its growth. Unfortunately, managers cannot know how many so-called "invisible users" exist, and these users as a rule do not inform managers about problems in accessing the online catalog. Even if users frequently encounter busy signals when they try to access the catalog, library managers may never find out that their remote access ports are constantly busy. In cases where the majority of remote access comes from links into existing campus networks, there may be no easy method to determine how often users are denied access to the catalog because its network slots are filled. (Interestingly, while in-house use increased dramatically at UNC-CH in fall 1989, dial access use showed no corresponding increase.)

MEASURE and ENLIGHTEN are used to monitor processor loads, memory use, disk activity, and other processes. With these tools, the systems

FIGURE 5.





manager can generate graphical displays showing the use of processor capacity and memory resources, and the distribution and timing of disk use among multiple disk drives. In a multi-processor configuration, the tools can show how the load is distributed over the processors (e.g., where the load is heaviest and where the load is lightest). All of this information is necessary to "balance" or "tune" the system load across the available processors and disks. System tuning directly impacts the efficiency of the system and the user-apparent response time. As hardware is added or new programs installed, the resource balance must be re-examined and the system tuned to preserve optimal use of resources.

BIS is implemented as multiple copies of a suite of programs. One advantage to this approach is to provide redundancy in the event of a system problem or crash. NCSU, for instance, runs six copies of the BIS software. If a problem occurs on one copy, it affects only one-sixth of the terminals. The terminals are distributed across the six systems based upon the load level, location, type of activity, and other factors. As new copies of the software are added to the system, the systems manager can use daily transaction data to redistribute terminals and maintain optimal transaction balance among the copies.

In addition to load balancing, these performance measurement tools can be used with the individual programs to gauge their relative efficiency and to identify where improvements can be made. In one project at TRLN, MEASURE was used to calculate the CPU time in milliseconds per transaction for each program.[5] A program could then be selected for closer scrutiny and MEASURE was again used to identify, within the program code, paragraphs that used a large percentage of CPU time. At this level of detail, problems generally become fairly easy to recognize and correct. TRLN used such procedures to achieve a thirty-five percent reduction in CPU use.

## Conclusion

Librarians make use of a variety of tools and techniques to assess the performance of library systems. The different stages in the life cycle of a system require different performance measures that deliver data appropriate to the decisions required for each stage. During the initial acquisition of a system, performance measures that deliver benchmark and peak load data, such as simulation and stopwatch response time measures, are crucial to deciding the suitability of a product to a given library's environment, and they figure importantly in developing the initial hardware configuration for installation. Throughout the production life of a system, system monitor data provides regular assessments of the capacity, response time, and utilization growth of the system. In particular, library managers closely monitor response time, because it remains the single most important determinant of user satisfaction. At the end of the life cycle, system monitor data forms a significant part of the management data required for functional design, performance specifications, and hardware configuration for migration to a new library system.

Performance measurement tools provide basic management data to support a variety of decision points during the production life of a system. Initial purchase, system tuning, terminal allocation, load balancing, optimal timing for resource-intensive processing, and system migration all depend upon comprehensive data concerning the kinds of activities and their resource demands on the system. It behooves library managers to develop an understanding of the nature and use of performance measures, to become familiar with different performance measures, and to ensure that their systems provide the data they require for system management decisions.

**References**

1. "Special Section: Measuring System Performance," *Information Technology and Libraries* 7 (June 1988): 173-97.
2. Jerry V. Caswell, "Performance evaluation of computerized library systems," in *Advances in Library Automation and Networking*, vol. 2, ed. Joe A. Hewitt (Greenwich, Conn.: JAI Press, 1988).
3. Clifford A. Lynch, "Response time measurement and performance analysis in public access information retrieval systems," *Information Technology and Libraries* 7 (June 1988): 177-83.
4. Robert N. Bland, "Evaluating the performance of the online public access catalog: a redefinition of basic measures," *North Carolina Libraries* vol. 47 (Fall 1989): 168-73.
5. Gwyneth M. Duncan, "Using MEASURE to identify performance bugs in COBOL programs," *Tandem Users' Journal* 9 (Nov./Dec. 1988): 13.

**"Anyone can learn just about anything they want to know by using the library. It's the means for completing our quest."**

—Gov. Jim Martin speaking August 1990 to about 150 people at a regional Governor's Conference on Library and Information Services at the Public Library of Charlotte and Mecklenburg County in uptown Charlotte.