SYSTEMS PERFORMANCE EVALUATION COOPERATIVE

spec newsletter

Volume 2, Issue 4 Fall 1990

INSIDE THIS ISSUE

SPECIAL NOTE

This issue of the spec newsletter also includes the quarterly benchmark results. This format will be used until Release 2.0 is published.

RESULTS SUMMARIES

System Level Benchmarking

Krishna Dronamraju
Asif Naseem
Sivaram (Ram) Chelluri

AT&T Bell Laboratories
Naperville, Illinois

Release 1 of the SPEC Benchmark Suite has given the industry a more objective and verifiable speed rating (SPECmark) than the old "mips" ratings, which has been discredited through marketing exaggeration and hype. SPEC Release 1.1 introduced the SPECthruput metric, which is an enhanced SPECmark, prorated according to the number of CPUs in a system.[1] Undoubtedly both these metrics provide a level playing field for comparing CPU speeds of different systems, as the workload of the benchmark suite on which they are defined is the same. This leads to the next issue of setting up additional methodologies for measuring system level performance.

In this article, an attempt is made to distinguish between the various throughput methodologies that are in vogue. We discuss the pros and cons of speed versus throughput methodologies, as well as fixed multi-threading versus peak-throughput methodologies, in running a wide variety of benchmarks with significant I/O. This discussion is in context of SPEC Release 2, for which a powerful array of good and proven benchmarks is being considered by the SPEC Steering Committee.

Speed Versus Throughput

The Dhrystone, Whetstone, Linpack, and SPEC Release 1 benchmarks could all be categorized as speed benchmarks. They all are single-threaded, CPU-intensive (with negligible I/O content) benchmarks. Performance on these benchmarks can be greatly affected by how:

- compilers are used
- the data/instruction caches are configured
- floating point is performed
- integer arithmetic (including single and double precision arithmetic) is performed
- logic is performed

The SPEC benchmarks are a decided improvement over the other speed metrics because they represent varied "real" application workloads. In addition, the SPEC suite provided a much larger (156,000 lines of code) and more complex workload, thereby making it difficult to write optimizing compilers for the entire SPEC Release 1 suite as was previously done for the Dhrystone benchmarks. The large, complex SPEC workload also stresses the memory subsystem in a more realistic manner.

Continued on page 4

Running SPEC Release 1 Benchmarks

By George Fichter
Apollo Division of Hewlett-Packard Company

This article will describe the form in which the SPEC benchmarks are distributed and briefly explain how to run them.

Step One: Loading the Tape

Tapes shipped by Waterside Associates (SPEC's organization arm) are in QIC-24 format on a 600 foot cartridge tape, and they are written in *tar* format. To install the entire suite, all that is required is an empty directory, which we will call the spec home directory, on a disk with about 25 megabytes of disk space. The tar command (*tar xv*/ will then read the tape and create all the necessary subdirectories.

SPEC Directory Structure

The SPEC home directory contains some text files with hints on running the benchmarks. There are two important files which require your immediate attention: 1) README explains in detail how to run the benchmarks; 2) RUNRULES describes the conditions under which the benchmarks must be run.

Several subdirectories are included on the tape, and others are created during the life of the benchmarks. Only two require your attention:

binsrc contains source programs for the tools used by the SPEC benchmarks. These tools must be installed before running any of the benchmarks.

benchspec is the directory containing the individual benchmarks. This is where the benchmarks are run and the results are generated.

Step Two: Set up SPEC System Variables

Whenever you work on SPEC benchmarks, it is necessary to set certain system variables specific to SPEC. This is done by executing

source *cshrc* (for users of C-shell)

or

. *shrc* (for Bourne shell users)

Step Three: Installing the Tools

The next step is to create the SPEC tools. Starting in the SPEC home directory, the tools will be automatically compiled and installed with the following command:

make IDENT=vendor VERSION=yyy bindir

The tools thus installed will reside in a directory called ***bin*** under the SPEC home directory.

About M.vendor Files

Most vendors have created "wrapper" files to assure that benchmarks are run correctly on their machines. Currently the following M.vendor files exist:

M.apollo	M.att	M.dec_risc	M.dgc
M.hp	M.mips	M.motMP	M.motorola
M.new	M.solbourne	M.sun	M.sun.sc0
M.sun3.4XX	M.vax	M.880	

When running the benchmarks, select the "vendor" file which is most appropriate for your machine. This might require some trial and error.

Step Four: Run the Benchmarks

Once you complete the preceding steps, you are ready to run the benchmarks. The easiest way to do this is with the command: make IDENT=vendor VERSION=1.0

This command will execute all the benchmarks, compiling them if necessary, and place the results in the tests.results directory.

Results will be found in a directory called **tests.results**.

It will take several hours to compile and run all the benchmarks, so be patient. On the reference machine, a Vax 11/780, it takes 21 hours just to run them.

Step Five: Running Individual Benchmarks

For debugging purposes, another method of running the benchmarks is quite convenient. This is done by moving to the benchmark's directory and using the make command.

For example, if you want to run spice on an HP machine, execute: ***cd benchspec/013.spice2g6***
make -f M.hp

Timing results will be displayed on the screen and saved in a file called ***result/time.out.***

Benchmarks in the SPEC suite are available to the general public and can be ordered from Waterside Associates. An order form is included in each SPEC newsletter.

George Fichter is in the performance analysis group at the Apollo Division of Hewlett-Packard Company.

System Level Benchmarking

Continued from front page

After defining the SPECmark metric, SPEC developed a new methodology in an attempt to measure performance for multiprocessor systems. The result was the SPECthruput methodology, which is defined as a fixed load, multi-threaded metric calculated as a geometric mean of elapsed time ratios.

Here the multi-threading is fixed arbitrarily as two copies of a SPEC benchmark per CPU. Still both the SPECmark and SPECthruput answer the question, "How fast can you run one?" The SPECthruput methodology is still based on a CPU-intensive fixed workload and does not cover all the dimensions of a true throughput benchmark. The question of "how many can I run at once?" still needs to be addressed in a more comprehensive manner. SPEC is endeavoring to solve this in a judicious manner in Release 2 and future releases.

What is System Level Performance?

CPU performance, floating point speed, compiler efficiency, cache size and speed (evaluated by the speed metrics) do form a subset in a field of components that constitute system performance. System performance more broadly includes all of the following:

- Operating system (system calls & how fast they are executed)
- Memory (bandwidth) and management
- System bus (bandwidth)
- I/O subsystem (I/O bus bandwidth, latency, and throughput)
- Various I/O control mechanisms (IOP's)
- Disk/tty/network subsystems.

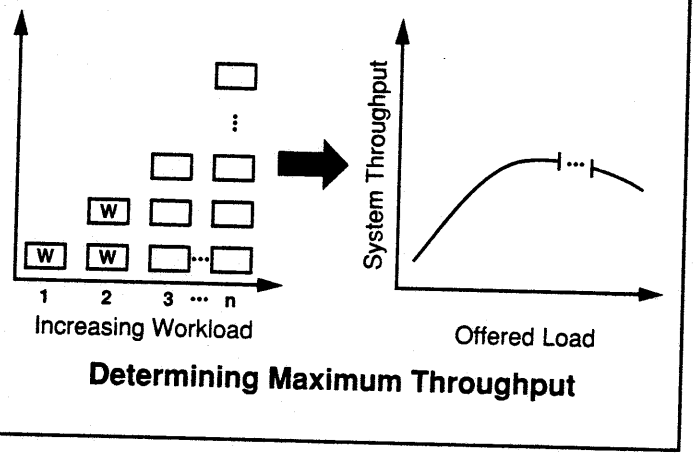
When considering a system throughput methodology, both peak throughput and average response time are good measures of system performance.[2] Peak throughput is of interest to anyone who is attempting to buy a system and wants to find out how much work it can handle.

Response time is of interest to a user of the system. Both of these metrics are inter-dependent (inverse relation, i.e., more throughput results in less response time in a macro sense).

In the view of a performance analyst/evaluator, both of these approaches have equal significance, as they directly answer the question, "How many workloads can you run at once?" The difference is in the experimentation. The throughput measurement involves simple benchmark design with an internal driver capable of running multi-threads of the benchmark. The response time measurement involves two systems (the system under test and an external remote terminal emulator driver). Here the discussion is focused on the merits of throughput methods only.

The delicate difference between a fixed load throughput metric (as defined by the first SPEC Release 1 throughput methodology) and a peak throughput metric (generated through varying the workload until a maximum value is obtained) is that the former measures a single point in a linear universe, and the latter gives all the points in the universe and includes the best point in the same universe, as shown in figure 1.[2] Here increasing the benchmark load increases the stress on the system progressively until all the system resources are fully utilized and a bottleneck occurs, either with CPU capacity, memory or I/O capabilities.[2] Such a workload, which gradually and uniformly stresses the system, needs to be defined.

Fig. 1: Throughput as a Performance Index



Determining Maximum Throughput

A Throughput Technique

In general, in an ideal throughput benchmark, the gradual increase of workload on the system is achieved by increasing the number of concurrent copies of the benchmark being executed. Here each copy of the benchmark workload, called a "script," should stress all the layers of the system as described above, balancing the utilization of system resources. For the purposes of this article, a script under execution is referred to as a "process."

1. Randomization of Workload Scripts

Running concurrent workloads may lead to misleading results or conclusions. If multiple identical copies of a benchmark are executed, each one of them will obviously access the same resource at approximately the same time. This will not yield a realistic picture of the system performance. So, ideally each instance of the benchmark should be executing a different workload. At the same time the overall content of the workload should be the same. Thus, ideal workload scripts should consist of a random mixture of a set of basic workload modules.

Continued on page 5

For example, in the case of the well-known TP1 benchmark, though each teller is executing identical transactions, the randomness is achieved through the random account and branch numbers accessed by each teller. In the case of the SPEC throughput benchmark, sdet, a candidate for Release 2, the randomness is achieved through mixing 21 basic modules in a random manner to constitute a workload script.

2. Throughput Metrics

Throughput is broadly defined as the total amount of work done in a given time. The work done can be enumerated in several ways.

a. Quantifying the Work Done

(1) Total number of instructions or commands executed by the scripts (as in the case of the *sdet* benchmark, a SPEC Release 2 candidate).

(2) The number of scripts (as done by the *kenbus1* benchmark, a SPEC Release 2 candidate).

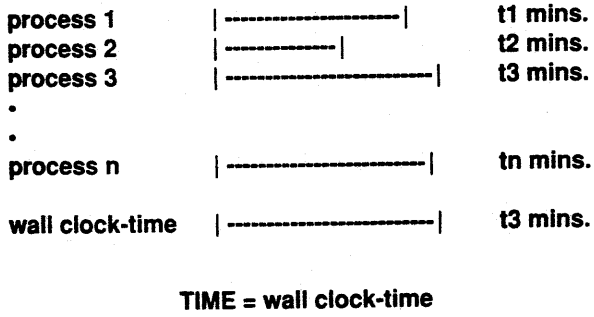
(3) The number of operations completed, if the workload consists of operations performed and each operation is either identical or the total operations executed by each script contain the same mix of different size operations (as in the case of *iobenchp*, another SPEC Release 2 candidate).

b. Timing the Benchmark Runs

The way a benchmark is timed may cover or uncover certain inherent problems in the systems. Here the three timing mechanisms in vogue are mentioned.

(1) Wall-clock timing. This is the benchmark's elapsed time.

Fig. 2: Wall Clock-timing–Elapsed Time

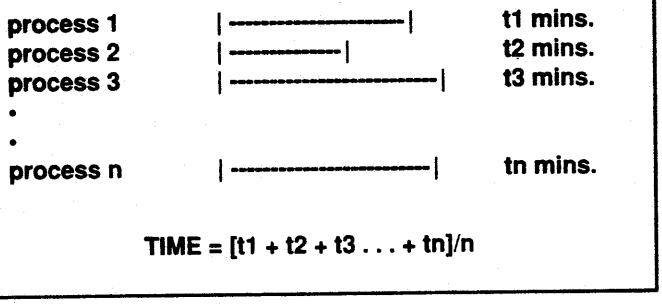


Assuming that all the scripts are synchronized to start at the same time, this method clearly shows a tail-end effect. Here the tail-end effect is defined as present when discrepancies exist in the run times of different copies of the benchmark when they are executed concurrently, after they are all started at the same time. These discrepancies in run time do occur due to scheduler behavior, context-switching, or when considerable number of tasks are waiting for the same resource. The performance data from this throughput metric, affected by this tail effect, will be less than the true optimal throughput of the system.

(2) Timing Each Process (script-run):

Another alternative would be to measure the individual process time and to consider the AVERAGE time.

Fig. 3: Wall Clock-timing–Average Time



In this method, the accuracy is improved but still the final throughput metric does not reflect the optimal throughput the system is capable of, due to the same tail effect. This measurement definitely exposes scheduler problems when compared with the elapsed time above. (If the wall clock time for a 10 user run is approximately equal to the sum of the 10 individual timings, then certainly we are dealing with a primitive scheduler!)

Regarding tail effects, the optimistic way is to measure the minimum time, and the pessimistic way is taking the wall clock-time (maximum) time. The first one tends to give higher throughput result, and the second one gives the least. The average value gives the in-between result, nearer to the system capability. On the other hand, averaging may hide the scheduler problems.

(3) Fixed Time Runs

In this run all the processes are started at a synchronized instant, and all are terminated after the lapse of a predetermined time, say 15 minutes or 30 minutes. The amount of work done by each script is measured and conveyed to the parent process.

Alternately, as recommended by the Transaction Processing Performance Council (TPC), the timing (or workload count) is started after a steady state of resource consumption rate is achieved in a benchmark run and stopped after a predetermined interval without actually stopping the processes (i.e., a window snap shot).

Continued on page 6

System Level Benchmarking

Continued from page 5

Fig. 4: Workload Count

```
process 1  |----------|  t1 mins.  Work done = W1
process 2  |----------|  t2 mins.  Work done = W2
process 3  |----------|  t3 mins.  Work done = W3
.
.
process n  |----------|  tn mins.  Work done = Wn

TOTAL WORK DONE = W1 + W2 + W3 . . . + Wn
```

Ideally, this yields the maximum system throughput; the method, however, may be suitable only for operation oriented workloads, where the time taken for completion of a single operation is negligible when compared to the total run time chosen.

Comments

Whatever method is chosen for quantifying work done and measuring time, the tail effects can be made negligible, if sufficiently longer-running workloads (longer compared to the tail-end differences) are considered.

With clearly defined workload and time measurements, a complete throughput curve (see figure 5), speaks volumes about the performance of a system. Sometimes comparing one system with another using this two dimensional metric may not be sufficient, unless a third dimension of price/performance is added.

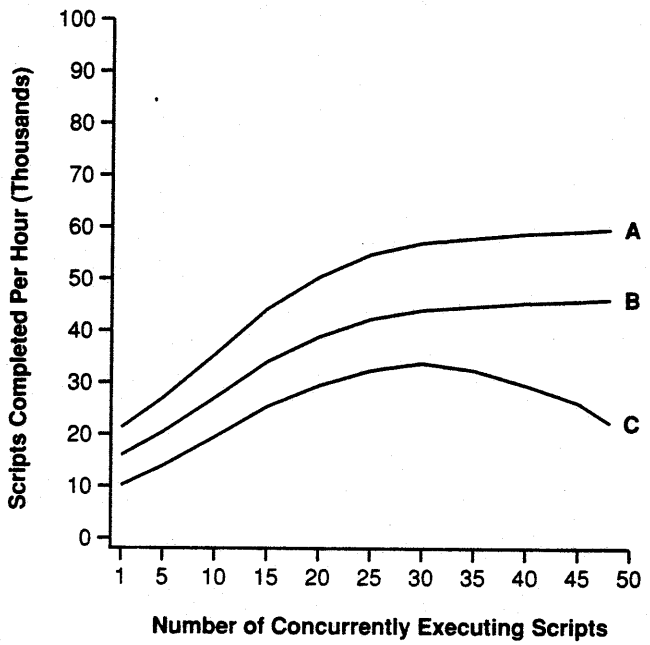
Typical Analysis of System Throughput

The general system behavior is described by the complete throughput curve from a balanced workload throughput benchmark. The following analysis can be deduced from the shape of the curve (shown in Figure 2, where curves B and C are taken from the same machine, before and after tuning, and curve A is derived from a different machine).

For an ideal machine, the curve should remain flat after reaching a maximum value. But in reality, the curve may be flat for a short interval and then trail off with increase in the "offered" workload. How fast the curve rises indicates good response time, as shown in curve A of Figure 5.

A slow-rising curve (as in curve C) indicates a poorly tuned system. This could be caused by the disk subsystem, kernel disk buffer management routines, memory management, or scheduling policies.

Fig. 5: Throughput Improvements for System



In comparing throughput curves of different systems:

- The system with a high one script value (curve A) would indicate better response time.
- The system with a higher throughput value (curve A) would indicate better multitasking capability. The peak throughput is the most obvious metric, and it indicates the ideal match of work and system behavior for the particular configuration being measured.
- The width of the flat area around and after the peak value indicates how well the system responds to demands beyond its optimal capacity. It shows whether the system gives good and consistent response to an increasingly stressful workload, as indicated in the flatness of curve B.
- A steep drop in throughput after reaching the peak indicates that the system has reached a bottleneck; for example, this applies to the scheduler operation or interrupt service times on other sources, as shown in curve C.
- Depending on where the system bottleneck is, the peak throughput may improve for a given configuration by adding more CPU power or overlapped disk, I/O, or memory. The system activity data can be used to identify changes required for achieving the above optimum.

Continued on page 7

Conclusion

System throughput benchmarks, in order to compare a wide range of systems, must have scaled workloads. By using throughput curves as the measure, the scaling is automatic. Each system will run a workload which is appropriate for its peak throughput capability. Comparison of systems is accomplished by analyzing the throughput curve for each system.

Note:

1) Valid comparisons between two systems' throughput capability cannot be accomplished without considering the price of the configurations measured.

2) Also, it should not be interpreted that there is a direct correlation between the number of scripts and the actual number of users supported by the system under test.

Footnotes:

1. Greenfield, Mike et al 1990: "SPEC Adopts New Methodology for Measuring System Throughput," SPEC Newsletter, Spring 1990.

2. Gaede, S.L. 1982: "A Scaling Technique for Comparing Interactive System Capacities," Proceedings of the Computer Measurement Group conference, Dec. 14-17, 1982.

3. Hennessy, J.L., and Patterson, D.A. 1990: Computer Architecture: A Quantitative Approach, Morgan Kauffman, 2929 Campus Drive, San Mateo, CA 94403.

4. TPC Benchmark™ B Specification, 1990: contact Waterside Associates, Fremont, CA for copies.

Krishna Dronamraju, Asif Naseem, and Sivaram (Ram) Chelluri are engineering managers in the Performance and Competitive Analysis Group at AT&T Bell Laboratories in Naperville, Illinois. ***(Contributed Article)***

SPEC Member Companies

Arix Corp.

AT&T

Bull S.A.

Compaq Computer Corp.

Control Data Corporation

Data General Corporation

Digital Equipment Corporation

E.I. DuPont De Nemours & Co., Inc.

Fujitsu Ltd.

Hewlett-Packard

IBM

Intel Corporation

Intergraph Corp.

MIPS Computer Systems

Motorola

NCR Corporation

Prime Computer

Siemens AG

Silicon Graphics

Solbourne Computer

Stardent Computer

Sun Microsystems

Unisys Corporation