Benchmarking

Reinhold Weicker

Fujitsu Siemens Computers, 33094 Paderborn, Germany
reinhold.weicker@fujitsu-siemens.com

Abstract. After a definition (list of properties) of a benchmark, the major benchmarks currently used are classified according to several criteria: Ownership, definition of the benchmark, pricing, area covered by the benchmark. The SPEC Open Systems Group, TPC and SAP benchmarks are discussed in more detail. The use of benchmarks in academic research is discussed. Finally, some current issues in benchmarking are listed that users of benchmark results should be aware of.

1 Introduction: Use, Definition of Benchmarks

A panel discussion on the occasion of the ASPLOS-III symposium in 1989 had the title "Fair Benchmarking – An Oxymoron?" The event looked somewhat strange among all the other, technically oriented presentations at the conference. This anecdotal evidence indicates the unique status that benchmarking has in the computer area: Benchmarking is, on the one hand, a highly technical endeavor. But it also is, almost by definition, related to computer marketing.

We see that benchmarks are used in three areas:

1. Computer customers as well as the general public use benchmark results to compare different computer systems, be it a vague perception of performance or a specific purchasing decision. Consequently, marketing departments of hardware and software vendors drive to a large degree what happens in benchmarking. After all, the development of good benchmarks is neither easy nor cheap, and it is mostly the computer vendors' marketing departments that in the end, directly or indirectly, pay the bill.
2. Developers in hardware or software departments use benchmarks to optimize their products, to compare them with alternative or competitive designs. Quite often, design decisions are made on the basis of such comparisons.
3. Finally, benchmarks are heavily used in computer research papers because they are readily available, and they can be expected to be easily portable. Therefore, when quantitative claims are made in research papers, they are often based on benchmarks.

Although the first usage of benchmarks (comparison of existing computers) is for many the primary usage, this paper tries to touch all three aspects.

What makes a program a benchmark? We can say that a benchmark is a standardized program (or detailed specifications of a program) designed or selected to be run on

M.C. Calzarossa and S. Tucci (Eds.): Performance 2002, LNCS 2459, pp. 179–207, 2002.

different computer systems, with the goal of a fair comparison of the performance of those systems. As a consequence, benchmarks are expected to be

1. Portable: It must be possible to execute the program on different computer systems.
2. Fair: Even portable programs may have a certain built-in bias for a specific system, the system that they have been originally written for. Benchmarks are expected to minimize such a bias.
3. Relevant, which typically means "representative for a relevant application area": A performance comparison makes no sense if it is based on some exotic program that may be portable but has no relation to a relevant application area. The benchmark must perform a task that is in some identifiable way representative for a broader area.
4. Easy to measure: It has often been said that the best benchmark is the customer's application itself. However, it often would be prohibitively expensive to port this application to every system one is interested in. The next best thing is a standard benchmark accepted by the relevant players in the field. Both customers and developers appreciate it if benchmark results are available for a large number of systems.
5. Easy to explain: It helps the acceptance of a benchmark if readers of benchmark results have some feeling about the meaning of the result metrics. Similarly, a benchmark is often expected to deliver its result as one "single figure of merit". Experts can and should look at all details of a result but it is a fact of life that many readers only care about one number, a number that somehow characterizes the performance of a particular system.

In his "Benchmark Handbook" [3], Jim Gray lists another property: Scalability, i.e. the benchmark should be applicable to small and large computer systems. While this is a desirable property, some benchmarks (e.g., BAPCo) are limited to certain classes of computers, and still are useful benchmarks.

We will come back to these goals during the following presentations of some important benchmarks. Not surprisingly, some of the goals can be in conflict with each other.

2 Overview, Classification of Benchmarks

The benchmarks that have been or are widely used can be classified according to several criteria. Perhaps the best starting point is "Who owns / administers a benchmark?" It turns out that this also roughly corresponds to a chronological order in the history of benchmarking.

2.1 Classification by Benchmark Ownership, History of Benchmarks

In earlier years, benchmarks were administered by single authors, then industry associations took over. In the last years, benchmarks became popular that are administered by important software vendors.

2.1.1 Individual Authors, Complete Programs

The first benchmarks were published by individual authors and have, after initial publication, spread basically through "word of mouth". Often, their popularity was a surprise for the author. Among those benchmarks are

Table 1. Single-author complete benchmarks

Name and Author(s)	Year	Language	Code size, in byte	Characterization
Whetstone (Curnow/Wichman)	1976	ALGOL 60 / Fortran	2,120	Synthetic Numerical Code, FP-intensive
Linpack (J. Dongarra)	1976	Fortran	(Inner loop:) 230	Package Linear Algebra Numerical Code, FP-intensive
Dhrystone (R. Weicker)	1984	Ada / Pascal / C	1,040	Synthetic System-type code, integer only

A detailed overview of these three benchmarks, written at about the peak of their popularity, can be found in [11]. Among these three benchmarks, only Linpack has retained a certain importance, mainly through the popular "Top 500" list (www.top500.org). It must be stated, and the author, Jack Dongarra, has acknowledged, that it represents just "one application" (solution of a system of linear equations with a dense matrix), resulting in "one number". On the other hand, this weakness can turn into a strength: There is hardly any computer system in the world for which this benchmark has not been run; therefore many results are available. This has lead to the effect that systems are compared on this basis, which will never run, in real life, scientific-numeric code like Linpack.

2.1.2 Individual Authors, Microbenchmarks

The term "microbenchmark" is used for program pieces that intentionally have been constructed to test only one particular feature of the system under test; they do not claim to be representative for a whole application area. However, the feature that they test is important enough that such a specialized test is interesting. The best-known example, and an often-used one, is probably John McCalpin's "Stream" benchmark (www.cs.virginia.edu/stream). It measures "sustainable memory bandwidth and the corresponding computation rate for simple vector kernels" [7]. The once popular "lmbench" benchmark (www.bitmover.com/lmbench), consisting of various small programs executing individual Unix system calls, seems to be no longer actively pursued by its author.

2.1.3 Benchmarks Owned and Administered by Industry Associations

After the success of some small single-author benchmarks in the 1980's, it became evident that small single-author benchmarks like Dhrystone were insufficient to characterize the emerging larger and more sophisticated systems. For example, benchmarks with a small working set cannot adequately measure the effect of memory hierarchies (multi-level caches, main memory). Also, small benchmarks can easily be subject to targeted compiler optimizations. To satisfy the need for benchmarks that are larger and cover a broader area, technical representatives from various computer manufacturers founded industry associations that define benchmarks, set rules for measurements, and review and publish results.

Table 2. Benchmarks owned by industry associations

Benchmark group	Since	URL	Language of bench-marks	Application area	Systems typically tested
GPC / SPEC GPC	1986	www.spec.org/gpc	C	Graphics programs	Work-stations
Perfect / SPEC HPG	1987	www.spec.org/hpg	Fortran	Numerical programs	Supercom-puters
SPEC CPU	1988	www.spec.org/osg/cpu	C, C++, Fortran	Mixed programs	Workstation s, Servers
TPC	1988	www.tpc.org	Specifica-tion only	OLTP programs	Database Servers
BAPCo	1991	www.bapco.com	Object Code	PC Application s	PCs
SPEC System	1992	www.spec.org/osg	C (driver)	Selected system functions	Servers
EEMBC	1997	www.eembc.org	C	Mixed programs	Embedded processors

Among these benchmarking groups, SPEC seems to be some kind of a role model: Some later associations (BAPCo, EEMBC, Storage Performance Council) have followed, to a smaller or larger degree, SPEC's approach in the development and administration of benchmarks. Also, some older benchmarking groups like the "Perfect" or GPC groups decided to use SPEC's established infrastructure for result publication) and to continue their efforts as a subgroup of SPEC. The latest example is the ECPerf benchmarking group, which is currently continuing its effort for a "Java Application Server" benchmark within the SPEC Java subcommittee.

2.1.4 Benchmarks Owned and Administered by Major Software Vendors

During the last decade, benchmarks became popular that were developed by some major software vendors. Often, these vendors are asked about sizing decisions: How many users will be supported on a given system, running a specific application package? Therefore, the vendor typically combined one or more of his application packages with a fixed input and defined this as a benchmark. Later, the major system vendors who run the benchmark cooperate with the software vendor in the evolution of the benchmark, and there is usually some form of organized cooperation around a software vendor's benchmark. Still, the responsibility for result publication typically lies with the software vendor. The attractiveness of these benchmarks for computer customers lies in the fact that they immediately have a feeling for the programs that are executed during the benchmark measurement: Ideally, they are the same programs that customers run in their daily operations.

Table 3. Benchmarks administered by major software vendors

Software vendor	Since	URL	Benchmarks covered	Systems tested
SAP	1993	www.sap.com/ benchmark/	ERP software	Servers
Lotus	1996	www.notesbench. org	Domino and Lotus software, mainly mail	Servers
Oracle Applications	1999	www.oracle.com/ apps_benchmark/	ERP software	Servers

There are more software vendors that have created their own benchmarks, again often as a byproduct of sizing considerations, among them are Baan, Peoplesoft, Siebel, and others. Table 3 only lists those where the external use as a benchmark has become more important than just sizing.

Note that in this group, there is no column "Source Language": Although the application package typically has been developed in a high-level language, the benchmark code is the binary code generated by the software vendor and/or the system vendor for a particular platform.

2.1.5 Result Collections by Third Parties

For completeness, result collections should also be mentioned where a commercial organization does not develop a new benchmark but rather collects benchmark results measured elsewhere. The best-known example is IDEAS International (www.ideasinternational.com). Their web page on benchmarks displays the top results for the TPC, SPEC, Oracle Applications, Lotus, SAP, and BAPCo benchmarks. Such lists or collections of benchmark results from various sources may

serve a need of those that just want a short answer to the non-trivial question of performance ranking: “Give me a simple list ranking all systems according to their overall performance”, or “Just give me the top 10 systems, without all the details”. Often, the media, or high-level managers hard pressed on time, want such a ranking. However, the inevitable danger of such condensed result presentations is that important caveats, important details of a particular result get lost.

2.2 Other Classifications: Benchmark Definition, Pricing, Areas Covered

2.2.1 Benchmark Definition

There are basically three classes of benchmark definitions, with an important additional special case:

1. Benchmarks that are defined in source code form: This is the “classical” form of benchmarks. They require a compiler for the system under test but this is usually no problem. SPEC, EEMBC, and all older single-author benchmarks listed here belong to this group.
2. Benchmarks that are defined as binary codes: By definition, these benchmarks cover a limited range of systems only. However, the market of Intel/Windows compatible PCs is large enough that benchmarks covering only this area can be quite popular. The BAPCo benchmarks and other benchmarks often used by the popular PC press (not covered here) belong to this category.
3. Benchmarks that are defined as specifications only: The TPC benchmarks are defined by a specification document, TPC does not provide source code. Because of the need to provide a level playing field in the absence of source code, and to prevent loopholes, these specification documents are quite voluminous. For example, as of 2002, the current TPC-C definition contains 137 pages, the TPC-W definition even 199 pages.
4. Benchmarks administered by a software vendor are a somewhat special case: The code running on the system under test is machine code, but it usually is the code sold by the software vendor to his customers, and there is typically a version for every major instruction set architecture / operating system combination. The only problem can be that in the case of a small system vendor, where less systems are sold, the software vendor may not have tuned the code (compilation, use of specific OS features) as well as he does in the case of a big system vendors, where many copies of the software are sold. On the one hand, the software systems sold to customers will also have this property; so one can say that the situation represents real life. On the other hand, the feeling remains that this is somewhat unfair, penalizing smaller system vendors.

In all cases, even in the case where the code executed on the system is given in source or binary form, a benchmark is defined not only by the code that is executed but also by input data and by a document, typically called “Run and Reporting Rules”; it describes the rules and requirements for the measurement environment.

2.2.2 Price / Performance

Some benchmarks include "pricing" rules, i.e. result quotations must contain not only a performance metric but also a price/performance metric. Since its beginnings, TPC results have included such a metric, e.g. "price per tpm-C". Among the other benchmarks, only the Lotus benchmark and the SPEC GPC benchmarks have a price/performance metric.

The value of pricing in benchmarks is often subject to debate, in the benchmark organizations themselves and in the press. Arguments for pricing are:

- Customers naturally are interested in prices, and prices determined according to uniform pricing rules set by the benchmark organization have a chance to be more uniform than, say, prices published by a magazine.
- There is always a tendency among system vendors to aim for the top spot in the performance list. The requirement to provide price information can be a useful corrective against benchmark configurations that degrade into sheer battles of material: If a benchmark scales well, e.g. for clusters, then whoever can accumulate enough hardware in the benchmark lab, wins the competition for performance. The requirement to quote the price of the configuration may prevent such useless battles.

On the other hand, there are the arguments against pricing:

- In the case of systems that are larger than just a single workstation, prices are difficult to determine and have many components: Hardware, software, maintenance. It is hard to find uniform criteria for all components, in particular for maintenance; different companies may have different business models.
- In the computer business, prices get outdated very fast. It is tempting but misleading to compare a price published today with a price published a year ago.
- With the goal to be realistic, some pricing rules (e.g. TPC's rules) allow discounts, provided that they are generally available. On the other hand, this allows system vendors to grant such discounts just for configurations that have been selected with an eye on important benchmark results, making the price less realistic than it appears.
- Experience in benchmark organizations like TPC shows that a large percentage of result challenges have to do with pricing.. This distracts energy from the member organizations that could be better spent in the improvement of benchmarks.

Overall, the arguments against pricing appear to be more convincing. The traditional approach of the SPEC Open Systems Group (OSG) "Have the test sponsor publish, in detail, all components that were used, and encourage the interested reader to get a price quotation directly from vendors' sales offices" seems to work quite well.

2.2.3 Areas Covered by Benchmarks

Finally, an important classification of benchmarks is related to the area the benchmarks intend to cover. One such classification is shown in figure 1.

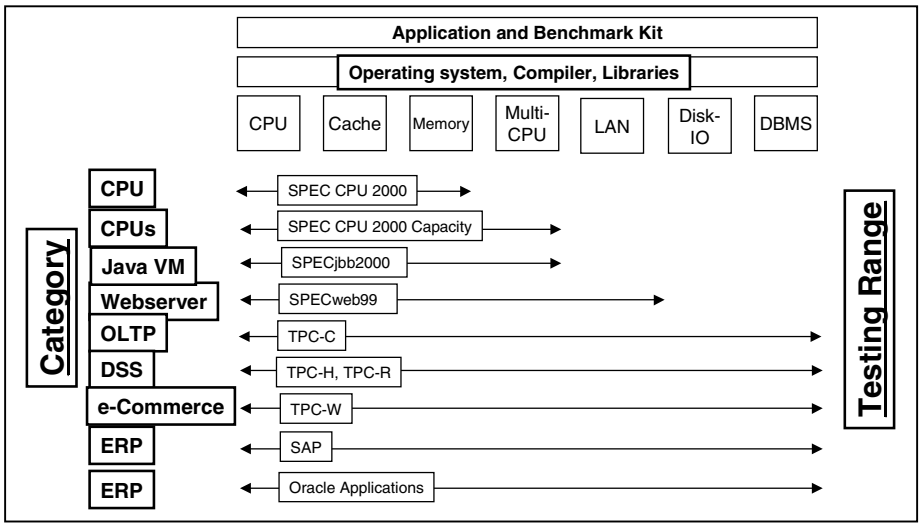

Fig. 1. Areas covered by some major benchmarks (from [10]).

Of course, the benchmark coverage also has some relation to the cost of benchmarking. For example, if networking is involved, the benchmark setup typically includes one or more servers and several (often many) clients. It is not surprising that the number of results for such benchmarks, where measurements take weeks or months, is much smaller than for those that involve only one system.

3 A Closer Look at the More Popular Benchmarks

In this section, the more important benchmarks from SPEC, TPC, and SAP are covered. They are, for large systems (servers), the most widely quoted benchmarks.

3.1 SPEC CPU Benchmarks, Measurement Methods

A CPU benchmark suite, now called CPU89, was SPEC's first benchmark product. When SPEC was founded in 1988, the explicit intention was to provide something better than the small single-author benchmarks that had been used before, and to provide standardized versions of larger programs (e.g. gcc, spice) that were already used by some RISC system vendors.

Since 1989, SPEC has replaced the CPU benchmarks three times, with CPU92, CPU95, and CPU2000. Currently, the SPEC CPU subcommittee is working on CPU2004, intended to replace the current suite CPU2000. On the other hand, the principle of SPEC CPU benchmarking has been remarkably consistent:

- A number of programs are contributed by the SPEC member companies, by the open source community, or by researchers in the academic community. For the CPU2000 suite, and again for CPU2004, SPEC has initiated an award program to encourage such contributions.
- The member companies that are active in the CPU subcommittee port the benchmarks to their various platforms; dependency on I/O or operating system activity is removed, if necessary. Care is taken that all changes are performance-neutral across platforms. If possible, the cooperation of the original program's author(s) is sought for all these activities.
- The benchmarks are tested in a tool harness provided by SPEC. Compilation and execution of the benchmarks is automated as much as possible. The tester supplies a "configuration file" with system-specific parameters (e.g. location of the C compiler, compilation flags, libraries, description of the system under test).

Table 4. SPEC's CPU benchmarks over the years

	CPU89	CPU92	CPU95	CPU2000
Integer programs	4	6	8	12
Floating-point programs	6	14	10	14
Total source lines, Integer	77,100	85,500	275,000	389,300
Total source lines, FP	24,200	44,000	20,600	158,300
Source languages	C, F77	C, F77	C, F77	C, C++, F77, F90
Number of results through Q1/2002	191	1292	1881	1043

SPEC's current CPU2000 suite is described in detail in [5]; this article also describes how SPEC selects its CPU benchmarks, and some problems that have to be solved in the definition of a new benchmark suite. Table 4 summarizes some key data on SPEC's CPU benchmarks (Note: Source line counts do not include comment lines).

The popular "speed" metric typically measures the execution time of each program in the suite when it is executed on one CPU (other CPUs are removed or deactivated), it replaces old "MIPS" metrics that had previously been used. The execution time of each program is set in relation to a "SPEC Reference Time" (execution time on a specific older popular system), and the geometric mean of all these performance ratios is computed as an overall figure of merit (SPECint, SPECfp).

Soon after the introduction of the CPU89 suite, manufacturers of multi-CPU systems wanted to show the performance of their systems with the same benchmarks. In response, SPEC developed rules for a "SPECrate" (throughput) computation: For each benchmark in the suite, several copies are executed simultaneously (typically, as many copies as there are CPUs in the system), and total throughput of this parallel

execution (jobs per time period) is computed on a per-benchmark basis. A geometric mean is computed, similar to the “speed” case.

In its official statements, SPEC emphasizes the advice “Look at all the numbers”. This means that a customer interested in a particular application area weights benchmarks from this area higher than the rest. Experts can draw even more interesting conclusions, correlating, for example, the working sets for particular benchmarks with the test system’s cache architecture and cache sizes [5]. However, such evaluations remained, to a large degree, an area of experts only; customers rarely look at more than the number achieved in the overall metric.

3.2 Evolution of the SPEC CPU Benchmarks, Issues

Despite the consistency of the measurement method, a number of new elements were brought into the suite over the years. Both the SPEC-provided tool harness and the Run Rules regulating conformant executions of the suite grew in complexity. The most important single change was the introduction of the “baseline” metric in 1994.

In the first publications of SPEC CPU benchmark results, the “Notes” section where all compilation parameters (in Unix terminology: “Flags”) must be listed, consisted of a few short lines, like

```
Optimization was set to -O3 for all benchmarks except fppp and
spice2g6, which used –O2
```

Over time, this changed drastically; today, one to three small-print lines per benchmark like

```
197.parser: -fast -xprefetch=no%auto -xcrossfile -xregs=syst
          -Wc,-Qgsched-trace_late=1,-Qgsched-T4 -xalias_level=strong
          -Wc,-Qipa:valueprediction –xprofile
```

listing the optimization flags used just for this single benchmark, are not uncommon. Compiler writers had always used internal variables controlling, for example, the tradeoff between the benefit of inlining (elimination of calling sequences etc.) and its negative effects (longer code, more instruction cache misses). Now, these variables were made accessible to the compiler user – an easy and straightforward change in the compiler driver. However, the user often does not have the knowledge to apply the additional command-line parameters properly, nor the time to optimize their use. In addition, sometimes “assertion” flags are used which help a particular benchmark but which would cause other programs to fail. This soon prompted the question whether such excessive optimizations are representative for real programming, or whether it would be better to “burn all flags" [13]. The issue was later discussed in academic publications [1,8] and in the trade press.

After a considerable debate, the final compromise for SPEC, established in January 1994, was that two metrics were established, “baseline” and “peak”: Every SPEC CPU benchmark measurement has to measure performance with a restricted “baseline” set of flags (metric name: SPECint_base2000 or similar), and optionally with a more extended “peak” set of flags. In its own publications, on the SPEC web

site, SPEC treats them equally. In marketing material, most companies emphasize the higher peak numbers; therefore even the existence of a baseline definition may be unknown to some users of SPEC results. The exact definition of "baseline" can be found in the suite's Run and Reporting Rules. For CPU2000, it can be summarized as follows:

- All benchmarks of a suite must be compiled with the same set of flags (except flags that may be necessary for portability).
- The number of optimization flags is restricted to 4.
- Assertion flags (flags that assert a certain property of the program, one that may not hold for other programs, and that typically allows more aggressive optimizations) are not allowed: This property may not hold for other programs, and one of the basic principles of baseline is that they are "safe", i.e. do not lead to erroneous behavior for any language-compliant program.

The idea is now generally accepted that it makes sense to have, in addition to the "everything goes (except source code changes)" of the "peak" results, a "baseline" result. However, the details are often controversial, they emerge as a compromise in SPEC's CPU subcommittee. The question "What is the philosophy behind baseline?" may generate different answers if different participants are asked. A possible answer could be:

Baseline rules serve to form a "baseline" of performance that takes into account

- reasonable ease of use for developers,
- correctness and safety of the generated code,
- recommendations of the compiler vendor for good performance,
- representativity of the compilation/linkage process for what happens in the production of important software packages.

Again, it cannot be disputed that in some cases, individual points may contradict each other: What if the vendor of a popular compiler sets, for performance reasons, the default behavior to a mode that does not implement all features required by the language definition? This usage mode may be very common – most users lack the expertise to recognize such cases anyway -, but correctness of the generated code, as defined by the language standard, is not guaranteed. Since SPEC releases the CPU benchmarks in source code form, correct implementation of the programming language as defined by the standard is a necessary requirement for a fair comparison.

The baseline topic is probably the most important single issue connected with the SPEC CPU benchmarks. But there are other important issues as well:

- How should a compute-intensive multi-CPU metric be defined? The current SPECrate method takes a simplistic approach: Execute, for example on an n-CPU system, n copies of a given benchmark in parallel, record their start and end times. This introduces some artificial properties into the workload. In a real-life compute-intensive environment, e.g. in a university computing center, batch jobs come and go at irregular intervals and not in lockstep. Typically, there is also an element of overloading: More jobs are in the run queue than CPUs in the system.
- SPEC needs to distribute its CPU benchmarks as source code. For the floating-point suite, it seems possible to get, with some effort, good, state-

of-the-art codes from researchers (engineering, natural sciences). It is more difficult to get similarly good, representative source codes in non-numeric programming. Such programs are typically very large, and the developers, the vendors of such software cannot give them away for free in source code form. Traditionally, SPEC has relied heavily on integer programs from the Open Source community (GNU C compiler, Perl, etc.), but these programs are only one part of the spectrum.

- It has been observed [2] that the SPEC CPU benchmarks, in particular the integer benchmarks, deviate in their cache usage from what is typically observed on "live" systems. The observations in [2] were for the CPU92 benchmarks but the tendency still can be observed today. To some degree, such differences are an unavoidable consequence of the measurement method: For the purpose of benchmarking, with a reasonably long measurement interval, a SPEC CPU benchmark runs for a longer time uninterrupted on one CPU than in real-life environments where jobs often consist of several cooperating processes. Due to the absence of context switches, SPEC CPU measurements show almost no process migration with subsequent instruction cache invalidations.

In particular the last point should be taken by SPEC as a reason to think about other alternatives to measure the raw computing power of a system: In a time when multiple CPUs are placed on a single die, does it make sense to artificially isolate the speed of one CPU for a traditional "speed" measurement?

3.3 SPEC System Benchmarks

SPEC OSG started with CPU benchmarks but very soon also developed benchmarks that measured system performance. The areas that SPEC chose to put efforts in were determined by a perception of the market demands as seen by the SPEC OSG member companies. For example, when the Internet and Java gained popularity, SPEC OSG soon developed its Web and JVM benchmarks. Currently, Java on servers and mail servers are seen as hot topics; therefore, Java server benchmarks and mail server benchmarks are areas where SPEC member companies invest considerable efforts on the development of new benchmarks. There are also areas where SPEC's efforts were unsuccessful: SPEC worked for some time on an I/O benchmark but finally could not find a practical way between raw device measurements and system-specific I/O library calls. (These efforts apparently are taken up now by a separate industry organization, the "Storage Performance Council", see www.storageperformance.org). The only area intentionally left out by SPEC is transaction processing, the traditional domain of the sister benchmarking organization TPC.

It is important to realize that SPEC's system benchmarks are both broader and narrower than the component (CPU) benchmarks:

- They are broader than the CPU benchmarks because they test more components of the system, typically including the OS, networking, and – for some benchmarks – the I/O subsystem.

- They are narrower than the CPU benchmarks because they test the system when it is executing specific, specialized tasks only, e.g. acting as a file server, a web server or a mail server.

This narrower scope of some system benchmarks is not unrelated to real-life practice: Many computers are exclusively used as file servers, web servers, database servers, mail servers, etc. Therefore it makes sense to test them in such a limited scenario only.

Most system benchmarks present results in the form of a table or a curve, giving, for example, the throughput correlated with the response time. This corresponds to SPEC's philosophy "Look at all the numbers", similarly as the CPU benchmark suites gives the results for each individual benchmark. However, SPEC is realistic enough to know that the market often demands a "single figure of merit", and has defined such a number for each benchmark (e.g. maximum throughput, throughput at or below a specific response time). Table 5 lists SPEC's current system benchmarks, with the number of publications over the years.

Table 5. SPEC OSG system benchmark results over the years

	SDM	SFS	SPEC web96	SPEC web99	SPEC jvm98	SPEC jbb2000	SPEC mail2001
1991	51						
1992	17						
1993	5	21					
1994	6	18					
1995	15	21					
1996		14	22				
1997		36	50				
1998		19	80		21		
1999		96	61	5	26		
2000		62	12	49	23	22	
2001		25+34		57	4	57	6
Q1/2002		21		13	3	21	3
Overall	94	215	225	124	77	100	9

Some general differences to the CPU benchmarks are:

- The workload is, almost inevitably, synthetic. Whereas SPEC avoids synthetic benchmarks for the CPU suites, workloads for file server or web server benchmarks cannot be derived from "real life" without extensive trace capturing / replay capabilities that make use in benchmarks impractical. On the other hand, properties that are important for system benchmarks, e.g. file sizes, can be more easily modeled in synthetic workloads.
- The results are more difficult to understand; therefore the benchmarks are possibly not as well known and not as popular as the CPU benchmarks.

- Finally, the measurement effort is much larger. Typically, these benchmarks need clients / workload generators that add considerably to the cost of benchmarking. Therefore, fewer results exist than for the CPU benchmarks.

3.3.1 SDM Benchmark

SPEC's first system benchmark suite, released in 1991, was SDM (System Development Multiuser, released 1991). It consists of two individual benchmarks that differ in some aspects (workload ingredients, think time) but have many properties in common: Both have a mix of general-purpose Unix commands (ls, cd, find, cc, ...) as their workload. Both use scripts or simulated concurrent users that put more and more stress on the system, until the system becomes overloaded and the addition of users results in a decrease in throughput.

After initial enthusiasm, interest in the benchmark died down. Today, SDM results are no longer requested by customers, and therefore not reported by SPEC member companies. This is an interesting phenomenon because we know that many SPEC member companies still use SDM heavily as an internal performance tracking tool: Whenever a new release of the OS is developed, its performance impact is measured with the SDM benchmarks. Why has it then been unsuccessful as a benchmark? It seems that there are several reasons:

- SDM measures typical Unix all-around systems; today, we have more client-server configurations.
- For marketing, the fact that there is not an overall, easy-to-explain single figure of merit (only individual results for sdet and kenbus; no requirement to report both) is probably a severe disadvantage.
- Most important, the ingredients are the system's Unix commands, and several of them (cc, nroff) take a large percentage of time in the benchmark. This opens up possibilities for SDM-specific versions of these commands, e.g. a compiler (cc command) that does little more than syntax checking: Good for great SDM numbers but useless otherwise.

SDM is a good example for the experience that a good performance measurement tool is not yet necessarily a good benchmark: An in-house tool needs no provisions against unintended optimizations ("cheating"); the user would only cheat himself or herself. A benchmark whose results are used in marketing must have additional qualities: It must be tamper-proof against substitution of fast, special-case, benchmark-specific components where normally other software components would be used.

3.3.2 SFS Benchmark

The SFS benchmark (System benchmark, File Server, first release 1993) is SPEC's first client-server benchmark and has established a method that was later used for other benchmarks also: The benchmark code runs solely on the clients, they generate NFS requests like lookup, getattr, read, write, etc. Unix is required for the clients, but the server, the system under test, can be any server capable of accepting NFS

requests. SFS tries to be as independent from the clients' performance as possible, concentrating solely on measuring the server's performance.

Despite the large investment necessary for the test sponsor (the setup for a large server may include as many as 480 disks for the server, and as many as 20 load-generating clients), there has been a steady flow of result publications, and SFS is the established benchmark in its area. In 1997, SFS 1.0 was replaced by SFS 2.0. The newer benchmark covers the NFS protocol version 3, and it has a newly designed mix of operations.

In summer 2001, prompted by observations during result reviews, SPEC discovered significant defects in its SFS97 benchmark suite: Certain properties built into the benchmark (periodic changes between high and low file system activities, distribution of files accesses, numeric accuracy of the random process selecting the files that are accessed) were no longer guaranteed with today's fast processors. As a consequence, SPEC has suspended sales of the SFS97 (2.0) benchmark and replaced it by SFS97 R1 V3.0. Result submissions had to start over, since results measured by the defective benchmark cannot be used without considerable care in interpretation.

3.3.3 SPECweb Benchmarks

The SPECweb benchmark (Web server benchmark, first release 1996) was derived from the SFS benchmark, and it has many properties in common with SFS:

- It measures the performance of the server and tries to do this, as much as possible, independently from the clients' performance.
- A synthetic load is generated on the clients, generating HTTP requests. In the case of SPECweb96, these were static GET requests only, the most common type of HTTP requests at that time. SPECweb99 added dynamic requests (dynamic GET, dynamic POST, simulation of an ad rotation scheme based on cookies).
- The file size distribution is based on logs from several large web sites; the file set size is required to scale with the performance

Different from SFS, the benchmark code can run on NT client systems as well as on Unix client systems. Similar to SFS, the server can be any system capable of serving HTTP requests. As apparent from the large number of result publications (see table 5), SPEC's web benchmarks have become very popular.

When it became evident that electronic commerce now constitutes a large percentage of WWW usage, and that this typically involves use of a "Secure Socket Layer" (SSL) mechanism, SPEC responded with a new Web benchmark, SPECweb99_SSL. In the interest of a fast release of the benchmark, SPEC did not change the workload but just added SSL usage to the existing SPECweb99 benchmark. A new SSL handshake is required whenever SPECweb99 terminated a "keepalive" connection (in the average, every tenth request). Of course, it makes no sense to compare SPECweb99_SSL results with (non-SSL) SPECweb99 results. An exception can be results that have been obtained for the same system; they can help to evaluate the performance impact of SSL encryption on the server.

3.3.4 SPEC Java Benchmarks

In the years 1997/1998, when it became clear that Java performance was a hot topic in industry, SPEC felt compelled to produce, during a relatively short time, a suite for Java benchmarking. Previous benchmark collection in this area (Coffeinemark etc.) had known weaknesses that made them vulnerable to specific optimizations. The first SPEC Java benchmark suite followed the pattern of the established CPU benchmarks: A collection of programs, taken from real applications where possible, individual benchmark performance ratios, and the geometric mean over all benchmarks as a "single figure of merit". In the design on the benchmark suite and the run rules for the suite, several new aspects had to be dealt with:

- Garbage collection performance, even though it may occur at unpredictable times, is important for Java performance.
- Just-In-Time (JIT) compilers are typically used with Java virtual machines, in particular for systems for which the manufacturer wants to show good performance.
- During the first years, Java was typically used on small client systems (e.g. within web browsers), and memory size is an important issue for such systems.
- Finally, SPEC had to decide whether benchmark execution needed to follow the strict rules of the official Java definition, or whether some sort of offline compilation should be allowed.

SPEC decided to start with a suite that requires strict compliance with the Java Virtual Machine model. A novel feature of SPECjvm98 results is their grouping into three categories according to the memory size: Under 48 MB, 48 – 256 MB, over 256 MB. The experience seems to show that the test sponsors (mostly hardware manufacturers) followed SPEC's suggestion and produced results not only for big machines but also for "thin" clients.

During the last years, Java became more popular as a programming language not only for small (client) systems, but also for servers. SPEC responded with the SPECjbb2000 Java Business Benchmark which has become quite popular. Manufacturers of server systems now use it to demonstrate the capabilities of their high-end servers. The Java code executed in SPECjbb2000 resembles TPC's TPC-C benchmark (warehouses, processing of orders; see section 3.3) but the realization is different: Instead of real database objects stored on disks, Java objects (in memory) are used to represent the benchmark's data; therefore Java memory administration and garbage collection play an important role for performance. Overall, the intention is to model the middle tier in three-tier software systems; such software is now often written in Java.

Currently, SPEC is working on another benchmark that relies more directly on Java application software. A "Java Application Server" benchmark SPECjAppServer will include the use of program parts that use popular Java application packages (Enterprise JavaBeans ™) and will also involve the use of database software. In this benchmark, client systems drive the server; therefore the cost of benchmarking will be much higher.

3.4 TPC Benchmarks

Like SPEC, the Transaction Processing Performance Council (TPC), founded in 1988, is a non-profit corporation with the mission to deliver objective performance evaluation standards to the industry. However, TPC focuses on transaction processing and data base benchmarks. Another important difference to the SPEC OSG benchmarks is the fact that TPC, from its beginning, included a price/performance metric and made price reporting mandatory (see section 2.2). The most widely used benchmark of this organization, and therefore *the* classic system benchmark, is the TPC-C published in 1992, an OLTP benchmark simulating an order-entry business scenario.

The TPC-C benchmark simulates an environment where virtual users are concurrently executing transactions against a single database. The environment models a wholesale supplier, i.e. a company running a certain number of warehouses, each with an inventory of items on stock. Each warehouse is serving ten sales districts, and each district has 3000 customers. Sales clerks for each district are the virtual users of the benchmark who execute transactions. There are five types of transactions of which the new-order type is the most important, representing the entry of an order on behalf of a customer. The number of order entries per minute the system can handle is the primary performance metric, it is called tpmC (transactions per minute, TPC-C).

The user simulation has to obey realistic assumptions on keying and think times. For the database response times, the benchmark defines permissible upper limits, e.g. five seconds for the new-order transaction type. Further constraints apply to the shares of the transaction types, e.g. only about 45% of all transactions are new order entries, the remaining share is distributed to payments, posting of deliveries, and status checks. To only report the rate of new order entries is motivated by the desire to express a business throughput instead of providing abstract technical quantities. As transactions cannot be submitted to the system at arbitrary speed due to keying and think times, a dependency is enforced between the throughput and the number of emulated users, or the number of configured warehouses and ultimately the size of the database. The warehouse with its ten users is the scaling unit of the benchmark, it comes with customer and stock data and an initial population of orders that occupy physical disk space.

Since TPC does not define its benchmarks via source code, the database vendors, often in cooperation with the major system vendors, typically produce a “benchmark kit” which implements the benchmark for a specific hardware/software configuration. It is then the task of the TPC-certified auditor to check that the kit implements the benchmark correctly, in addition to verifying the performance data (throughput, response times). An important requirement checked by the auditor, which is unique to TPC, are the so-called “ACID properties”

- Atomicity: The entire sequence of actions must be either completed or aborted.
- Consistency: Transactions take the resources from one consistent state to another consistent state.

- Isolation: A transaction's effect is not visible to other transactions until the transaction is committed.
- Durability: Changes made by committed transactions are permanent and must survive system failures.

The rationale for these "ACID properties" is the fundamental requirement that a benchmark must be representative for an application environment, and that therefore, the database used needs to be a "real database". In the case of transaction processing, faster execution could easily be achieved if the implementation would drop one or more of the "ACID property" requirements. But such an environment would not be one into which customers would have enough trust to store their business data in it, and any performance results for such an unstable environment would be meaningless.

Over the years, TPC-C underwent several revisions, the current major revision is 5.0. It differs from revision 3.5 in some aspects only (revised pricing rules); the algorithms remained unchanged. An earlier attempt ("revision 4.0") to make the benchmark easier to handle (e.g. fewer disks required), and at the same time more realistic (e.g. more complex transactions) failed and did not get the necessary votes within TPC. It can be assumed that the investment that companies had made into the existing results (which would then loose their value for comparisons) played a role in this decision.

Complementing TPC-C, TPC introduced, in 1995, its "decision support" benchmark TPC-D. While TPC-C is update-intensive and tries to represent on-line transactions as they occur in real life, TPC-D modeled transactions that change the data base content quite infrequently but perform compute-intensive operations on it, with the goal of supporting typical business decisions. This different task immediately had consequences for practical benchmarking:

- Decision support benchmarks are more CPU-intensive and less I/O intensive than transaction processing benchmarks.
- Decision support benchmarks scale better for cluster systems

In 1999, TPC faced a problem that can be, in a larger sense, a problem for all benchmarks where the algorithm and the input data are known in advance to the implementors: It is possible for the data base implementation to store data not only in the customary form as relations (n-tuples of values, in some physical representation) but to anticipate computations that are likely to be performed on the data later, and to store partial results of such computations. The database can already present "materialized views", as they are called, to the user. It is only a short step from this observation to the construction of materialized views that are designed with an eye towards the requirements of the TPC-D benchmark. Within a short time, such implementations brought a huge increase in reported performance, and TPC was forced to set new rules or to withdraw the benchmark. It decided to replace TPC-D with two successor benchmarks that are different with respect to optimizations based on advance knowledge of the queries: TPC-R allows them, TPC-H does not. It turned out that apparently, the users of TPC benchmark results considered a situation with materialized views as not representative for their environment. After two initial results in 1999/2000, no more TPC-R results were published, and TPC-R can be considered dead. In a broad sense, TPC-R and TPC-H, can be compared with SPEC's "peak" and "baseline" metrics: Similar computations, but in one case, more optimizations are allowed. It is interesting to note that the TPC customers apparently

were more interested in the "baseline" version of decision support benchmarks. Realizing that the split into two benchmarks was a temporary solution only, TPC currently works on a common successor benchmark for both TPC-H and TPC-R.

TPC's latest benchmark, TPC-W, covers an important new area; it has been designed to simulate the activities of a business oriented transactional Internet web server, as it might be used in electronic commerce. Correspondingly, the application portrayed by the benchmark is a retail store on the Internet with customer "browse and order" scenario. The figure of merit computed by the benchmark is "Web Interactions Per Second" (WIPS), for a given scale factor (overall item count). The initial problem of TPC-W seems to be its complexity since there are many components that can influence the result:

- Web server, application server, image server, database software, all of which can come from different sources
- SSL implementation
- TCP/IP realization
- Route balancing
- Caching

It could be due to this complexity that there are still relatively few TPC-W results (currently, as of June 2002, 13 results), much less than for TPC-C (79 active results for version 5). This may be an indication that in the necessary tradeoff between representativity (realistic scenarios) and easy of use, TPC might have been too ambitious and might have designed a benchmark that is too costly to measure. Also, the benchmark does not have an easy-to-understand, intuitive result metric. SPEC's "HTTP ops/sec" (in SPECweb96) may be unrealistic if one looks closer at the definition, but it at least *appears* more intuitive than TPC-W's "WIPS" (Web Interactions Per Second). In addition, when the first results were submitted, it became clear that more rules are necessary for the benchmark with respect to the role of the various software layers (web server, application server, database server).

3.5 SAP Benchmarks

Similarly to SPEC and TPC, SAP offers not only a single benchmark but a family of benchmarks; there are various benchmarks for various business scenarios. The SAP Standard Application Benchmarks have accompanied the SAP R/3 product releases since 1993. Since 1995, issues relating to benchmarking, and the public use of results are discussed by SAP and its partners in the SAP Benchmark Council. Benchmark definition and certification of results, however, is at the discretion of SAP. The number of benchmarks in the family is about a dozen, with the majority simulating users in online dialog with SAP R/3.

Two of the benchmarks, Sales and Distribution (SD) and Assemble-to-Order (ATO), cover nearly 90% of all published results. Historically, SD came first and gained its importance by being the tool to measure the SAPS throughput metric (SAP Application Benchmark Performance Standard) that is at the center of all SAP R/3 sizing:

100 SAPS are defined as 2000 fully business processed order line items per hour in the Standard SD benchmark. This is equivalent to 6000 dialog steps and 1600 postings per hour or 2400 SAP transactions per hour.

The SD and ATO business scenario is that of a supply chain: A customer order is placed, the delivery of goods is scheduled and initiated, an invoice in written. In SD, an order comprises five simple and independent items from a warehouse. In ATO, an individually configured and assembled PC is ordered, which explains the differences in complexity. The sequence of SAP transactions consists of a number of dialog steps or screen changes. By means of a benchmark driver the benchmarks simulate concurrent users passing through the respective sequence with 10 seconds think time after each dialog step. After all virtual users have logged into the SAP system and started working in a ramp-up phase, the users repeat the sequence as many times as is necessary to provide a steady state measurement window of at least 15 minutes. It is required that the average response time of the dialog steps is less than two seconds. In case of SD, users, response time, and the throughput expressed as SAPS are the main performance metrics. For ATO where the complete sequence of dialog steps is called *a fully business processed assembly order*, only the throughput in terms of assembly orders per hour is reported.

Looking at SAP benchmark results, it is important to distinguish "two-tier" and "three-tier" results, with the difference lying in the allocation of the different layers for presentation, application and database. The presentation layer is where users are running their front-end tools, typically PCs running a tool called *sapgui* (SAP Graphical User Interface*)*. In the benchmark, this layer is represented by a driver system where for each virtual user a special process is started that behaves like a "sapgui". This driver layer corresponds to the RTE layer in the TPC-C benchmark. It is the location of the SAP application layer software that determines the important distinction between "two-tier" and "three-tier" results:

- If it resides on the same system as the database layer, the result is called a "two-tier" or "central" result. In this case, about 90 % of the CPU cycles are spent in the SAP R/3 application and only 10 % in the data base code.
- If it resides on separate application servers (typically, there are several of them since the application part of the workload is easily distributable), the result is called a "three-tier" result.

Since in result quotations, the number of users is typically related to the database server, the 90-10 relation explains why the same hardware system can have a much higher "three-tier" result than "two-tier" result: In the "three-tier" case, all activities of the SAP application code are offloaded to the second tier. Typically, the benchmarker configures as many application systems as are necessary to saturate the database server. In a large three-tier benchmark installation in November 2000, Fujitsu Siemens used 160 R/3 application servers (4-CPU PRIMERGY N400, Pentium/Xeon-based) in connection with a large 64-CPU PRIMEPOWER2000 as the data base server. Figure 4 shows the impressive scenario of this large SAP three-tier SD measurement.

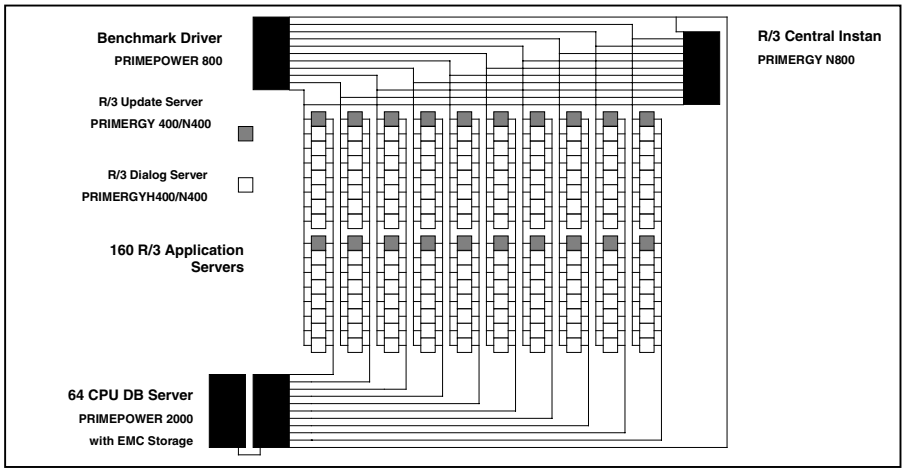

Fig. 2. Example of a large SAP three-tier installation (Fujitsu Siemens, Dec. 2000)

The sponsor of a three-tier SAP benchmark is free how to accommodate the processing needs of the application layer. For example, in another SAP measurement (IBM, with the p680 as database server), the application layer consisted of large p680 systems, too. The benchmarker is free to choose between configurations, based for example on ease of handling, or on the desire to implement a typical implementation for real-life client/server application environments.

It may be of interest to provide some insight into the workload on the database server as the most important factor in the three-tier case. There is a similarity to the TPC-C because both benchmarks deal with database server performance. But while in the TPC-C disk I/O is at the center of interest, it is network I/O in the SAP benchmark. The two workloads are complementing each other in this regard in a very nice way. In the benchmark setup summarized in Figure 4 there were 65000 network roundtrips or 130000 network I/Os per second. Five out of 64 CPUs of the data base server were assigned to handle the interrupt processing for this traffic between application layer and database. The network stack was standard TCP/IP, a mature industry standard.

4 Benchmarking and Computer Research

It is well known that several benchmarks, in particular the SPEC CPU benchmarks, are used extensively in the manufacturers' development labs; we shall discuss this aspect in section 5. A critical look at recent computer science conferences or journals, especially in the area of computer architecture and compiler optimization, shows that this phenomenon is not restricted to manufacturers, it appears in academic research also. Table 6 shows a snapshot for several major conferences in 2000/2001, listing how often benchmarks were used in conference papers (Note that some papers use more than one benchmark program collection).

Table 6. Use of benchmark collections in conference papers

	ASPLOS Nov. 2000	SIGPLAN June 2001	SIGARCH June 2001
Overall number of papers	24	30	24
SPEC CPU (92, 95, 2000)	8	4	17
SPEC JVM98	1	4	1
SPLASH benchmarks (Parallel Systems)	2	-	-
Olden benchmarks (Pointer, Memory)	-	-	3
OLTP / TPC	1	-	1
Various other program collections	11	13	8
No benchmark used	6	9	3

Looking at table 6, we can say:

- Benchmarks that are composed of several individual components (SPEC CPU, SPEC JVM98, Olden, SPLASH) are particularly popular. These are also the benchmarks that are relatively easy to run, compared with others.
- For specific research topics, some benchmark collections that emphasize a particular aspect are popular: SPECJVM98 for Java, the parallelizable SPLASH codes for research on parallel processing, the Olden benchmark collection of pointer- and memory-intensive programs for research on pointers and memory hierarchy.
- The SPEC CPU benchmarks are the most popular benchmark collection overall.
- Very few papers base quantitative data on OLTP workloads.

Given the importance of OLTP, TPC- and SAP-type workloads in commercial computing, one can ask whether academic research neglects to give guidance for computer developers as far as these environments are concerned. Fortunately, specialized workshops like the "Workshop on Computer Architecture Evaluation using Commercial Workloads" held in conjunction with the "Symposium on High Performance Computer Architecture" (www.hpcaconf.org) fill this gap.

"A Quantitative Approach" is the subtitle of one of the most popular and influential books on computer architecture [4]. The underlying thesis is that computer architecture should be driven by ideas whose value can be judged not only by intuition but also by quantitative considerations. A new design principle has to show a quantitative advantage over other designs. Following this approach, there is an apparent trend in Ph.D. dissertations and other research papers:

- A new idea is presented.
- A possible implementation (hardware, software, or a combination of both) is discussed, with operation times for individual operations.
- Simulation results are presented, on the basis of some popular benchmarks, very often some or all of the SPEC CPU benchmarks.
- The conclusion is "We found an improvement of xx to yy percent".

Such a statement is considered proof that the idea has value to it, and is relevant for successor projects, for research grants, and academic promotion.

This tendency, both in manufacturers' development labs and in academic research, places a responsibility on benchmark development groups that can be frightening: Sometimes, aspects of benchmarks become important for such design optimizations that were not yet thought of, or never discussed, when the benchmarks were selected. Suddenly, they develop an influence not only on the comparison of today's computers but also on the design of tomorrow's computers. For example, when the earlier CPU benchmark suites were put together, SPEC looked mainly at the source codes. Now, with techniques like feedback optimization and value prediction becoming more popular in research and possibly also in state-of-the-art compilers, one also has to look much more closely at the input data that are used with the benchmarks: Are they representative for typical problems in the area covered by the benchmarks? Do they encourage optimization techniques that have an over-proportional effect on benchmarks, as opposed to normal programs?

This author once attended an ASPLOS (Architectural Support for Programming Languages and Operating Systems) conference and made a critical remark about some papers that relied, in his opinion, too much on the SPEC CPU benchmarks. He was asked: "Do you mean that we should not use the SPEC benchmarks?" This, of course, would mean to "throw out the child with the bathwater". The result was a short contribution in a newsletter widely read by academic computer architects [12], asking to continue using the SPEC CPU benchmarks, but to use – if possible – all of them, and to report all relevant conditions (e.g. compiler flags). The main request was not to take benchmarks blindly as given, but to include a critical discussion of those properties of the benchmarks that may be relevant for the architectural feature that is studied. For example, in [14] it is shown that the program "HEALTH", one of the often-used "Olden Benchmarks" supposedly representative for linked-list data structures, is algorithmically so inefficient that any performance evaluations based on it are highly questionable. Surprisingly few research papers discuss the value of SPEC's benchmarks from an independent perspective, [2] is one of them (however, discussing the old CPU92 benchmarks). Occasionally, when surprising new results come up, online discussion group are full of statements "Benchmarks, and in particular the SPEC CPU benchmarks, are bogus anyway". But it is hard to find good, constructive criticisms of benchmark programs. More papers that compare benchmarks with other programs that are heavily used would be particularly useful. In one such paper [6], the authors compare the execution characteristics of popular Windows-based desktop applications with that of selected SPEC CINT95 benchmarks and find similarities (e.g., data cache behavior) as well as differences (e.g., indirect branches). Given the large influence – direct or indirect, even in academic research – that benchmarks can have, it would be beneficial for both computer development and computer science research if the benchmarks get the attention and critical scrutiny they deserve.

One has to acknowledge that the three usage areas for benchmarks mentioned in the introduction

1. Customer information and marketing, goal: Compare today's computers on a fair basis;

2. Design in manufacturers' labs, goal: Build better computers;
3. Computer science research, goal: Develop design ideas for the long-term future;

can call for different selection criteria: For goal 1, it makes sense to have representatives of today's programs in a benchmark suite, including instances of "dusty deck" code. For goals 2 and 3, it would make much more sense to only have programs of tomorrow, programs with good coding style, possibly programs of a type that rarely exists today. For example, it has been observed [6] that object-oriented programs and programs that make frequent use of dynamically linked libraries (DLL's) have different execution characteristics than the classical C and Fortran program in today's SPEC CPU suites.

In addition to its under-representation in academic research, a critical discussion of benchmarks is also often missing from computer science curricula. In the area of computer architecture, universities tend to educate future computer designers, not so much future computer users. On the other hand, it is obvious that many more graduates will eventually end up making purchase decisions than making design decisions for computers. They would benefit from a deeper knowledge about the values and the pitfalls of benchmarks.

5 Use of Benchmarks, Issues, and Opportunities

The presentation of various important benchmarks in section 3 has already shown some aspects that seem to appear across several benchmarks. Benchmarks as drivers of optimizations, conflicting goals for benchmarks, and a possible broader use of benchmarks, for more than just the generation of magical numbers, are some of these aspects.

5.1 Benchmarks as Drivers of Optimization

It is well known, and intended and encouraged by benchmark organizations, that good benchmarks drive industry and technology forward. Examples are:

- Advances in compiler optimization encouraged by the SPEC CPU benchmarks.
- Advances in database software encouraged by the TPC benchmarks.
- Optimizations in Web servers like caching for static requests, encouraged by the SPECweb benchmarks.

It must be emphasized that such advances in performance are a welcome and intended effect of benchmarks, and that the benchmarking organizations can take pride in these developments. However, some developments can also be counterproductive:

- Compiler writers may concentrate on optimizations that are allowed for the SPEC CPU benchmarks but that are not used by 80 or 90 % of the software developers: Then, the benchmarks lead to a one-sided and problematic allocation of resources. SPEC has tried to counter this with the requirement to publish "baseline" results. However, this can only be

successful as long as "baseline" optimizations allowed by SPEC are really relevant in the compilation/linkage process of real-life programs.

- Hardware designers may optimize the size of their caches to what the current benchmarks require, irrespective of longer-term needs.
- Database designers may concentrate too much on query situations that occur in benchmarks.

Looking at such issues from a more general point of view, the overall issue is that of the representativity of the benchmark; it tends to appear in several contexts:

- TPC had its problems with materialized views;
- SPEC CPU has its ongoing debates on peak vs. baseline;
- SPEC Web had debates about web server software that was, in one way or another, integrated with the operating system, leading to a sudden increase in performance.

Let us look at the example of the SPEC CPU benchmarks where the component benchmarks are given in source code form. If some aspect of the benchmark turns out to be particularly relevant for the measured performance, but if this property is not shared by important state-of-the-art programs, a particular optimization can suggest to the naive reader a performance improvement that is unreal, i.e. that is based on the specific properties of a particular benchmark only. An issue that seems to come up repeatedly with the SPEC CPU benchmarks is a "staircase" effect of certain single benchmarks with respect to caches. Some programs, together with their input data sets, have a critical working set size: If the working set fits into a cache, performance is a magnitude better than in the case that the working set size makes many memory accesses necessary, possibly connected with "cache thrashing" effects. Several SPEC CPU benchmarks (030.matrix300 in CPU89, 023.eqntott in CPU92, 173.art in CPU2000) apparently had such "magical" working size parameters and, consequently, showed sudden increases in their performance through compiler optimizations that managed to push the working set size below a critical, cache-related boundary. Such optimizations typically generated heated discussions inside and outside of SPEC: "Can such an increase in performance be real?" The optimizations themselves can be "legal", i.e. general enough that they are applicable to other programs also. But the specific performance gain may not be representative; it may come from a particular programming style that SPEC overlooked in its benchmark selection process.

It is not only the compiler and the CPU benchmarks where the question of representativity is relevant. System benchmarks often test the performance of a layered software system. The seven-layer ISO reference model for network architectures is a good example: In the interest of good software engineering practices (modularity, encapsulation, maintainability), software is often organized in layers, each performing a specific task. On the other hand, it is well known that shortcuts through these layers usually result in better performance; the various Web server caches now used in most SPECweb results are a good example. Where should benchmark specifications draw the line when such layer-transgressing optimizations occur for the first time at the occasion of a new benchmark result? What will be seen as an advance in technology, and what will be seen as a "benchmark special", a construct that is outlawed in most benchmarks' rules? Everyone will agree on extreme cases:

- Results where the traditional barrier between system mode and user mode is abandoned in favor of performance do not reflect typical usage and are rightly forbidden.
- On the other hand, the fact that databases running on top of Unix typically omit the Unix file system layer and operate directly on raw devices will be considered standard practice.

It is the cases in between such extremes that often cause lengthy debates in benchmark organizations. Because such optimizations are rarely foreseen at the time when the benchmark, together with its rules, is adopted, these cases typically arise during result reviews, where immediate vendor interests are also involved. TPC has a separate body, the "Technical Advisory Board" (TAB) that makes suggestions in cases of result compliance complaints or when TPC auditors ask for a decision. SPEC leaves these decisions to the appropriate subcommittee, where such discussions often take considerable time away from the committee's technical work, e.g. new benchmark development. But someone has to decide; the alternative that a test sponsor can do anything and remain unchallenged would be even worse.

With its ACID tests, TPC has found an interesting way to enforce the requirement that the software used during the benchmark measurement is "real database software". The ACID tests are a required part of the benchmark; if the system fails them, no result can be submitted. But they are executed separately from the performance measurement, outside the measurement interval. So far, SPEC has not formulated similar tests for its benchmark although one could imagine such tests:

- For the CPU benchmarks, SPEC requires that "hardware and software used to run the CINT2000/CFP2000 programs must provide a suitable environment for running typical C, C++, or Fortran programs" (Run Rules, section 1.2). For baseline, there is the additional requirement that "the optimizations used are expected to be safe" (2.2.1), which is normally interpreted by the subcommittee as a requirement that "the system, as used in baseline, implements the language correctly" (2.2.7). The idea of a "Mhillstone" program that would, outside the measurement interval and with more specialized programs, test for correct implementation, and that would flag illegal over-optimizations, has been debated a few times. However, with its limited resources, SPEC so far has not yet implemented such a test.
- The SPECweb benchmarks are heavily dominated by TCP/IP activity. SPEC requires in the Run Rules that the relevant protocols are followed; but there is no check included in the benchmark. Since the benchmark typically is executed in an isolated local network, an implementation could, theoretically, take a shortcut around requirements like the ability to repeat transmissions that have failed.
- The SPEC SFS benchmark requires stable storage for the file system used in the benchmark measurement. However, the benchmark does not contain code that would test this.

Other aspects of representativity so far have barely been touched by benchmark designers: In [9], Shubhendu Mukherjee points out that future processors might have several modes of operation in relation to cosmic rays: A "fault tolerant" mode where

intermittent errors caused by such cosmic rays are compensated by additional circuits (which make the CPU slower), and a "fast execution" mode where such circuitry is not present or not activated. He asks what would be the relevance of benchmark results have that can only be reproduced in the basement of the manufacturer's testing lab where the computer is better shielded from cosmic rays. In the software area, future Shared Memory Processor (SMP) systems may have cache coherency protocols that can switch from "snoopy" to "directory based" and vice versa, with possible performance implications for large vs. small systems. Benchmark results that have been obtained in one environment may not be representative for the other environment.

5.2 Conflicting Goals for Benchmarks

The discussions in the previous sections can be subsumed under the title "Representativity". An obvious way towards achieving this goal is the introduction of new benchmarks (like SPEC's sequence of CPU benchmarks, from CPU89 to CPU2000), or at least the introduction of new rules for existing benchmarks (like TPC's sequence of "major revisions" for its TPC-C benchmark). New benchmarks can make over-aggressive optimizations irrelevant, and the expectation that benchmarks will be retired after a few years can discourage special-case optimizations in the first place. On the other hand, marketing departments and end users want benchmarks to be "stable", to be valid over many years. This leads, for example, to one of the most frequently asked questions to SPEC: "Can I convert SPECint95 ratings to SPECint2000 ratings?" SPEC's official answer "You cannot – the programs are different" is not satisfying for marketing but necessary from a technical point of view.

5.3 More Than Just Generators of Single Numbers: Benchmarks as Useful Tools for the Interested User

Looking at official benchmark result publications, readers will notice that almost all results have been measured and submitted by hardware or software vendors. Given the large costs that are often involved, this is understandable. From a fairness point of view, this can also make sense: Every benchmarker tries to achieve the best result possible; therefore, if all measurements are governed by the same rules, the results should be comparable.

On the other hand, some important questions remain unanswered with this practice: What would be the result if the measurements were performed under conditions that are not optimal but reflect typical system usage better? Examples are:

- What CPU benchmark results would be achieved if the compilation uses a basic "-O" optimization only?
- What would be the result of a Web benchmark that intentionally does not use one of the popular Web cache accelerators?

- What would be the result of a TPC or SAP benchmark measurement if the CPU utilization of the server is at a level recommended for everyday use, and not as close to 100 % as possible?

Manufacturers typically do not publish such measurements because they are afraid of unfair comparisons: Everyone not using a performance-relevant optimization that is legal according to the benchmark's rules would run the risk that the result would let his system appear inferior to competitive systems for which only highly optimized results are published

As a consequence, such measurements under sub-optimal conditions are often performed in the development labs, but the results are never published. The only means to change this would be results published by independent organizations such as technical magazines. For its CPU benchmarks, SPEC has rules stating that if results measured by third parties are submitted to SPEC, a vendor whose products are affected can protest against such a result and hold up result publication for a few weeks but cannot prevent an eventual publication. The idea is that such a publication could be based on really silly conditions ("Let's make this manufacturer's system look bad"), and that the manufacturer should have a fair chance to counter with his own measurement. But in principle, measurements and result submissions by non-vendors are allowed; they just happen very rarely. On the other hand, such measurements, with appropriate accompanying documentation, could have considerable value for an informed and knowledgeable reader.

At least for easy-to-use benchmarks like SPEC's CPU and SPECjbb2000 benchmarks, valuable insights could be gained by the publications of more results that those promoted for marketing purposes. With appropriate documentation, readers could ask for answers to questions like

- What is the performance gain for a new processor architecture if programs are not recompiled? This often happens with important ISV codes.
- What is the performance loss if a PC uses RAM components that are slower but cheaper?
- For different CPU architectures, how much do they depend on sophisticated compilation techniques, how well can they deliver acceptable performance in environments that do not include such techniques?

It would be unrealistic to expect answers to such questions from vendor-published benchmark results; vendors cannot afford to publish anything but the best results. But if benchmarks are well-designed, representative, and portable – as they should be -, then, with some efforts from informed user organizations, researchers, or magazines, they could be used for much more than what is visible today.

Acknowledgments. I want to thank my colleagues in the Fujitsu Siemens Benchmark Center in Paderborn for valuable suggestions, in particular Walter Nitsche, Ludger Meyer, and Stefan Gradek, whose "Performance Brief PRIMEPOWER" [10] provided valuable information on several benchmarks that I have not been pursuing

actively myself. In addition, I want to thank my colleagues in the SPEC Open Systems Group; a large part of this paper is based on many years of experience in this group. However, it is evident that it contains a number of statements that reflect the author's personal opinion on certain topics, which may or may not be shared by other SPEC OSG representatives. Therefore, it must be stated that this is a personal account rather than a policy statement of either SPEC or Fujitsu Siemens Computers

References

1. Yin Chan, Ashok Sundarsanam, and Andrew Wolfe: The Effect of Compiler-Flag Tuning in SPEC Benchmark Performance. ACM Computer Architecture News 22,4 (Sept. 1994), 60–70
2. Jeffrey D. Gee, Mark D. Hill, Dionisios N. Pnevmatikos, and Alan Jay Smith: Cache Performance of the SPEC92 Benchmark Suite. IEEE Micro, August 1993, 17–27
3. Jim Gray: The Benchmark Handbook for Database and Transaction Processing Systems. Morgan Kaufmann, San Mateo, 2nd Edition, 1993, 592 pages
4. John Hennessy and David Patterson: Computer Architecture. A Quantitative Approach. 2nd Edition, Morgan Kaufmann, San Francisco 1996, 760 pages plus appendixes
5. John L. Henning: SPEC CPU2000: Measuring CPU Performance in the New Millenium. Computer (IEEE) 33,7 (July 2000), 28–35
6. Dennis C. Lee, et al: Execution Characteristics of Desktop Applications on Windows NT. 25th Annual Symposium on Computer Architecture, = ACM Computer Architecture News 26,3 (June 1998), 27–38
7. John McCalpin: http://www.cs.virginia.edu/stream/ref.html: FAQ list (Frequently Asked Questions) about Stream
8. Nikki Mirghafori, Margret Jacoby, and David Patterson: Truth in SPEC Benchmarks. ACM Computer Architecture News 23,5 (Dec. 1995), 34–42
9. Shubhendu S. Mukherjee: New Challenges in Benchmarking Future Processors. Fifth Workshop on Computer Architecture Evaluation using Commercial Workloads, 2002
10. Performance Brief PRIMEPOWER (PRIMEPOWER Performance Paper). Fujitsu Siemens, LoB Unix, Benchmark Center Paderborn, March 2002. Available, as of June 2002, under http://www.fujitsu-siemens.com/bcc/performance.html
11. Reinhold P. Weicker: An Overview of Common Benchmarks. Computer (IEEE) 23, 12 (Dec. 1990), 65–75
12. Reinhold Weicker: On the Use of SPEC Benchmarks in Computer Architecture Research. ACM Computer Architecture News 25,1 (March 1997), 19–22
13. Reinhold Weicker: Point: Why SPEC Should Burn (Almost) All Flags, and Walter Bays. Counterpoint: Defending the Flag. SPEC Newsletter 7,4, Dec. 1995, 5-6
14. Craig B. Zilles: Benchmark HEALTH Considered Harmful. ACM Computer Architecture News 29,3 (June 2001), 4–5