# *PERSPECTIVES ON THE SPEC SDET BENCHMARK*

STEVEN L. GAEDE
LONE EAGLE™ SYSTEMS INC.
JANUARY 1999

ABSTRACT

The year 1999 marks the twentieth anniversary of the benchmark that was adopted in 1991 as the SPEC Software Development Environment Throughput (SDET) benchmark. The fact that this benchmark has been in use for twenty years is remarkable, and indicates that it continues to fill a niche today.

SDET had its beginnings at Bell Laboratories in 1979, where it was used to evaluate whether the UNIX operating system would scale to large mainframe platforms. The benchmark was designed to apply a representative model workload to a system at increasing levels of concurrency in order to generate a graph of throughput vs. offered load. Given a set of curves representing the performance characteristics of a set of systems, a unit-less scaling factor could be calculated for comparing system scalability. Because of its use throughout the industry as a means for evaluating system performance, SPEC adopted SDET as a standard in 1991.

The most plausible explanation for SDET's continued use is that it provides measure of operating system and hardware platform performance. As such, there are ways in which SDET could be improved to provide a higher-quality measure of system performance. Although still useful as a 'system' benchmark, workloads today are significantly different than they were in 1979, and care must be taken when using SDET — or any standard benchmark — as a means for predicting performance under today's workloads.

## DEVELOPING A SCALING BENCHMARK

In the fall of 1979, Bell Laboratories was interested in determining whether the UNIX® operating system would scale to run on large platforms. In particular, if the same workload currently in use were moved UNIX running on a mainframe system, would it run with improved performance commensurate with the more powerful system? At the time, scalability was a serious issue, as nobody had tried running UNIX on hardware much larger than the DEC™ VAX™-11/780. In those days, UNIX was known for its use of linear table searches — and whether these would become disabling bottlenecks when supporting large numbers of users was totally unknown.

The means to explore these issues was with a benchmark designed for comparing the capacities of various implementations of the UNIX operating system. The first task in developing such a benchmark was to collect measurements of several days of activities on systems being used for software development. A model workload was developed by creating a set of commands and input data whose execution would match the real workload in terms of the frequency of commands used and in the resources consumed — only the model workload would be a miniature version that would be easier to use as a benchmark than trying to reproduce the multi-day, real workload. Tools developed at U. C. Berkeley (Gaede, 1981) were used to make multiple copies of the workload and execute it at increasing levels of concurrency. Workload segments were permuted in the model workload so that each level of concurrency would perform the same amount of work, however the



command segments would be executed in different sequence — avoiding any synchronization effects. Care was taken to duplicate the file hierarchy so that each concurrent script would access different files — hopefully reproducing the file access patterns of the real workload.

The result of running the benchmark is a curve that represents the performance characteristics of each system measured (Figure 1). With this graph of throughput versus offered load, one could take a ratio of the peak performance of a system under test to that of a reference platform and obtain a unit-less scaling factor. By observing the range of concurrency over which the system's performance stayed roughly level, a subjective measure of stability could be obtained. This was a useful benchmark for the developers of UNIX for large platforms because the measure of stability provided a sense of whether each release of the operating system was indeed an improvement. This was the first time that a representative model workload was applied at increasing levels of concurrency to obtain a scaling factor in this manner (Gaede, 1982). Since this benchmark was to have been short-lived (after all, it had answered the question for which it had been designed!) it had never been named and, as a result it became known within Bell Labs as the "Gaede Benchmark."

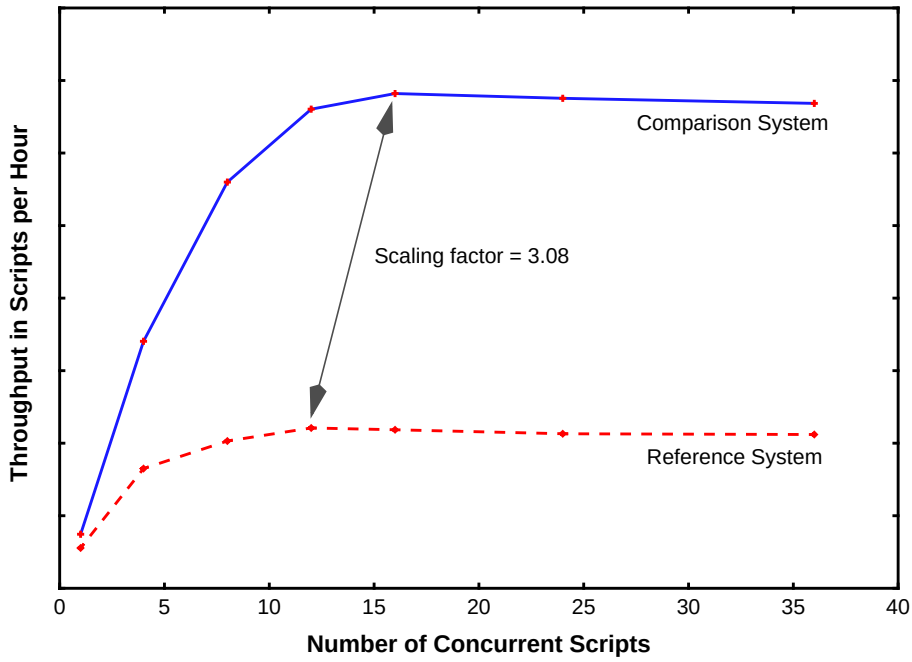


Figure 1 The ratio of peak throughputs of the comparison system to that of the reference system yields a unit-less scaling factor.

## SPEC Releases the SDET Benchmark

It was quite a surprise when, in 1991, the Standard Performance Evaluation Corporation (SPEC™) decided to standardize a revised version of the Gaede Benchmark — the Software Development Environment Throughput (SDET) benchmark. This was a matter of concern because, in order to be a good predictor of performance under a real workload, a benchmark *must be representative of that workload*. And certainly few would be using this metric because their workload is well-represented by this job mix from 1979!

Who has been using this benchmark for two decades, and for what have they been using it? During the 1980's, the benchmark had taken on a life of its own. Its use had continued within Bell Labs to help in procurement decisions, and in evaluating the performance of successive releases of the UNIX operating system (Feder, 1984). It had been used outside of Bell Labs by a wide variety of



vendors as a way to tune their systems and to present performance comparisons to customers. It turns out that the benchmark had become known as a 'system' benchmark, and was used in measuring the efficiency of the operating system and underlying hardware. When the SDET and KENBUS (McDonnell, 1990) benchmarks were released by SPEC, they were the only two standard techniques for measuring the response to multiple, concurrent processes all competing for resources through the operating system.

Even today, other standard benchmarks focus on very specific areas of performance. They provide excellent metrics for comparing network file servers, transaction processing systems, decision support systems, and Web servers. The SPECint™ and SPECfp™ benchmarks use primarily CPU-bound programs to evaluate processor performance. SPECrate™ runs multiple concurrent copies of the CPU performance benchmarks — however because the workload is homogeneous and demands little of the operating system, they give little indication a system's ability to scale up with a heterogeneous workload.

## A System-Level Benchmark

The SDET benchmark has been used for so many years because it satisfies the need for a system-level benchmark. It is relatively simple to run. In contrast to transaction-based benchmarks, it requires no remote terminal emulators, no network set-up, and a modest amount of disk space. In recognition of its use for system regression testing, SPEC's 1991 release of SDET began with the Gaede Benchmark from Bell Labs and modified it to be more relevant and more easily executed. SPEC changed the AT&T 3B processor cross-compilations to be regular native C compilations, changed the input files slightly, and gave it a framework to ease the task of benchmark execution.

Over the years since SDET's first release, it has become clear that the benchmark's primary use has not been in evaluating how many 1979 work units a system performs per time, but how well a system responds under stress, and whether a set of operating system modifications result in performance improvements.

If SDET is truly a system benchmark, then it is less important to measure a workload that includes both the efficiency of system utilities and of the operating system/hardware configuration, and more important to measure only the latter. As a system benchmark, SDET should be holding constant the utilities that make up the workload — the *cc*, *ls*, and *nroff* commands, for example — and allow the operating system and hardware to be the only variables. The issue is not whether one vendor's *cc* is faster than another's — in fact, it may be desirable to have a slow C compiler that generates efficient code than a fast one that generates un-optimized code. The issue is not the speed of system commands, but how well the system handles the load they impose.

## Focusing on System Performance

Reducing the number of variables measured by the benchmark would allow it to be more universally applied to measuring system performance. One suggestion that has been made for how to accomplish this is to require use of GNU software rather than the native system commands. This would level the playing field by ensuring that the same software — except for the operating system — is executed on every system measured with SDET. And because it has been ported to almost every computing platform imaginable, standardizing on the GNU software sets the stage for cross-operating system comparisons — even, for example, between various implementations of UNIX and the Microsoft™ Windows NT™ operating systems.

Another way in which SDET could be made more useful as a measure of scalability is to return to the intent of the original benchmark — to compute a scaling factor for comparing interactive system capacities, and to give an indication of a system's stability under heavy load. Rather than reporting results in terms of jobs per hour, SDET should present peak throughput as a scaling factor



of some reference platform. If the reference platform were the same as that used for the current SPECint and SPECfp measures, direct comparisons could be made between processor performance and its ability to support an operating system and timesharing workload.

Finally, the notion of stability should be quantified. By determining the points at which the throughput climbs, and then drops, through some percentage of the peak, the distance between these points could be used to compute a stability measure. For well-tuned systems, the point at which the system performance drops below some percentage of the peak should be significantly beyond the peak. For systems exhibiting non-ideal scheduling behaviors, this point should be closer to the peak, indicating less of a stable system.

## EVALUATING SYSTEMS TODAY

In the twenty years since its initial development, the SDET benchmark has gone from being an accurate predictor of how well a timesharing workload would run on a particular platform to filling a niche as an operating system performance and regression testing tool. During the same period, the nature of computing has changed significantly — resulting in different requirements for benchmarks like SDET. Timesharing has largely given way to networked client-server and multi-tier architectures (Figure 2) with vastly different workloads than those seen in 1979.

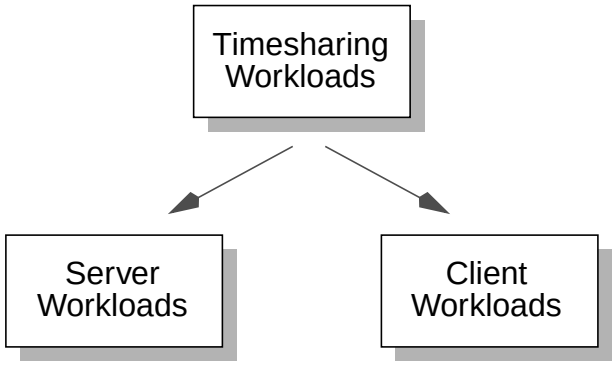


Figure 2 Timesharing has given way to client-server computing, and the what was one workload on one system has not split into server and client workloads.

### MEASURING SERVER PERFORMANCE

Today's servers most often provide a single function on behalf of a large number of clients — for example database, network file, application, or Web services. While they are timesharing workloads in the sense that they concurrently serve many clients, their behavior is quite different from the workload represented in the SDET benchmark. Whereas SDET models a workload that is intensive in process creation and parallel execution of single-threaded processes, servers today host long-lived, multi-threaded single processes, placing a different workload on the operating system and underlying hardware platform.

There are many standard benchmarks that can help organizations wishing to compare server performance for procurement and tuning efforts. For example, the SPEC SFS benchmark measures network file system performance, SPECweb96 evaluates Web server performance, and a collection of benchmarks from the Transaction Processing Performance Council (TPC) can be used to evaluate different aspects of database performance.



For those services not addressed by the standard benchmarks, and for servers which provide more than one function concurrently, there is no substitute for measuring the target system using a representative model workload. The SDET approach of applying a representative model workload at increasing levels of intensity to generate a curve of throughput vs. offered load is one that can be used regardless of the workload being applied.

### MEASURING CLIENT PERFORMANCE

Workstations and personal computers have mostly replaced the functions of yesterday's timesharing systems in the sense that they perform activities under the direct command of the user. What the user is concerned about is often not the system's throughput, but its *responsiveness* — often to a single input such as a mouse click.

These workloads also vary significantly from those of SDET and KENBUS. For example, today's software developers use document processing systems rather than *nroff*, and computer-aided software engineering tools rather than just *cc* and *make* for managing the development process. These tools present graphical user interfaces through the window system; they store files onto network file servers, and prepare reports and documents in PostScript format. Many of these technologies had not even been invented at the time when SDET and KENBUS were designed.

There are several reasons why there has not been a successor to SDET and KENBUS for measuring performance under workstation-specific workloads:

— It is difficult to characterize these workloads because designing a "standard" workload would require measuring user activities for a broad set of applications over many users and many days. Whereas AT&T was generous in its donation of the representative SDET workload to SPEC, no organization has since contributed characterizations of their workloads.

— It is a time-consuming process to set up tests which reproduce mouse and keyboard events in a platform-independent manner, and measuring response times is a particular challenge in systems where graphics engine performance must be factored into the equation.

— Finally, it is difficult to configure the systems for such a test. Any commercial product — a spreadsheet package, for example — would have to be available on all platforms to be measured, and purchasing these products adds to the costs of running a benchmark. Additionally, since most workstations today operate in a client-server environment, the configuration would have to include the server systems that are typically used in the course of the standard workload.

## CONCLUSION

The SPEC Software Development Multi-Tasking (SDM) benchmark suite that includes the SDET and KENBUS benchmarks has received sustained interest and use over a long period of time. One of the reasons for SDET's continued use is that it is far more easy to configure and execute than a benchmark which takes into account all of the complex applications and network dependencies of today's network computing environments. SDET is a good measure of a system's response to increasing, concurrent demands for system services, and gives an indication of the true scalability of multiprocessor configurations. Modifying SDET with a platform-independent workload could further enhance its usefulness in this arena, an would enable a more broad set of platform comparisons than ever before.

There is a large number of standard benchmarks available today which can be used to compare systems running very specific workloads, and yet there remains a need to measure the diverse activities of client-server network computing. If such a new, comprehensive benchmark were to be developed, one of the first steps would be to characterize and model workstation workloads so that



performance could be measured on a wide variety of workstations and personal computers. With well-characterized workloads, harnesses could be created to re-create mouse and keyboard input, network traffic, and to isolate variables that are not relevant to the measurement question at hand.

As the performance community moves towards providing more narrowly-focused benchmarks, the key factor to remember when using standard benchmarks to predict future performance is how well they represent the workload that is to be executed. Often the standard benchmarks provide extremely valuable insights. And sometimes the best choice to model the specific workload that is of interest and to evaluate target systems with benchmarks based on representative model workloads.

## REFERENCES


Feder, Jerry, The Evolution of UNIX System Performance, AT&T Bell Laboratories Technical Journal 63(8) (October 1984), 1791-1814.

Gaede, Steven L., Tools for Research in Computer Workload Characterization and Modeling, in: Ferrari, D., and Spadoni, M. (eds.), Experimental Computer Performance Evaluation (North-Holland, 1981), 235-247.

Gaede, Steven L., A Scaling Technique for Comparing Interactive System Capacities, Proceedings Computer Measurement Group XIII (December 1982), 62-67.

McDonnell, Ken, An Introduction to the KENBUS Benchmark, in: The SPEC SDM Benchmark Suite, Standard Performance Evaluation Corporation (SPEC) (December 1990).

Information on the Standard Performance Evaluation Corporation (SPEC) and their benchmarks may be obtained by visiting the Web site: *http://www.specbench.org/*.



The author is available through Lone Eagle Systems Inc., 3023 Fourth Street, Boulder Colorado 80304 USA, by telephone: +1 (303) 444-9114, by electronic mail: *gaede@loneagle.com*, and through the Web site: *http://www.loneagle.com*.


## COPYRIGHT NOTICE