# Benchmark Suites to Measure Computer Performance

Hernâni da Silva Costa Correia

*Departamento de Informática, Universidade do Minho*
*4710 Braga, Portugal*
*hernanic@aeiou.pt*

**Abstract**. The purpose of this work is to show one particular technique to measure and compare the performance of computers – benchmarking. First an overview is given of what is relevant in a bench rehearsal, and then some of the main benchmarking techniques are presented. A reference is made to the work carried out by the main international benchmarking organisms, focusing on the SPEC consortium and on its CPU2000 benchmark suite.

## 1 Introduction

Assessing computer performance is something that everyone that deals with computers, in some point wanted to do. But how this is done, by whom and under what circumstances this assessment is done are subjects that are going to be explored in this communication.

By nature, everyone wants the best and the fastest machine but how can I be sure that I have really the fastest machine? From the simple user's point of view a computer can be faster if it runs a computer program in a shorter period of time (*response time*), while from the point of view of a computer manager, one computer is faster than any other if it completes more tasks in one defined period of time (*throughput* of the system).

No matter what view, it is always possible to obtain correct values from computers/systems performance evaluation. But obtaining an absolute measure is another thing, since the final result always depends on the characteristics of the overall system. It is secure to say that no single metric can measure the performance of a computer on all applications, because system performance varies from one application domain to another.

From time to time some standard of measurement or evaluation - benchmarks - are brought out to the general public. These benchmarks are usually computer programs that load a well defined workload on a computer, and return some form of result – metric – that describes the performance of the tested computers.

Several benchmarks are currently available, and the purpose of this communication is to give an overview of the more popular, stressing some benchmark suites from SPEC.

## 2 Benchmarking Overview

While comparing different design alternatives, the most common thing to do is associate the performance of two different machines, it is usual to say that machine X is n times faster than machine Y, if the term faster means response time or execution time we can transform this statement in the following mathematical equation:

$$\frac{ExecutionTimeY}{ExecutionTimeX} = n$$

No matter if response time or throughput is taken into account, the key measurement is time. It is always the time that the system takes to complete one task (response time), or the time the system takes to complete many tasks (throughput) that really matters.

Obtaining the system performance by measuring the execution time is still a very difficult thing to do. If the execution time depended only in the processor speed, the task of evaluating different system surely will be a very easy and peaceful task, but unfortunately that is not the reality. The speed for executing one or more tasks depends also on disk and memory accesses, the amount and type of memory present, I/O activities, operating system, user applications, compilers, level of optimization, etc.

Benchmarking can be defined has a task of measuring the performance of a system/subsystem (or application) on a task or on a well defined set of tasks. The task/workload that is measured is the so called benchmark suite.

The user's own application and his/hers own workload would be the best comparison test. Unfortunately, it is very difficult to get a wide base of reliable, repeatable and comparable measurements of the many existing systems on the user's application and workload. The solution is to use standardized benchmarks as a reference point

Nevertheless it is important to understand the correlation between the user's computing needs and what the benchmark is really measuring – the workload put on the system by the benchmark and it's characteristics should be the more proximate as possible, in order to get more trustful results.

Nowadays manufacturers consider of extremely importance that the results of any "standard" benchmark should take in account some aspects like:

- The precise hardware configuration used – type of processor, clock speed, number of CPUs, memory size, cache size, video processor and memory, bus, disk speed, and so on. The full description of the computational system is of extreme importance (manufacturers may fall in temptation to say that a certain benchmark result is from one particular system, but in reality the system tested was a more powerful one).
- The operating system environment like OS version, file system, number of users, etc.
- The version of the benchmark used.
- The program language used, because the same program could have different execution times if implemented in different languages.
- The compiler used and the level of optimization when compiling the benchmarks.

In order to get plausible results, the system under test should be very well mentioned and all of the variables of the system should be very well controlled.

As seen above benchmarks are mere programs used to evaluate performance [2]:

- *Real programs*: the best benchmark is the user's own application program. While the common user may not know the exact amount of time spent in the execution of these programs, the benchmark developer knows that the specific program will be used to solve problems in real life (e.g. Spice, GNU and C compilers). These programs have I/O and several options that the user can select while running the application.
- *Kernels*: these benchmarks are based on a thoroughly analysis and in the knowledge that in many cases only 10 percent of the code uses about 90 percent of the CPU resources. So performance analysts extracted these code fragments and have used them as benchmarks (e.g. Livermore Loops and Linpack benchmarks).
- *Toy benchmarks*: these are very small programs that produce results already known to the user. These are useful due to their portability and because they are easy run.
- *Synthetic benchmarks*: are based on performance analyst's experience and knowledge of instruction executed, then synthetic code is created trying to match an average execution profile (e.g. Dhrystone and Whetstone benchmarks).

Nowadays there is an entire industry devoted to develop benchmarks; some of them are developed by research organizations, computer magazines, companies and even individual persons for benchmarking specific hardware and software systems. In addiction to these, there are benchmark suites sponsored by groups/consortia of different (and concurrent) computer companies. Next will be presented an overview of some specific benchmarks used for the task of performing evaluation.

### 2.1 Generic Benchmarks

MIPS (or Million Instructions per second) has been one alternative to metrics that use only time. For a certain program MIPS refers to:

$$MIPS = \frac{Instruction\ count}{Execution\ time\ x\ 10^6} = \frac{Clock\ rate}{CPI\ x\ 10^6} \qquad and \qquad Execution\ Time = \frac{Instruction\ count\ x\ CPI}{Clock\ rate}$$

Being MIPS a rate of operations per unit time, performance can be specified as the inverse of execution time – faster machines have bigger MIPS.

MIPS specify the instruction execution rate but it is dependent on the instruction set, making difficult to compare MIPS of different machines with different instruction sets, the definition does not consider the mix or the number of instructions [1].

The Whetstone and Dhrystone benchmarks are examples of synthetic benchmarks, and so these are not real problems and so may not reflect program behaviour for factors not measured.

Whetstone benchmark was the first intentionally written to measure computer performance and was designed to simulate floating point numerical applications.

Developed in 1984 by R.P. Wecker, Dhrystone is a benchmark program written in C, Pascal or Java that tests a system's integer performance. The program is CPU bound, performing no I/O functions or operating system calls. Dhrystones per second is the metric used to measure the number of times the program can run in a second.

The Linpack benchmark was derived from a real application which originated as a collection of linear algebra subroutines in Fortran. It tests floating point performance and results are presented in MFlops (millions of floating point instructions per second).

The Linpack Benchmark has evolved from a simple listing for one matrix problem to an expanded benchmark describing the performance at three levels of problem size on several hundred computers. The benchmark today is used by scientists worldwide to evaluate computer performance, particularly innovative advanced-architecture machines.

The Linpack Benchmark provides three separate benchmarks that can be used to evaluate computer performance on a dense system of linear equations: the first for a 100 by 100 matrix, the second for a 1000 by 1000 matrix. The third benchmark, HPL, is a software package that generates and solves a random dense linear system of equations on distributed-memory computers using 64-bit floating point arithmetic and portable routines for linear algebra operations and message passing. This is the benchmark used by TOP500 [8] which list the 500 fastest computers systems being used today.

### 2.2 TPC Benchmarks

The scientific community has evolved benchmarks that measure system performance on numeric computations. These scientific benchmarks do not give much guidance to someone evaluating a database system or a transaction processing system, because database system performance is dominated by the performance of software algorithms rather than by raw hardware speed.

TPC (Transaction Processing Performance Council) [3] is an internationally recognized consortium of vendors (e.g. AT&T, Fujitsu, HP, IBM, NEC Corp., Oracle, Siemens, SUN, to name only a few), that defines benchmarks for transaction processing and database domains.

Considering that it is difficult to quantify project risks, programming costs, and operations costs but computer performance *can* be more easily quantified and compared.

This quantitative comparison starts with the definition of a *benchmark* or *workload*. The benchmark is run on several different systems, and the performance and price of each system is measured and recorded. *Performance* is typically a throughput metric (work/second) and *price* is typically a five-year cost-of-ownership metric. Together, they give a price/performance ratio.

The TPC metrics capture peak performance and price/performance of transaction processing and database systems running simple update-intensive transactions. In addition, the TPC reporting procedures have a "full disclosure" mechanism that makes it difficult to stretch the truth too much. The benchmark must be done on standard hardware and released software. Any special tuning or parameter setting must be disclosed. In addition, the TPC highly recommends that an independent organization audit the benchmark tests.

TPC benchmarks define how the tests should run, how system price should be measured and how the results should be reported. The TPC began by formalizing the ad hoc DebitCredit and TP1 benchmarks. The resulting two benchmarks were called TPC BM™ A and TPC BM™ B [4].

Nowadays TPC-A and TPC-B were replaced by TPC-C: characterise the performance of on-line transaction processing (OLTP) systems. This benchmark aims to characterise adequately the key aspects of a system intended for commercial on-line processing work.

TPC-D is another database benchmark, whose intent is to simulate ad hoc queries that provide answers to real-world business questions, characterizing the performance of decision support systems. Performance in TPC-D [4] is especially influenced by the intelligence of the query optimizer, table partitioning schemes, SQL functionality and advanced indexing techniques. TPC-D requires a full-function DBMS, SQL-based and able to support concurrent query and update.

TPC-W is a transactional web benchmark [5]. The workload on the system is performed in a controlled internet commerce environment that simulates the activities of a business oriented transactional web server. The workload tests some characteristics from some systems components related to this environment: simultaneous execution of multiple transaction; multiple on-line browser sessions; dynamic page generation with database access and update; on-line transaction execution modes; consistent web objects; databases consisting of many tables with a wide variety of sizes, attributes, and relationships and transaction integrity (ACID properties).

The performance metric reported by TPC-W is the number of web interactions processed per second.

### 2.3 SPEC Benchmarks

Measurements such as MIPS are too simple, and they do not capture the software performance and they miss the non-CPU component of the hardware performance.

The scientific and industrial communities developed specific benchmarks to replace MIPS as a measure of processor's performance. They developed suites of programs typical of scientific computations. One of these groups founded SPEC (Standard Performance Evaluation Corporation) [6], a non-profit consortium [7] made up by hardware and software vendors, universities, customers and different consultants, whose mission is to develop

technically credible and objective system-level benchmarks. Although no set of particular tests can fully characterize overall system performance, SPEC believes that the user community will benefit from an objective series of tests which can serve as a common reference point.

SPECs benchmarks allow comparison across many different technologies, architectures, implementations, memory systems, I/O subsystems, operating systems, clock rates, bus protocols, compilers, libraries, and application software.

The raw hardware performance depends on many components: CPUs, floating-point units (FPUs), I/Os, graphics and network accelerators, peripherals, and memory systems. However, system performance as seen by the user depends on the efficiency of the operating system, compiler, libraries, algorithms, and application software.

With so many variables, SPECs major goal was that the same source code (machine independent) would run on all members' machines, which required that all benchmarks be ported to SPEC members' machines.

The produced benchmarks are derived from real programs, placing on the system under test real workloads allowing computer designers and also users to take decisions based on realistic results.

SPEC chose a simple measure, elapsed time, to run the benchmark. Simple speed metric and machine-independent codes were keys to providing a comprehensive and fair comparison between competing machines.

For each benchmark suite SPEC decide to use the geometric mean of all the individual results, by doing so the publication of results is made simpler, each benchmark has the same weight, provides a consistent and fair metric.

As seen before it is very difficult to achieve one particular set of benchmarks (benchmarks suites) that can test the entire system, so SPEC has been developing several benchmark suites for certain specific domains, following are presented some benchmarks suites developed by SPEC that are oriented for different purposes.

CPU2000 – is a benchmark that provides a comparative measure of compute intensive performance across a wide range of hardware. The applications on this benchmark suite are dependent on the processor, memory and compiler of the tested system (this benchmark will be detailed in the next section).

SPECmail2001 – is a mail server benchmark designed to measure the system's ability to act as a mail server providing services for email requests based on the Internet standard protocols SMTP and POP3. It also allows mail-server and computer systems vendors to test and fine-tune products under development.

Results from SPECmail2001 are based on a *messages-per-minute* rating that indicates the load the mail server can sustain with a reasonable quality of service.

SPECweb99 – is the benchmark for measuring performance of web servers. The SPECweb99 workload simulates the access to a web service provider, where the server supports several home pages for a given number of different organizations; it also simulates dynamic operations such as "rotating" advertisements on a web page, customized web page creation, and user registration.

The benchmark's metric is SPECweb99. It represents the number of simultaneous connections the web server can support using the predefined workload.

SPECjvm98 – measures performance of Java Virtual Machines. It is applicable to standalone and networked Java client computers. SPECjvm98 allows users to evaluate performance on the software side, measuring the efficiency of JVM, the just-in-time (JIT) compiler, and operating system implementations. Relating to hardware, it includes CPU (integer and floating-point), cache, memory, and other platform-specific performance.

# 3 SPEC CPU2000

As seen before this benchmark suite (a set of different benchmarks or applications) provide an accurate comparative measure of computers performance. CPU2000 benchmark replaces the "old" CPU95 in measuring the performance of the computer's processor (CPU), memory architecture and compilers on the tested system.

It's important to remember the contribution of the last two components, because performance is more than the single processor.

SPEC CPU2000 is made up of two different components that focus two different types of computer's operations:

- CINT2000 for measuring and comparing computers integer performance.
- CFP2000 for measuring and comparing computers floating point performance.

CPU2000 does not put stress on other computer's components like I/O, networking, operating system or graphics. It provides a comparative measure of integer and/or floating point compute intensive performance – it is a good reference point if it matches the workload the user is interested in. Besides this the benchmark applications that compose this suite are developed from end-user applications, there are multiple vendors that use this suite which is a good thing for comparing different machines.

These benchmarks are required to be run and reported according to a rigid set of rules to ensure comparability and repeatability.

SPEC selected a Sun Ultra10 workstation with a 300 MHz SPARC processor and 256 MB of memory as a reference machine. All benchmark results are computed as ratios against the reference machine, which has a SPECint2000 and SPECfp2000 score of 100. SPEC ran each benchmark on the Ultra5_10 to establish a reference time. The speed and throughput metrics for a system are calculated as the geometric mean of the ratio (time the reference machine takes to complete a program/time the system under test takes) for each of the tests in the benchmark suite.

**CINT2000**

CINT2000 contains eleven applications that are used as benchmarks:

| Benchmark | Language | Resident size (Mb) | Virtual size (Mb) | Description |
|---|---|---|---|---|
| 164.gzip | C | 181 | 200 | Compression |
| 175.vpr | C | 50 | 55.2 | FPGA circuit placement and routing |
| 176.gcc | C | 155 | 158 | C programming language compiler |
| 181.mcf | C | 190 | 192 | Combinatorial optimization |
| 186.crafty | C | 2.1 | 4.2 | Game playing: Chess |
| 197.parser | C | 37 | 62.5 | Word processing |
| 252.eon | C++ | 0.7 | 3.3 | Computer visualization |
| 253.perlbmk | C | 146 | 159 | Perl programming language |
| 254.gap | C | 193 | 196 | Group theory, interpreter |
| 255.vortex | C | 72 | 81 | Object-oriented database |
| 256.bzip2 | C | 185 | 200 | Compression |
| 300.twolf | C | 1.9 | 4.1 | Place and route simulator |

All of these applications are based on real programs including loss-free data compression (the tools *gzip* and *bzip2* ), design of integrated circuits (component placement and auto-routing), logistics, mathematics, ray tracing, natural language processing, a modified Perl interpreter and a chess program.

The CINT2000 can measure the following metrics:

- SPECint2000: geometric mean of the twelve normalized ratios (one for each integer benchmark) when compiled with aggressive optimization for each benchmark.
- SPECint_base2000: geometric mean of twelve normalized ratios when compiled with conservative optimization for each benchmark.
- SPECint_rate2000: geometric mean of twelve normalized throughput ratios when compiled with aggressive optimization for each benchmark.
- SPECint_rate_base2000: geometric mean of twelve normalized throughput ratios when compiled with conservative optimization for each benchmark.

**CFP2000**

CFP2000 contains 14 applications (6 Fortran-77, 4 Fortran-90 and 4 C) that are used as benchmarks for floating-point operations:

| Benchmark | Language | Resident size (Mb) | Virtual size (Mb) | Description |
|---|---|---|---|---|
| 168.wupwise | F77 | 176 | 177 | Physics: Quantum chromodynamics |
| 171.swim | F77 | 191 | 192 | Shallow water modelling |
| 172.mgrid | F77 | 56 | 56.7 | Multigrid solver: 3D potential field |
| 173.applu | F77 | 181 | 191 | Partial differential equations |
| 177.mesa | C | 9.5 | 24.7 | 3D graphics library |
| 178.galgel | F90 | 63 | 155 | Computational fluid dynamics |
| 179.art | C | 3.7 | 5.9 | Image recognition/neural networks |
| 183.equake | C | 49 | 51.1 | Seismic wave propagation simulation |
| 187.facerec | F90 | 16 | 18.5 | Image processing: Face recognition |
| 188.ammp | C | 26 | 30 | Computational chemistry |
| 189.lucas | F90 | 142 | 143 | Number theory/primarily testing |
| 191.fma3d | F90 | 103 | 105 | Finite-element crash simulation |
| 200.sixtrack | F77 | 26 | 59.8 | Nuclear physics accelerator design |
| 301.apsi | F77 | 191 | 192 | Meteorology: Pollutant distribution |

This suite of applications includes numerous analysis and simulation programs from the scientific field: physics, chemistry, geology, mathematics and computer science. It also includes a data visualization program implemented in OpenGL, measuring the calculating speed, without displaying the result on the monitor. With these applications at least three compilers are required C, C++ and one that can handle Fortran77 as Fortran90.

As for CINT2000, CFP2000 can have different metrics:

- SPECfp2000: geometric mean of fourteen normalized ratios (one for each floating point benchmark) when compiled with aggressive optimization for each benchmark.
- SPECfp_base2000: geometric mean of fourteen normalized ratios when compiled with conservative optimization for each benchmark.
- SPECfp_rate2000: geometric mean of fourteen normalized throughput ratios when compiled with aggressive optimization for each benchmark.
- SPECfp_rate_base2000: geometric mean of fourteen normalized throughput ratios when compiled with conservative optimization for each benchmark.

The ratio for each of the benchmarks is calculated using a SPEC determined reference time and the run time of the benchmark. A higher value means a better performance on the given workload.

**Difference between base and peak metric**

In order to provide comparisons across different computer hardware, SPEC provides the benchmarks as source code that must be compiled. Any user can set the level of optimization he thinks adequate, but SPEC imposes strict rules for levels of optimization: base metrics (e.g. SPECint_base2000), are required for all results and must follow rigid

and well established rules. Peak metrics (e.g. SPECfp2000) are optional and have less strict requirements, in this way the best possible performance can be more easily to get, even though all the options must be very well detailed in the final report.

**Difference between rate and speed metric**

There are several different ways to measure computer performance. One way is to measure how fast the computer completes a single task; this is a speed measure (e.g. SPECfp2000). Another way is to measure how many tasks a computer can accomplish in a certain amount of time; this is called a throughput, capacity or rate measure (e.g. SPECfp_rate2000).

## 4 Conclusions

The issue of performance evaluation has been placed since always. In the early days, metrics like MIPS, MFLOPS or the ones given by the Dhrystone and Whetstone were good enough, but with the development of computers it became clear that these metrics were inadequate. Then several organizations attempted to solve that old issue, and there was a boom of new benchmarks. So what is the best benchmark?

The ideal benchmark would be your own workload on your own application, but unfortunately this is not possible. It is important to note that no standardized benchmark can provide a perfect model of a particular system, so the user should use the benchmark that fits best its need: if he wants to measure transactions and database access then TPC benchmarks might be a good option, but if he is more interested in measure the processor performance, then CPU2000 from SPEC might be the choice. TPC and SPEC benchmarks may be good starting points since they (i) are selected by a consortium of different key role players that control and validate all the proposed results, and (ii) all the variables in the system are controlled as much as possible: full hardware characteristics must be given, and all levels of optimizations are rigorously controlled.

What is valid today will be obsolete with time and the standards organizations are continuously working to update their suites: SPEC will shortly release CPU2004, the successor of CPU2000.

## References


[1] Patterson, David, Hennessy, John, The role of performance, Computer Organization & Design: the hardware/software interface, Morgan Kaufman Publishers

[2] Patterson, David, Hennessy, John, Computer Architecture a Quantitative Approach, Morgan Kaufman Publishers

[3] Transaction Processing Performance Council. http://www.tpc.org

[4] Gray, Jim , The Benchmark Handbook, Morgan Kaufman Publishers

[5] Standard Performance Evaluation Corporation. http://www.spec.org

[6] Henning, John, SPEC CPU2000: Measuring CPU Performance in the New Millennium, IEEE Computer (July 2000)

[7] Bramer, Bryan. System Benchmarks, Lecture Notes, DeMontfort University, (2003) http://www.cse.dmu.ac.uk/~bb/Teaching/ComputerSystems/SystemBenchmarks/BenchMarks.html

[8] TOP 500 Supercomputer sites. www.top500.org