Notes on Calculating Computer Performance

Bruce Jacob and Trevor Mudge
Advanced Computer Architecture Lab
EECS Department, University of Michigan
{blj,tnm}@umich.edu

Abstract

This report explains what it means to characterize the performance of a computer, and which methods are appropriate and inappropriate for the task. The most widely used metric is the performance on the SPEC benchmark suite of programs; currently, the results of running the SPEC benchmark suite are compiled into a single number using the geometric mean. The primary reason for using the geometric mean is that it preserves values across normalization, but unfortunately, it does not preserve total run time, which is probably the figure of greatest interest when performances are being compared.

Cycles per Instruction (CPI) is another widely used metric, but this method is invalid, even if comparing machines with identical clock speeds. Comparing CPI values to judge performance falls prey to the same problems as averaging normalized values.

In general, normalized values must not be averaged and instead of the geometric mean, either the **harmonic** or the **arithmetic** mean is the appropriate method for averaging a set running times. The arithmetic mean should be used to average times, and the harmonic mean should be used to average rates (1/time). A number of published SPECmarks are recomputed using these means to demonstrate the effect of choosing a favorable algorithm.

1.0 Performance and the Use of Means

We want to summarize the performance of a computer; the easiest way uses a single number that can be compared against the numbers of other machines. This typically involves running tests on the machine and taking some sort of mean; the mean of a set of numbers is the central value when the set represents fluctuations about that value. There are a number of different ways to define a mean value; among them the arithmetic mean, the geometric mean, and the harmonic mean.

The **arithmetic mean** is defined as follows:

$$ArithmeticMean(a_1, a_2, a_3, \ldots, a_N) = \frac{\sum_i^N a_i}{N}$$

The **geometric mean** is defined as follows:

$$GeometricMean(a_1, a_2, a_3, \ldots, a_N) = \sqrt[N]{\prod_i^N a_i}$$

The **harmonic mean** is defined as follows

$$HarmonicMean(a_1, a_2, a_3, \ldots, a_N) = \frac{N}{\sum_i^N \frac{1}{a_i}}$$

In the mathematical sense, the geometric mean of a set of *n* values is the length of one side of an *n*-dimensional cube having the same volume as an *n*-dimensional rectangle whose sides are given by the *n* values. As this is neither intuitive nor informative, the wisdom of using the geometric mean for anything is questionable[1]. Its only apparent advantage is that it is unaffected by normalization: whether you normalize by a set of weights first or by the geometric mean of the weights afterward, the result is the same.

This property has been used to suggest that the geometric mean is superior, since it produces the same results when comparing several computers irrespective of which computer's times are used as the normalization factor [Fleming86]. However, the argument was rebutted in [Smith88], where the meaninglessness of the geometric mean was first illustrated.

In this report, we will consider only the arithmetic and harmonic means. Since the two are inverses of each other,

$$ArithmeticMean\,(a_1, a_2, a_3, \ldots) \;=\; \frac{1}{HarmonicMean\,(\frac{1}{a_1}, \frac{1}{a_2}, \frac{1}{a_3}, \ldots)}$$

and since the arithmetic mean—the "average"—is more easily visualized than the harmonic mean, we will stick to the average from now on, relating it back to the harmonic mean when appropriate.

UNITS AND MEANS: ***An Example***

We begin with a simple illustrative example of what can go wrong when we try to summarize performance. Rather than demonstrate incorrectness, the intent is to confuse the issue by hinting at the subtle interactions of units and means.

A machine is timed running two benchmark tests and receives the following scores:

test1:	3 sec	(most machines run it in 12 seconds)
test2:	300 sec	(most machines run it in 600 seconds)

How fast is the machine? Let us look at different ways of calculating performance:

Method 1—one way of looking at this is by the ratios of the running times:

$$test1\colon\ \frac{3}{12} \qquad test2\colon\ \frac{300}{600}$$

The machine's performance on test 1 is four times faster than an average machine, its performance on test 2 is twice as fast as average, therefore our machine is (on average) three times as fast as most machines.

Method 2—another way of looking at this is by the ratios of the running times:

$$test1\colon\ \frac{3}{12} \qquad test2\colon\ \frac{300}{600}$$

The machine's running time on test 1 is 1/4 the time it takes most machines, its running time on test 2 is 1/2 the time it takes most machines, so our machine (on average) takes 3/8 the time a typical machine does to run a program, or, put another way, our machine is 8/3 (2.67) times as fast as the average machine.

Method 3—yet another way of looking at this is by the ratios of the running times:

$$test1\colon\ \frac{3}{12} \qquad test2\colon\ \frac{300}{600}$$

The machine ran the benchmarks in a total of 303 seconds, the average machine runs the benchmarks in 612 seconds, therefore our machine takes 0.495 the amount of time to run the benchmarks as most machines do, and so is roughly twice as fast as the typical machine (on average).

---

1. Compare this to just one physical interpretation of the arithmetic mean; finding the center of gravity in a set of objects (possibly having different weights) placed along a see-saw. There are countless other interpretations which are just as intuitive and meaningful.

Method 4—and then you can always look at the ratios of the running times ...

How can these calculations seem reasonable and yet produce completely different results? The answer is that they *seem* reasonable because they *are* reasonable; they all give perfectly accurate answers, just not to the same question. Like in many other areas, the answers are not hard to come by—the difficult part is in asking the right questions.

2.0 The Semantics of Means

In general, there are a number of possibilities for finding the performance, given a set of experimental times and a set of reference times. One can take the average of

- the raw times,
- the raw rates (inverse of time)[1],
- the ratios of the times (experimental time over reference),
- or the ratios of the rates (reference time over experimental).

Each option represents a different question and as such gives a different answer; each has a different meaning as well as a different set of implications. An average need not be meaningless, but it may be if the implications are not true. If one understands the implications of averaging rates, times, and their ratios, then one is less apt to wind up with meaningless information.

THE SEMANTICS OF TIME, RATE, AND RATIO

Remember the correspondence between the arithmetic and harmonic means:

$$ArithmeticMean\,(times) \leftrightarrow HarmonicMean\,(rates)$$

$$ArithmeticMean\,(rates) \leftrightarrow HarmonicMean\,(times)$$

The Semantics of Time

A set of **times** is a collection of numbers representing Time Taken per Unit Somethings Accomplished. The information contained in their arithmetic mean is therefore On Average, How Much Time is Taken per Unit Somethings Accomplished; the average amount of time it takes to accomplish a prototypical task.

“On Average” in this case is defined across Somethings and not Time. For example, a book is read in two hours, another in four; the average is 3 hours per book. If books similar to these are read continuously one after another and the reader’s progress is sampled in *time* (say once every minute) then the value of 4 hrs/book will come up twice as often as the value of 2 hrs/book, giving an incorrect average of 10/3 hours per book. However, if the reading time is sampled per *book* (say once every book), the average will come out correctly.

Time is what we are concerned with in omparing the performance of computers. Though it is just as important a measure of performance, we are not concerned with throughput since juggling both would confuse the point. In this paper we want to know how long it takes to perform a task, rather than how many tasks the machine can perform per unit time. If the set of times is taken from representative programs, then the average will be an accurate predictor of how long a typical program would take, and thus indicate the machine’s performance.

The Semantics of Rate

A set of **rates** is in units of Somethings Accomplished per Unit Time, and the information contained in their arithmetic mean is then On Average, How Many Somethings You Can Expect to Accomplish per Unit Time. Here, the average is

---

1. We will use the word *rate* to describe a unit where time is in the denominator despite what may be in the numerator (unless it is also time, in which case the unit is a pure number). Time and rate are related in that the arithmetic mean of one is the inverse of the harmonic mean of the other.

defined across Time and not Somethings; if you intend to take the arithmetic mean of a set of rates, the rates should represent instantaneous measures taken in Time and should **NOT** represent measurements taken for every Something Accomplished.

Take the above book example; if we try to average 1/2 book per hour and 1/4 book per hour (the values obtained if we sample over *books*), we obtain a measurement of 3/8 books per hour; what good is this information? It cannot be combined with the number of books we read to produce how long it should have taken (it took 6 hours, not 16/3 hours). This confusion arises because of an incorrect use of the arithmetic mean.

If, however, we sample the reading rate at periodic points in *time*, we find that there will be twice as many values of 1/4 book per hour as 1/2 book per hour, which will give us (1/4 + 1/4 + 1/2, divided by 3); an average rate of 1/3 book per hour, corresponding nicely with reality.

When measuring computers, we are generally presented with a set of values taken per task completed—a set of benchmark results, each the time taken to perform one of several tests—**not** a set of instantaneous measurements of progress, sampled every unit of time. Therefore, in general, finding the arithmetic mean of a set of rates is not a good idea, as it will lead to erroneous and misleading results. Use the harmonic mean instead.

The Semantics of Ratios

Computer performance is often represented by a **ratio** of rates or times. It is a unitless number, and when the reference time is in the numerator (as in a ratio of rates) the measurement means how much "faster" one thing is than another. When the reference time is in the denominator (as in a ratio of times) the measurement means what fraction of time the machine in question takes to perform a task, relative to the reference machine.

What does it mean to average a set of unitless ratios? The arithmetic mean of a set of ratios is a weighted average where the weights happen to be the running times of the reference machine. What information is contained in this value? If the reference times are thought of as the *expected* amount of time for each benchmark, the weighting might ensure that no benchmark result counts more than any other, and the arithmetic mean would then represent what proportion of the expected time the average benchmark takes.

3.0 Problems with Normalization

Problems arise if we take the average of a set of normalized numbers. The following examples demonstrate the errors that occur. The first example compares the performance of two machines, using a third as a benchmark. The second example extends the first to show the error in using CPI values to compare performance.

EXAMPLE I: *Average Normalized by Reference Times*

There are two machines, A and B, and a reference machine. There are two tests, T1 and T2, and we obtain the following scores for the machines:

Scenario I	**Test T1**	**Test T2**
Machine A:	10 sec	100 sec
Machine B:	1 sec	1000 sec
Reference:	1 sec	100 sec

In scenario I, the performance of machine A relative to the reference machine is 0.1 on test T1 and 1 on test T2. The performance of machine B relative to the reference machine is 1 on test T1 and 0.1 on test T2. Since *time* is in the denominator (the reference is in the numerator), we are averaging *rates*, therefore we use the harmonic mean. The fact that the reference value is also in units of time is irrelevant; the time measurement we are concerned with is in the denominator, thus we are averaging rates.

The performance results of Scenario I:

Scenario I	**Harmonic Mean**
Machine A:	HMean(0.1, 1) = 2/11
Machine B:	HMean(1, 0.1) = 2/11

The two machines perform equally well. This makes intuitive sense; on one test machine A was ten times faster, on the other test machine B was ten times faster. Therefore they should be of equal performance. As it turns out, this line of reasoning is erroneous.

Let us consider scenario II, where the only thing that has changed is the reference machine's times (from 100 seconds on test T2 to 10 seconds):

Scenario II	**Test T1**	**Test T2**
Machine A:	10 sec	100 sec
Machine B:	1 sec	1000 sec
Reference:	1 sec	10 sec

Here, the performance numbers for A relative to the reference machine are 1/10 and 1/10, the performance numbers for B are 1 and 1/100, and these are the results:

Scenario II	**Harmonic Mean**
Machine A:	HMean(0.1, 0.1) = 1/10
Machine B:	HMean(1, 0.01) = 2/101

According to this, machine A performs about 5 times better than machine B. And if we try yet another scenario changing only the reference machine's performance on test T2, we obtain the result that machine A performs *worse* than machine B.

Scenario III	**Test T1**	**Test T2**	**Harmonic Mean**
Machine A:	10 sec	100 sec	HMean(0.1, 10) = 20/101
Machine B:	1 sec	1000 sec	HMean(1, 1) = 1
Reference:	1 sec	1000 sec	

The lesson: do not average test results which have been normalized.

EXAMPLE II: ***Average Normalized by Number of Operations***

The example extends even further; what if the numbers were not a set of normalized running times but CPI measurements? Taking the average of a set of CPI values should not be susceptible to this kind of error, because the numbers are *not* unitless; they are not the ratio of the running times of two arbitrary machines.

Let us test this theory. Let us take the average of a set of CPI values, in three scenarios. The units are *cycles per instruction*, and since the time-related portion (cycles) is in the numerator, we will be able to use the arithmetic mean.

The following are the three scenarios, where the only difference between each scenario is the number of instructions performed in Test2. The running times for each machine on each test do not change, therefore we should expect the performance of each machine relative to the other to remain the same.

Scenario I	**Test1**	**Test2**	**Arithmetic Mean**
Machine A:	10 cycles	100 cycles	AMean(10, 10) = 10 CPI
Machine B:	1 cycle	1000 cycles	AMean(1, 100) = 50.5 CPI
Instructions:	1 instr	10 instr	**Result:** *Machine A faster*

Scenario II	**Test1**	**Test2**	**Arithmetic Mean**
Machine A:	10 cycles	100 cycles	AMean(10, 1) = 5.5 CPI

Machine B:	1 cycle	1000 cycles	AMean(1, 10) = 5.5 CPI
Instructions:	1 instr	100 instr	**Result:** *Equal performance*

Scenario III	**Test1**	**Test2**	**Arithmetic Mean**
Machine A:	10 cycles	100 cycles	AMean(10, 0.1) = 5.05 CPI
Machine B:	1 cycle	1000 cycles	AMean(1, 1) = 1 CPI
Instructions:	1 instr	1000 instr	**Result:** *Machine B faster*

However, we obtain the anomalous result that the machines have different relative performances which depend upon the number of instructions that were executed.

The theory is flawed. Average CPI values are not valid measures of computer performance. Taking the average of a set of CPI values is not inherently wrong, but the result cannot be used to compare performance. The erroneous behavior is due to normalizing the values before averaging them. If we average the running times before normalization, we get a value of 55 cycles for Machine A, and a value of 500.5 cycles for Machine B. This alone is the valid comparison. Again, this example is not meant to imply that average CPI values are meaningless, they are simply meaningless when used to compare the performance of machines.

ON SPECMARKS

We have demonstrated that the following is an erroneous method to find a performance number for a machine, based upon a set of test results.

$$AVG = \frac{\sum_{i}^{N} \frac{OurTime_i}{RefTime_i}}{N}$$

The implication is that ratios such as SPECmarks should never be averaged; they should first be converted back into the original time values or rate values, and then averaged. The following demonstrates the relation between this and SPEC.

AVG has been calculated using erroneous methods, and so it is a meaningless number. However, this meaningless number can be easily transformed into the harmonic mean of SPECmarks, as the following demonstrates:

$$\frac{1}{AVG} = \frac{N}{\sum_{i}^{N} \frac{OurTime_i}{RefTime_i}} = \frac{N}{\sum_{i}^{N} \frac{1}{SPECmark_i}} = HarmonicMean\,(SPECmarks)$$

To repeat, performance ratios should not be averaged. Normalized values should not be averaged. Individual SPECmarks, which are the ratio of the reference machine's running time to the test machine's running time, are normalized values. Their average is therefore meaningless. The only meaningful performance number is the ratio of the arithmetic means of the reference and test machines' running times.

4.0 The Meaning of Performance

We have determined that the arithmetic mean is appropriate for averaging times (which implies that the harmonic mean is appropriate for averaging rates), and that Normalization, if performed, should be carried out *after* the averaging. The question arises: *what does this mean?*

When we say that the following describes the performance of a machine based upon the running of a number of standardized tests (which is the ratio of the arithmetic means, with the constant *N* terms cancelling out),

$$\frac{\sum_{i}^{N} OurTime_i}{\sum_{j}^{N} RefTime_j}$$

then we implicitly believe that every test counts equally, in that on average it is used the same *number* of times as all other tests. This means that tests which are much longer than others will count more in the results.

POINT OF VIEW: ***Performance is Time Saved***

We wish to be able to say, "*this* machine is X times faster than *that* machine." Ambiguity arises because we are often unclear on the concept of performance. What do we mean when we talk about the performance of a machine? Why do we wish to be able to say *this machine is X times faster than that machine*? The *reason* is that we have been using *that* machine (machine A) for some time and wish to know how much time we would save by using *this* machine (machine B) instead.

How can we measure this? First, we find out what programs we tend to run on machine A. These programs (or ones similar to them) will be used as the benchmark suite to run on machine B. Next, we measure how often we tend to use the programs. These values will be used as weights in computing the average (programs used more should count more), but the problem is that it is not clear whether we should use values in units of time or number of occurrences; do we count each program the number of times per day it is used or the number of hours per day it is used?

We have an idea about how often we use programs; for instance, every time we edit a source file we might recompile. So we would assign equal weights to the word processing benchmark and the compiler benchmark. We might run a different set of 3 or 4 n-body simulations every time we recompiled the simulator; we would then weight the simulator benchmark 3 or 4 times as heavily as the compiler and text editor. Of course, it is not quite as simple as this, but you get the point; we tend to know how often we use a program, independent of how slowly or quickly the machine we use performs it.

What does this buy us? Say for the moment that we consider all benchmarks in the suite equally important (we use each as often as the other); all we need to do is total up the times it took the new machine to perform the tests, total up the times it took the reference machine to perform the tests, and compare the two results.

It does not matter if one test takes three minutes and another takes three days—if the reference machine performs the short test in less than a second (indicating that your new machine is *extremely* slow) and it performs the long test in three days and six hours (indicating that your new machine is marginally faster than the old one), the *time saved* is about six hours. Even if you use the short program a hundred times as often as the long program, the time saved is still an hour over the old machine.

The error is that we considered performance to be a value which can be averaged; the problem is our perception that performance is a simple number. The reason for the problem is that we often forget the difference between the following statements:

- on average, the amount of time saved by using machine A over machine B is ...
- on average, the relative performance of machine A to machine B is ...

HOW WRONG IS WRONG: ***Performance Comparisons of 7 High-Profile Computers***

What effect does this have upon performance calculations, besides being wrong? How wrong is it? Presented in the following figures are comparisons of machine performances, with the performance numbers calculated according to the geometric mean, the harmonic mean, and the arithmetic mean.

The number produced by the geometric mean is the number published as the computer's SPEC rating. It is found by taking the geometric mean of the SPEC ratios. It is a meaningless number. The number produced by the harmonic mean is simply the harmonic mean of the SPEC ratios; it, too, is a meaningless number. The final number is produced the correct way, by deriving the original time measurements from the SPECmark and the published runnings times for the

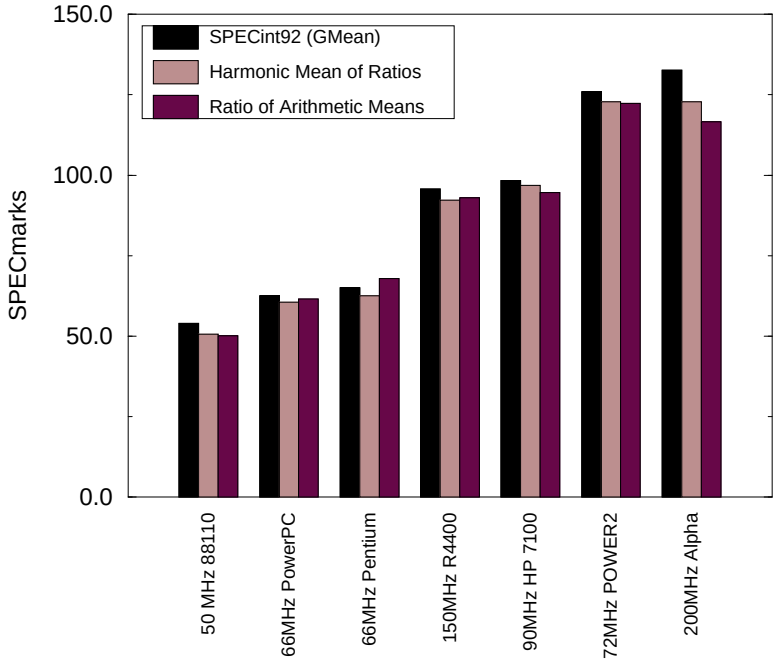

Figure 1. Comparative SPECmarks, Integer
A number of published SPECmarks are shown, compared to the values recomputed using the harmonic mean on the individual SPECmarks and the arithmetic mean on the individual running times. As neither of the means were computed with any weight information, all tests are weighted equally. Only the Ratio of Arithmetic Means is correct.

VAX 11/780. These numbers are averaged with the arithmetic mean and compared to the arithmetic mean of the VAX.

The numbers are shown in Fig. 1 and Fig. 2, showing the difference between using various appropriate and inappropriate methods. The published SPECint92 and SPECfp92 numbers are to the left, the value recomputed using the harmonic mean is in the middle, and the value recomputed with the raw running times is on the right. The differences are on the order of ten percent; not enormous, but certainly enough to reorder the list if the examples chosen had been clustered together. As it is, the 72MHz POWER2 chip turns out to be faster than the 200MHz Alpha in both integer and floating point when the averages are recomputed. The numbers are taken from [Corp93a] and [Corp93b].

5.0 Rethinking Performance

We usually know what we need to do; we are interested in how much of it we can get done with *this* computer versus *that* one. In this context, the only thing that matters is how much time is saved by using one machine over another. The fallacy is in considering performance a measure unto itself. Performance is in reality a specific instance of the following:

- two machines,
- a set of programs to be run on them,
- and an indication of how important each of the programs is to *us*.

Performance is therefore not a single number, but really a collection of implications. It is nothing more or less than the measure of how much time *we* save running *our* tests on the machines in question. If someone else has similar needs to ours, our performance numbers will be useful to them. However, two people with different sets of criteria will likely walk away with two completely different performance numbers for the same machine.

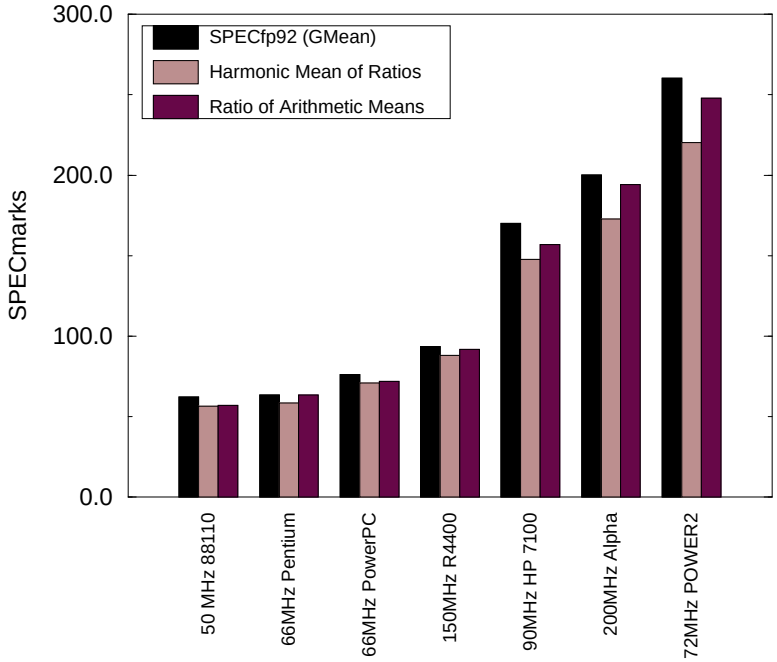

Figure 2. Comparative SPECmarks, Floating Point
A number of published SPECmarks are shown, compared to the values recomputed using the harmonic mean on the individual SPECmarks and the arithmetic mean on the individual running times. As neither of the means were computed with any weight information, all tests are weighted equally. Only the Ratio of Arithmetic Means is correct.

6.0 Summary

Interpretations of the arithmetic, geometric, and harmonic means have been given, with the geometric mean written off as a curiosity. Numerous examples have illustrated the reasons for using different means in different circumstances, with an attempt to give insight into the semantics of the various choices. The primary results include the following:

RULES TO LIVE BY

1. When presented with a number of *times* for a set of benchmarks, the appropriate average is the *arithmetic mean*.
2. When presented with a number of *rate ratios* for a set of benchmarks (reference time over experimental time, such as in SPECmarks), sum the individual running times and use the ratio of the sums (equivalent to the ratio of the *arithmetic means*).
3. When presented with a number of *time ratios* for a set of benchmarks (experimental time over reference time), sum the individual running times and use the ratio of the sums (equivalent to the ratio of the *arithmetic means*).
4. When presented with a set of *rates*, first determine if they are per benchmark or sampled periodically in time. If per benchmark (which is more likely), the *harmonic mean* is appropriate; if sampled in time, the *arithmetic mean* is appropriate.

References

[Corp93a] Standard Performance Evaluation Corp. SPEC Newsletter, September 1993.

[Corp93b] Standard Performance Evaluation Corp. SPEC Newsletter, December 1993.

[Fleming86] Philip J. Fleming and John J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *CACM*, 29(3):218–221, March 1986.

[Smith88] James E. Smith. Characterizing Computer Performance with a Single Number. *CACM*, 31(10):1202–1206, October 1988.

Notes on Calculating Computer Performance

Bruce Jacob and Trevor Mudge
Advanced Computer Architecture Lab
EECS Department, University of Michigan
{blj,tnm}@umich.edu

Abstract

This report explains what it means to characterize the performance of a computer, and which methods are appropriate and inappropriate for the task. The most widely used metric is the performance on the SPEC benchmark suite of programs; currently, the results of running the SPEC benchmark suite are compiled into a single number using the geometric mean. The primary reason for using the geometric mean is that it preserves values across normalization, but unfortunately, it does not preserve total run time, which is probably the figure of greatest interest when performances are being compared.

Cycles per Instruction (CPI) is another widely used metric, but this method is invalid, even if comparing machines with identical clock speeds. Comparing CPI values to judge performance falls prey to the same problems as averaging normalized values.

In general, normalized values must not be averaged and instead of the geometric mean, either the **harmonic** or the **arithmetic** mean is the appropriate method for averaging a set running times. The arithmetic mean should be used to average times, and the harmonic mean should be used to average rates (1/time). A number of published SPECmarks are recomputed using these means to demonstrate the effect of choosing a favorable algorithm.

University of Michigan Tech Report CSE-TR-231-95
March, 1995