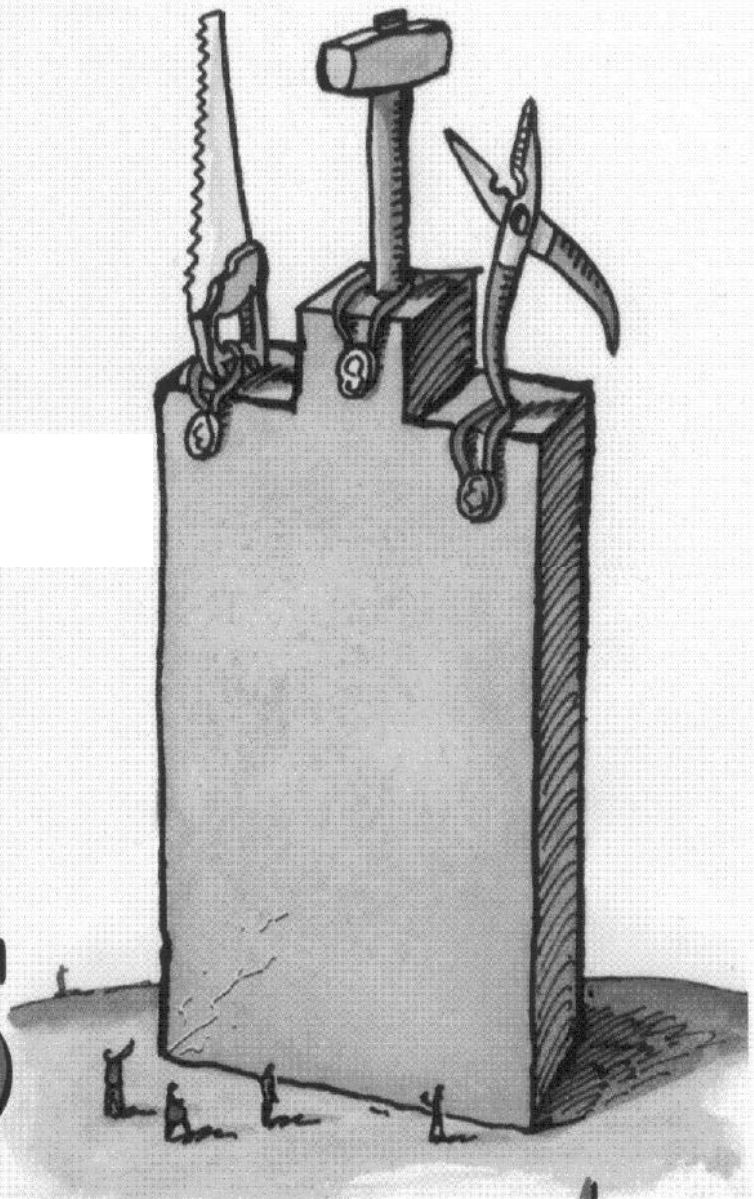
IEEE
Software
SEPTEMBER 1996



also:

tools
fair

software
process
impediments

seductive
interfaces

Tools Assessment

IEEE COMPUTER SOCIETY 50 YEARS OF SERVICE • 1946-1996

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Performance Testing a Large Finance Application

DAVID GROSSMAN and M. CATHERINE MCCABE, Office of Information Technology
CHRISTOPHER STATON, American Management Systems
BRET BAILEY, Sybase
OPHIR FRIEDER, George Mason University
DAVID C. ROBERTS, *Office of Information Technology*

The following case study shows how a simple prototype can be used to verify, before production, that a system will perform at an acceptable level under realistic conditions.

Developers have been performance testing large database applications for more than 20 years. Up to now, they have focused primarily on system-level testing of operating systems and database management systems. However, just because a machine, its operating system, and its database system are fast does not mean that an application using the database system will perform well. Design flaws in the application or in the tuning parameters specific to the application's environment often result in serious performance problems.

0740-7459/96/$05.00 © 1996 IEEE

TABLE 1
SAMPLE TRANSACTION WORKLOAD

Test type	Initial users	Concurrent users	Wait time	Number of increments	User increments for each test	Duration of test
Performance simulation	20	220	1-3 minutes	4	50 users every 12 minutes	60 minutes
Stress	20	220	1-3 seconds	4	50 users every 12 minutes	60 minutes

Most systems are tested for functionality but not for performance. Performance testing reduces the risk of poor performance at the time of application delivery. If a system does not perform well in production it quite often either never gets used, or it requires costly emergency repairs. Moreover, poor performance damages the reputation and credibility of the application and the project leader.

In this article, we present a case study in which we tested the performance of a large financial application. We used a popular test tool, TPNS (Teleprocessing Network Simulator), to simulate the application's performance. Our approach hinges upon the use of a simple prototype to verify that system performance falls within acceptable bounds. Once we developed the initial prototype, we began detailed tuning. This approach led to critical performance tuning improvement. We achieved dramatic improvement in our simulations, which two years of live operation have verified.

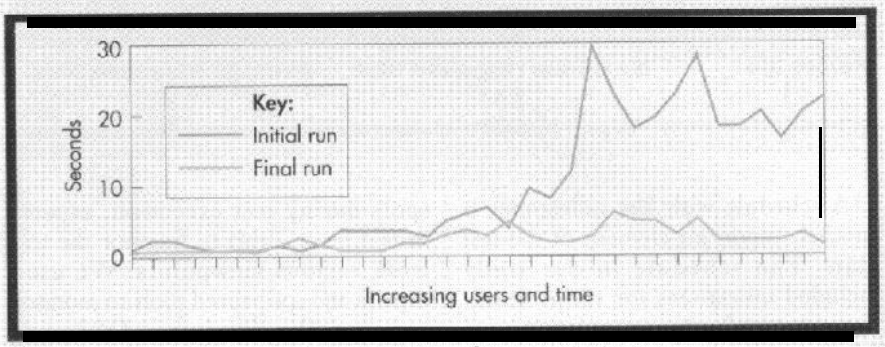


Figure 1. Average response times during performance testing.

THE APPLICATION

Our case study involves the first implementation of American Management System's Federal Financial System (FFS), a financial accounting application, in a Customer Information Control System (CICS) DB2 environment running on a large IBM mainframe. This application is written in Cobol and Assembler and contains approximately 1,000,000 lines of code. The underlying database contains more than 500 DB2 tables.

The total number of users anticipated for this application was 600, with 150 concurrent at peak performance. The required performance level, defined by the user, was that all on-line transactions must be completed in under three seconds with 150 concurrent users. We measure the time taken for a transaction as the interval from the pressing of the Enter key to the time the user receives the beginning of a response.

We implemented the following methodology:

- Identify user performance-related requirements, including the anticipated number of concurrent users, transaction response times, and details of the operating environment;
- develop a test plan containing transaction mixture, transaction workloads, data needed for the test, resources, schedule, and test procedures;
- choose a suitable test tool;
- implement the test, including execution and recording of the test activity and results; and
- make performance-related adjustments.

Further detail on our methodology and experiences is available elsewhere.[1,2]

The test plan. The transaction mix should model actual user activity during production usage. Table 1 shows a sample transaction workload. User activity logs from the old system gave us some idea of the activity level for the new system. For example, the old system processed 500 fiscal documents in a day. Thus, we assumed that the new system would need to process at least the same number.

The test data must consist of both database and application data. The database data, stored in the database prior to executing the test, contributed to appropriately sizing the database. The application data are input to the interactive data entry screens, through which simulated users actually update the database. A viable alternative approach is to automatically generate test data.[3] We used the data that the application developers converted from the legacy system as our baseline database data. To accurately model real-user activity, it is critical to avoid using the same application data repeatedly. The use of the same data may simulate the extensive use of buffers without ever addressing the I/O required by the application. To alleviate such problems, the test plan should describe internal reference tables of valid application-level values or combinations of related values. We identified valid values in our test by using SQL to extract data at randomly selected points from real database tables. However, this method required some manual work in identifying correct application-value combinations that we could use when developing our test cases.

The test plan must also include procedures that document the steps involved in executing a test, including

- backup and recovery procedures to return to the baseline data before and after tests,
- test execution procedures,
- test logging, and
- test results documentation.

While we were constructing the test, we found that our test log served as an invaluable resource for performance tuning, analyzing results, and resolving problems.

TABLE 2
RESULTS FOR PROTOTYPE TEST (TIMES GIVEN IN SECONDS)

Percentile	Maximum response time	Average response time
75	19.47	3.53
95	46.04	9.73

TABLE 3
RESULTS FOR PERFORMANCE TUNING TEST (TIMES GIVEN IN SECONDS)

Percentile	Maximum response time	Average response time
75	2.62	0.74
90	5.75	1.37

A schedule with key milestones is another important element of the test plan. The schedule in our study included milestones for script development, test database population, performance simulation, and a stress test. If our schedule had also contained milestones for developing backup and restore utilities, data population, familiarization with the test tool, and the analysis of results we would have been less hurried when developing the many utilities required to complete our testing.

Finally, the test plan should indicate who will monitor each test and the tools they will use.

Test execution. The key components to implementing the test plan are

- creating the scripts,
- running the test,
- monitoring the performance, and
- reviewing the results.

Since we based our entire method on simulation-driven performance testing of actual data, test script development is crucial. The TPNS tool provides an interactive data capture facility (IDC), which allows the script developer to record an actual execution of the application. When the recording session has stopped, TPNS translates the recording session into the Structured Translator Language (STL). We modified the output of IDC to include random table lookups and to create scripts that simulate a wide variety of application-specific data.

We carefully debugged the scripts by executing them from a driver that called each script sequentially. It took two days to debug the 45 scripts in our test and two weeks for the entire script creation process. To measure performance and identify possible trouble spots during test execution, experts should be available to monitor the test from various viewpoints. We used experts and specialized tools to monitor our test: We monitored DB2 with Insight, CICS with Omegamon, and MVS with Netview.[4-6]

Finally, we analyzed and reviewed the results. In addition to average response time for each minute of test execution, we tracked the minimum and maximum responses in that minute, and the number of times the simulated users pressed the Enter key and received a response. This *Number of Responses* is critical in verifying that user activity grows as the number of users grows, as opposed to one user obtaining good response time while other users are kept waiting. These waiting users would not be incorporated into the average response time for the interval. In our case study, the number of responses remained between 450 and 550 through most of the test. These numbers imply that there was no bottleneck prohibiting user activity during the tests. Therefore, the average response time results will be meaningful.

TEST RESULTS

The average response time for the initial prototype test and the final test are given in Figure 1. Table 2 presents the summary results for the initial prototype test for a 200-user performance simulation. The sharp rise in response time for more than 140 concurrent users shows the need for performance-related tuning. We did not anticipate this type of degradation prior to executing the test. By running with the default values, we quickly found a problem.

Considering our prototype result we selected appropriate system parame ters to alleviate the problem and re executed the test. Our use of extensi monitoring allowed us to quickly di count several possible problem area For instance, although each transactio uses a DB2 thread, monitoring showe that DB2 threads did not cause th wait. However, tracing activity in CIC identified a large number of short-on storage conditions. We identified th CICS TCLASS as the parameter tha determines how many tasks may exe cute concurrently in CICS. W changed this parameter and then re executed our test to determine whethe or not our configuration change had a effect on performance.

Table 3 provides the results, show ing that after performance tuning there was no serious performance degrada tion from zero users to 200 users.

RESULTS VALIDATION

After twenty-four months of liv operations, we validated the results o the test by comparing the actual execu tion statistics of the application with th estimates used for the test. First we val dated the number of concurrent users This was originally estimated at 60 users, with 150 concurrent. We execut ed our tests with up to 200 concurrent users. Figure 2 illustrates the actual daily average number of concurrent users over the three month period o September to November 1995. The overall average for this period is 191.4.

We then validated the average num ber of transactions, as Figure 3 shows The initial estimate given was 800 tran actions a day. However, to err on the side of caution, we used 1,250 as the basis for our test. The actual overall average for this period is 987 transactions a day.

Finally, as Figure 4 shows, we looked at the actual average response time for a typical two-week period from November 7-20, 1995. The overall

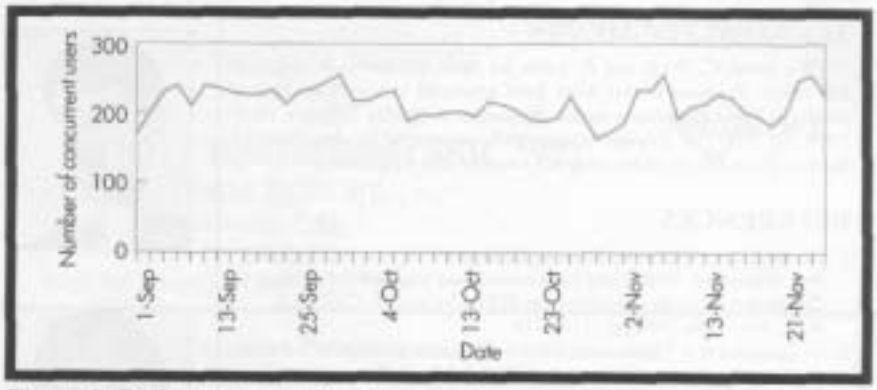


Figure 2. Daily average number of concurrent users.

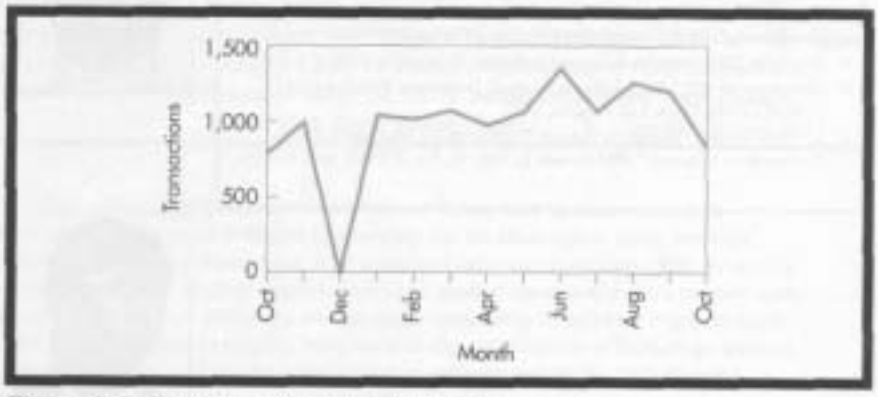


Figure 3. Daily average number of transactions.

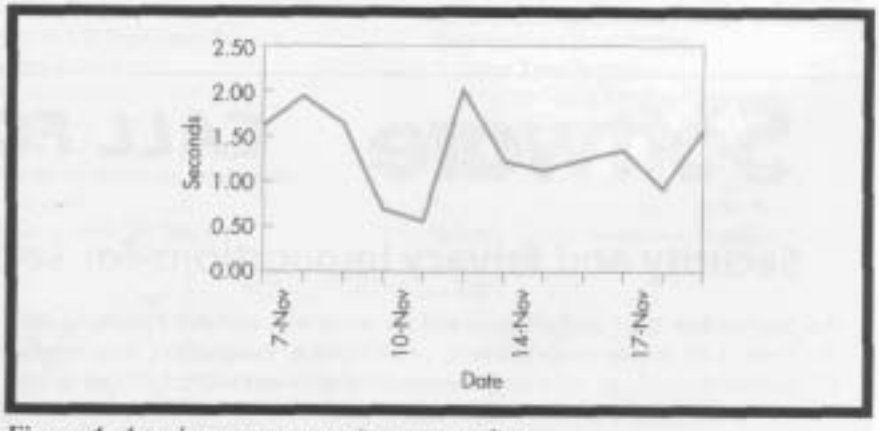


Figure 4. Actual average transaction response time.

average for this period is 1.28 seconds. Our test showed an average response time of 1.37 seconds for 90 percent of the transactions. This represents a 6 percent margin of error for our test.

As these results demonstrate, our test produced a fair approximation of the actual performance during the first 24 months of live operations. We predicted that the response time would be under 1.5 seconds and the actual response time was recently measured (over a two-week period) at 1.28 seconds; moreover, as expected, over six hundred users are using the system and concurrency levels have been as high as 210.

While our methodology has been successful in the mainframe environment, many of our current applications are developed in the client/server environment. Running TPNS to simulate hundreds of IBM 3270 terminal sessions is substantially easier than installing and configuring hundreds of PCs simply for testing. We have implemented some initial performance tests in the client/server environment with multiple workstations located at different sites. As we learn more about delivering applications in this environment, we will need to modify our existing methodology to accommodate it. ◆

David Grossman is a staff scientist in the Office of Information Technology. He is interested in database systems, information retrieval, parallel processing, and performance testing.

Grossman received a BS in computer science from Clemson University, an MS from American University, and a PhD in information technology from George Mason University.

M. Catherine McCabe is project manager of a data warehousing project and branch chief for the Office of Information Technology. Her interests include data warehousing technology, data mining, project management, and systems engineering methodologies, with over 10 years' experience in these areas.

McCabe received a BBA in management information systems from the University of Notre Dame and an MS in information systems technology from George Washington University.

C.J. Seaton is a principal with American Management Systems in Arlington, Va. He is currently the technical architect for the implementation of FFS at the United States Department of Agriculture. Previously, he worked on implementing a variety of financial systems at government and commercial clients. His technical interests include performance modeling and software development methodologies.

Seaton received a BS in computer technology from Purdue University and an MS in information systems from George Mason University.

ACKNOWLEDGMENTS

We thank C. Bock and A. Latts for their assistance throughout this effort. Portions of this work have appeared in the *Proceedings of the IEEE Third Symposium on the Assessment of Quality Software Tools*, 1994, pp. 125-135. Frieder was partially supported by the National Science Foundation under contract number IRI-93-577856.

REFERENCES

1. J.F. Blecha, Jr., "Testability Planning for Improved System Performance," *Proc. Globecom 88, IEEE Global Telecommunications Conf. and Exhibition—Communications for the Information Age*, IEEE Cat. No. 88, CH2535-3, IEEE, New York, 1988, pp. 1115–1119.
2. D. Grossman et al. "Application-Level Performance Testing: A Case Study of a Large Finance Application," *Proc. IEEE 3rd Symp. Assessment Quality Software Development Tools*, IEEE Computer Society Press, Los Alamitos, Calif., 1994, pp. 125–135.
3. J. Ortega et al. "Test-Pattern Generation Based on Reed-Muller Coefficients," *IEEE Trans. Computers*, Aug. 1991, pp. 968–980.
4. *Insight for DB2 Users Guide*, Legent Software, Pittsburgh, 1992.
5. *Omegamon for CICS/ESA Reference Manual*, Document Number 0CA53-3768-0, Candle Corp. Los Angeles, 1992.
6. L. Temoshenko. "Managing Session Performance Using the NetView Performance Monitor," *IBM Systems J.*, Vol. 31, No. 2, 1992, pp. 286–299.

Address questions about this article to Grossman or McCabe at the Office of Information Technology, Washington, D.C., 20505; dgrossm1@osf1.gmu.edu (Grossman) or 76236.3343@compuserve.com (McCabe).

Bret Bailey is a senior consultant with Sybase Professional Services. His interests include database systems, performance, and object-oriented application development, with over 16 years' experience in these areas.

Bailey received a BS in computer science from Indiana University of Pennsylvania in 1982 and is engaged in graduate studies at George Mason University.

Ophir Frieder joined the Florida Institute of Technology as the Harris Professor of Computer Science in 1996. Previously, he was a member of technical staff in the Applied Research Area of Bell Communications Research and on the faculty at George Mason University. Frieder has consulted with the Federal Bureau of Investigation, the Institute for Defense Analysis, IBM FSC/Loral Federal Systems, SAIC, and the Software Productivity Consortium. Frieder's research interests include parallel and distributed database and information retrieval systems and biological and medical data processing architectures.

David C. Roberts is a senior technologist in the Office of Information Technology and adjunct professor at George Washington University. His work deals with the technology issues associated with application services, including database systems and all aspects of application development.

Roberts received a BESc in electrical engineering from Johns Hopkins University, an MS in engineering from the Moore School of the University of Pennsylvania, and an MS in computer science from the University of Maryland.

IEEE **Software** ***CALL FOR ARTICLES***

Security and Privacy Implications for Software Development

The September 1997 special issue will focus on security and privacy concerns as they affect software development. Our goal is to advise programmers, practitioners, developers, and managers of the security implications of their development work; to encourage companies and researchers whose products have security implications to address those requirements, by giving examples of how others have done so and where to go for guidance; and to showcase positive achievements in developing secure applications.

Possible Topics:

- Security engineering to ensure a secure system designed from separate commercial components
- Integrating security tools and technologies with existing applications and in existing environments
- Determining the security of commercial products and systems
- Case studies in which tradeoffs between security and other attributes, such as performance and cost, have been balanced
- Examples of achieving security in innovative techologies, such as Web applications, pay-per-use applications, and electronic commerce

Guest Editors: **Charles P. Pfleeger**, Trusted Information Systems, 8000 Westpark Dr., Suite 600, McLean, VA 22102-3105, pfleeger@tis.com; **Deborah M. Cooper**, D.M. Cooper Co., PO Box 17753, Arlington, VA 22216, dmcooper@ix.netcom.com.

Submit a 200-500 word abstract to Pfleeger by **October 20, 1996**, and the complete article by **November 15**. Submit one electronic copy in ASCII, MSWord, or postscript format or eight hard copies.

For full author guidelines: Steve Woods, manuscript assistant, IEEE Computer Society, fax 1 (714) 821-4010, email swoods@computer.org