HAL
open science

Demystifying multicore throughput metrics

Pierre Michaud

To cite this version:

Pierre Michaud. Demystifying multicore throughput metrics. IEEE Computer Architecture Letters, 2012, pp.ISSN: 1556-6056. 10.1109/L-CA.2012.25 . hal-00737044

HAL Id: hal-00737044

https://hal.science/hal-00737044v1

Submitted on 1 Oct 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Demystifying Multicore Throughput Metrics

Pierre Michaud
INRIA Rennes
Rennes, France
Pierre.Michaud@inria.fr

Abstract—Several different metrics have been proposed for quantifying the throughput of multicore processors. There is no clear consensus about which metric should be used. Some studies even use several throughput metrics. We show that there exists a relation between single-thread average performance metrics and throughput metrics, and that throughput metrics inherit the meaning or lack of meaning of the corresponding single-thread metric. We show that two popular throughput metrics, the weighted speedup and the harmonic mean of speedups, are *inconsistent*: they do not give equal importance to all benchmarks. Moreover we demonstrate that the weighted speedup favors unfairness. We show that the harmonic mean of IPCs, a seldom used throughput metric, is actually consistent and has a physical meaning. We explain under which conditions the arithmetic mean or the harmonic mean of IPCs can be used as a strong indicator of throughput increase.

Index Terms—Computer architecture, performance metric, multicore throughput

1 Introduction

One of the most controversial issue in computer architecture studies is how to quantify the performance of a computer [2], [9], [5], [7], [1], [4]. Typically, we try to find out which of two machines is likely to execute programs faster. In computer architecture studies, it is not rare that some benchmarks run faster on one machine while other benchmarks run faster on the other machine. In such situation, depending on how we summarize the performance on the benchmark set, we might obtain different conclusions. Since the introduction of multicores, researchers use multiprogram workloads to quantify throughput. A multiprogram workload is a set of *independent* programs running simultaneously on a multiprocessor machine, e.g., a multicore. This paper considers only *single-threaded* programs (also called *threads* in the remaining). Because of resource sharing (cache, bus, ...), the execution time of a thread may be influenced by the other threads running concurrently. Several different metrics have been proposed for quantifying throughput. Some papers even use several such metrics. However, the meaning of some of these metrics is not very clear.

The goal of this paper is to clarify the meaning of multicore throughput metrics. We show in Section 3 that, under certain conditions, there is a relation between throughput metrics and single-thread performance metrics. We first recall in Section 2 the meaning of popular single-thread metrics. Among these metrics, we distinguish those that treat all the benchmarks equally, which we call *consistent* metrics. We show in Section 3 that the popular *weighted speedup* and harmonic mean of speedups are inconsistent throughput metrics. Moreover, we demonstrate in Section 4 that the weighted speedup actually favors unfairness. We propose that consistent throughput metrics be used, like the arithmetic mean of IPCs, the harmonic mean of IPCs, or the geometric mean of speedups. We explain in Section 5 under which conditions the arithmetic mean of IPCs or the harmonic mean of IPCs provide a strong indication of throughput increase.

Manuscript submitted: 18-Apr-2012. Manuscript accepted: 24-Jul-2012. Final manuscript received: 26-Jul-2012.

2 Single-thread average performance metrics

A performance metric P is a mathematical formula giving a single global performance number for a machine from a set of per-benchmark performance numbers. We consider n benchmarks numbered from 1 to n. The performance of a machine is $P(IPC_1, \cdots, IPC_n)$, where IPC_j is the number of instructions executed per clock cycle [1] by benchmark j. Such IPC-based metric ignores the total execution time of benchmarks. Weighting benchmarks according to their respective execution times would be like assuming that the benchmarks themselves, and their inputs, are representative of "random" programs. Actually, the performance metrics used in computer architecture studies generally do not consider the total execution time of benchmarks. Often, we are not interested by a benchmark itself but by its *behavior*, which we hope is representative of some behaviors that can be found in random programs. The IPC, for instance, better characterizes a benchmark's behavior than the total execution time. Ideally, one would like to weight the different behaviors according to how important they are in real life situations. But this would require a careful and extensive workload characterization analysis [7], or at least a justification. In practice, without any information on the representativeness of benchmarks, most people assume that all the benchmarks are equally important. To be in accordance with this assumption, a performance metric should be *consistent*.

We define a *consistent* single-thread IPC-based performance metric P as a metric such that, for any permutation σ on $[1, n]$,

$$P(IPC_1, \cdots, IPC_n) = P(IPC_{\sigma(1)}, \cdots, IPC_{\sigma(n)})$$

In words, a consistent metric considers all the benchmarks equally important, as one can permute their IPCs and still obtain the same performance. Table 1 lists five single-thread metrics and illustrate their consistency or non-consistency on an example with two machines and two benchmarks. Assuming the two benchmarks are equally important, it is objectively impossible to say that one machine outperforms the other.

1. In this paper we consider only IPC-based metrics. To compare machines with different or varying clock frequencies, the "C" in "IPC" should be a fixed reference clock cycle.

TABLE 1
Some single-thread average performance metrics (speedups are relative to machine X).

	machine X	machine Y	speedup Y/X
benchmark A	$IPC = 1$	$IPC = 2$	2
benchmark B	$IPC = 2$	$IPC = 1$	0.5
metric	**perf. X**	**perf. Y**	**consistent ?**
A-mean of speedups	1	1.25	no
H-mean of speedups	1	0.8	no
G-mean of speedups	1	1	yes
A-mean of IPCs	1.5	1.5	yes
H-mean of IPCs	1.33	1.33	yes

Therefore, a consistent metric must reflect this fact and give the same performance to both machines. Fleming and Wallace made a case for consistency, discouraging the use of metrics that depend on the choice of a particular reference machine [2]. But inconsistent metrics are still being used today in some studies. Our definition of consistency is more or less the same, but expressed in a way which, we hope, is more convincing.

The arithmetic mean (A-mean) and harmonic mean (H-mean) of speedups are both inconsistent metrics. However, the geometric mean (G-mean) of speedups is always consistent, whatever the machine used as reference for defining speedups [2]. The meaning of the G-mean has been questioned by some authors [9], [5]. The meaning of a performance metric always depends on some assumptions, and different metrics rely on different assumptions. The G-mean of speedups gives an estimate of the median speedup of random programs assuming speedups are distributed log-normally [7], [4].

The A-mean and the H-mean of IPCs are both consistent metrics whose meaning is based on different assumptions.

The A-mean of IPCs is proportional to the total quantity of work done when executing X seconds of each benchmark consecutively on the machine considered. It is equivalent to assuming that, when the machine runs random programs, its IPC at any instant is equal to one of the benchmarks IPCs, all benchmarks IPCs being equally likely. This implies that the total quantity of work done by a program is variable and adjusted as a function of the machine speed.

The H-mean of IPCs is inversely proportional to the total time it takes to execute N instructions from each benchmarks consecutively. An implicit assumption is that the IPC of a random program is not correlated with the number of instructions executed by that program. We could also consider the G-mean of IPCs, but it is *equivalent* to the G-mean of speedups (two performance metrics are equivalent if they define the same ranking on any set of machines).

3 Multicore throughput metrics

In this paper, we consider multicore throughput metrics intended for comparing multicores with the same fixed number k of logical cores (physical cores may be SMT [11]). In computer architecture studies, the most frequent method for evaluating throughput considers a fixed and predefined set of *workloads*, where each workload is a combination of k threads running simultaneously, each thread being taken from a set of n benchmarks. For instance, assuming the cores are identical and that a benchmark may occur several times in a workload, there are $\binom{n+k-1}{k}$ possible distinct workloads. In practice, because simulators are slow, most researchers work with a relatively small subset of w workloads.

Let $B_i[1], \cdots, B_i[k]$ be the k threads constituting the ith workload, a thread being one of the n benchmarks. For each workload, we obtain k IPC values $IPC_i[1], \cdots, IPC_i[k]$. In total for the w workloads, we obtain $w \times k$ IPC values. A global throughput metric reduces these $w \times k$ IPC values to a single throughput value. A common practice is to define a per-workload throughput T_i and then compute the global throughput as an average over all T_i's:

$$T_{global} = \underset{i \in [1,w]}{\text{X-mean}} \; T_i \tag{1}$$

where X-mean can be the A-mean, H-mean or G-mean.

The throughput metrics that have been used most frequently in computer architecture studies are the IPC throughput, the *weighted speedup* [10] and the H-mean of speedups [6]. These metrics can be generalized and divided in two categories: *weighted* and *unweighted* throughput metrics.

3.1 Weighted throughput metrics

Weighted throughput metrics like the *weighted speedup* and the H-mean of speedups are the most commonly used throughput metrics today. They compute the per-workload throughput as an average speedup:

$$T_i = \underset{j \in [1,k]}{\text{X-mean}} \; \frac{IPC_i[j]}{IPC_{ref}[B_i[j]]} \tag{2}$$

where X-mean may be the A-mean, H-mean or G-mean, and $IPC_{ref}[b]$ is a reference IPC for benchmark b (very often, the IPC for the benchmark running alone on a reference machine). Using the A-mean in formula (2) yields a metric equivalent to the weighted speedup [10], provided k is fixed[2]. Formula (2) with the H-mean is also an oft-used throughput metric [6].

Many authors use different X-means in formulas (2) and (1), without justification. For instance, the G-mean is often used for computing the global throughput (formula (1)), while it is rarely used for the per-workload throughput (formula (2)). However, it makes sense to use the same X-mean for the global throughput and for the per-workload throughput.

A property of the A-mean, H-mean and G-mean, is that, if we consider a set of values partitioned into equally-sized subsets, the mean on the set can be obtained by computing the mean for each subset, then the mean of all subsets means. Therefore, the global throughput can be computed as

$$T_{global} = \underset{(i,j)}{\text{X-mean}} \; \frac{IPC_i[j]}{IPC_{ref}[B_i[j]]} \tag{3}$$

If we consider that the benchmarks are equally important, we must define the w workloads in such a way that each of the n benchmarks contributes an equal number of values in the $w \times k$ IPC values. That is, for all $b \in [1, n]$,

$$\frac{w \times k}{n} = \sum_{(i,j), B_i[j]=b} 1 \tag{4}$$

If this condition holds, the global thoughput (3) can be viewed as a single-thread average performance metric

$$T_{global} \quad = \quad \underset{b \in [1,n]}{\text{X-mean}} \; P_b$$

2. The weighted speedup is k times the A-mean of speedups.

TABLE 2
The A-mean and H-mean of speedups are both inconsistent throughput metrics

	single thread	machine X running AB	machine Y running AB
benchmark A	$IPC = 1$	$IPC = 0.5$	$IPC = 1$
benchmark B	$IPC = 2$	$IPC = 1$	$IPC = 0.5$
A-mean of speedups		0.5	0.625
H-mean of speedups		0.5	0.4
G-mean of speedups		0.5	0.5

where per-benchmark performance P_b is defined as

$$P_b = \frac{\underset{(i,j),B_i[j]=b}{\text{X-mean}} \; IPC_i[j]}{IPC_{ref}[b]}$$

The throughput metric (3) inherits some properties and meaning (or lack of meaning) of the corresponding single-thread metric. In particular, using the A-mean or H-mean in formula (3) leads to a non-consistent metric, as illustrated by the example shown in Table 2. This example considers a single workload of two benchmarks A and B. According to the A-mean of speedups, machine Y offers 25% more throughput than machine X, whereas according to the H-mean of speedups it is machine X that offers 25% more throughput than machine Y. The A-mean gives more weight to the benchmark with a lower single-thread IPC, while the H-mean gives more weight to the benchmark with a higher single-thread IPC. The only conclusion consistent with the assumption that benchmarks A and B are equally important is the one given by the G-mean: the two machines offer the same throughput.

3.2 Unweighted throughput metrics

Unweighted throughput metrics use only raw IPC values. We still assume that the same X-mean is used for per-workload throughput and for global throughput. Consequently, the global throughput can be computed as

$$T_{global} = \underset{(i,j)}{\text{X-mean}} \; IPC_i[j] \tag{5}$$

where X-mean may be the A-mean or the H-mean (the G-mean of IPCs is equivalent to the G-mean of speedups). Under condition (4), the global throughput can be viewed as a single-thread average performance metric where the per-benchmark performance P_b is defined as

$$P_b = \underset{(i,j),B_i[j]=b}{\text{X-mean}} \; IPC_i[j]$$

As their single-thread counterparts, the unweighted throughput metrics (5) are consistent: one can permute the benchmarks IPCs without changing the global performance. Table 3 provides an example of computing throughput as the A-mean of IPCs, assuming two cores and three benchmarks. The table also shows the per-benchmark performance P_b. Providing P_b for each benchmark individually is another way to summarize performance, not as concise as global throughput, but potentially more informative.

The A-mean of IPCs is equivalent to the sum of IPCs, aka the *IPC throughput*. The IPC throughput used to be the prevalent metric for studying SMT microarchitectures in the 1990's [11]. It has been criticized for not conveying any notion of *fairness* [10], [8], [6]. This led to the definition of other throughput metrics like weighted speedup and H-mean of speedups. Throughput and fairness are two different notions [6], [12]. The IPC throughput is really a *throughput* metric, and a consistent one, unlike the weighted speedup and the H-mean of speedups. It represents the total quantity of work done when all the threads execute for an equal duration. This corresponds to a situation where the quantity of work per thread is variable and adjusted as a function of the machine speed.

TABLE 3
Example of computing throughput as the A-mean of IPCs on a dual-core processor using three benchmarks A,B,C

workload	IPC "left" core	IPC "right" core
AA	1	1
BB	2	2
CC	0.5	0.5
AB	1.3	1.8
BC	1.2	0.6
CA	0.8	1.1

benchmark	per-benchmark performance
A	$P_A = (1+1+1.3+1.1)/4 = 1.10$
B	$P_B = (2+2+1.8+1.2)/4 = 1.75$
C	$P_C = (0.5+0.5+0.6+0.8)/4 = 0.60$

global throughput
$T_{global} = (P_A + P_B + P_C)/3 = 1.15$

The H-mean of IPCs is more rarely used to quantify throughput. Yet, it is a consistent metric with a physical meaning, which can be seen as follows. We consider a job schedule where all the jobs execute the same fixed number N of instructions. Let $IPC[i,j]$ be the IPC of the ith job executed on core j. Under perfect load balancing, the quantity $\sum_i \frac{N}{IPC[i,j]}$ is the same for all j and is equal to the total time t, hence $kt = \sum_{(i,j)} \frac{N}{IPC[i,j]}$. The number of jobs executed per cycle is

$$\frac{\sum\limits_{(i,j)} 1}{t} = \frac{k}{N} \times \frac{\sum\limits_{(i,j)} 1}{\sum\limits_{(i,j)} \frac{1}{IPC[i,j]}} = \frac{k}{N} \times \underset{(i,j)}{\text{H-mean}} \; IPC[i,j]$$

The throughput metric (5) using the H-mean is proportional to the job throughput, assuming $IPC_i[j]$ and $IPC[i,j]$ are distributed identically (that is, assuming the fixed workloads are representative of the job schedule).

4 The weighted-speedup favors unfairness

The weighted speedup and the H-mean of speedups have been introduced with the intent to take fairness into account. Luo et al. have stated that the H-mean of speedups captures "some effect of the lack of fairness". This statement can be made more precise. Let us consider the case where the speedups S_i are all close to one, let's say in the $[0.5, 1.5]$ range. A second-order multivariate Taylor series expansion about $S_i = 1$ gives

$$\text{A-mean } S_i = \mu$$

$$\text{H-mean } S_i \approx \mu - \sigma^2 \tag{6}$$

$$\text{G-mean } S_i \approx \mu - \frac{\sigma^2}{2} \tag{7}$$

where $\sigma^2 = \text{A-mean } (S_i - \mu)^2$ is the variance. The standard deviation of speedups, σ, is considered a measure of unfairness [6], [12]. So it is true that the H-mean of speedups captures *some effect* of the lack of fairness, as it penalizes machines that

degrade fairness. But the same could be said about the G-mean of speedups. Yet, the G-mean of speedups ranks machines exactly like the G-mean of IPCs, which does not bear any notion of fairness. There is no paradox here: the G-mean of speedups is indeed neutral with respect to fairness, it is the weighted speedup which actually favors unfairness. The example in Table 2 illustrates this fact, which may surprise some people, considering the reasons usually advanced for using the weighted speedup.

5 Strong indicators of throughput increase

Attempts to embed notions of throughput and fairness in a single metric lead to metrics whose physical meaning is uncertain. Such metrics have been proposed for not having to provide two numbers, one for throughput and one for fairness. But it is frequent practice to use two *throughput* metrics to show the robustness of the conclusions. Some studies even use three different throughput metrics. Instead, we recommend using *true* fairness metrics when fairness is important, and *consistent* throughput metrics such as the A-mean of IPCs, H-mean of IPCs, or G-mean of speedups.

A question is, which metric to use? Choosing a metric is choosing the assumptions behind this metric. We may want to use several metrics. Nevertheless, it is not necessary to use all three metrics. If we increase both the A-mean and the H-mean of IPCs, we quite likely increase the G-mean too. Indeed, from approximations (6) and (7), and assuming they hold for IPC values[3], the G-mean is approximately the mid point between the A-mean and H-mean. If both the A-mean and H-mean increase, so does the G-mean.

If we prefer to use a single throughput metric, it is useful to have an idea of whether the variance of IPCs is increased or decreased. When the variance of IPCs is decreased, the A-mean of IPCs is a *strong indicator of throughput increase* (SITI), meaning that an increase of the A-mean implies an increase of the H-mean (and G-mean). Conversely, when the variance of IPCs is increased, the H-mean of IPCs is a SITI, as an increase of the H-mean implies an increase of the A-mean (and G-mean).

For example, consider the situation where we compare a multicore with private last-level caches, and one with a shared cache, the total cache capacity being the same. Programs having a high IPC with a private cache probably do not suffer from cache misses, and sharing the cache can only decrease their IPC. Programs whose IPC is low because the private cache is too small are more likely to benefit from the shared cache. The variance of IPCs is smaller with the shared cache. Hence, the A-mean of IPCs provides a strong indication that a shared cache increases throughput.

As another example, let us consider a multicore hitting the thermal limit, such that automatic thermal throttling triggers and decreases performance. Because the power density in a core depends on the characteristics of the thread running on that core, not all cores may be hitting the thermal limit. Hence we may want to implement activity migration and migrate a hot thread to a cold core whenever the thermal limit is reached [3]. In general, power density, hence temperature, increases with the IPC. Activity migration increases the IPC of threads suffering from thermal throttling, which have an already high IPC. Threads with a low IPC and that do not suffer from thermal throttling do not benefit from activity migration, and their average IPC decreases because of the migration penalty. So activity migration increases the variance of IPCs. If we want a strong indication that activity migration increases throughput, we should use the H-mean of IPCs.

3. If IPCs are not close to 1 but if a division by a common factor α makes them all close to 1, approximations similar to (6) and (7) can be obtained by replacing σ^2 with σ^2/α.

6 Conclusion

Because it is generally very difficult to justify weighting different benchmarks differently, researchers usually assume that all the benchmarks are equally important. To be coherent with this assumption, we should use *consistent* metrics that give the same weight to all benchmarks. We showed that there exists a relation between multicore throughput metrics and single-thread average performance metrics, provided the same type of mean is used for defining the throughput of each workload and for summarizing the throughput on several workloads. Under this condition, a throughput metric inherits the properties and meaning (or lack of meaning) of the corresponding single-thread metric. In particular, some commonly used throughput metrics, like the weighted speedup and the harmonic mean of speedups, are not consistent. Moreover, we demonstrated that, contrary to popular belief, the weighted speedup actually favors unfairness. On the other hand, the geometric mean of speedups, the arithmetic mean and the harmonic mean of IPCs, are three consistent throughput metrics whose physical meaning relies on different assumptions. We showed that, in situations where the variance of IPCs is decreased, the arithmetic mean of IPCs provides a strong indication of throughput increase (SITI), whereas when the variance is increased, a SITI is obtained with the harmonic mean. We considered solely single-threaded programs in this study. To the author's knowledge, defining meaningful throughput metrics generalizing to parallel programs and usable in computer architecture studies is still an open problem.

References

[1] D. Citron, A. Hurani, and A. Gnadrey, "The harmonic or geometric mean : Does it really matter ?" *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 4, Sep. 2006.

[2] P. J. Fleming and J. J. Wallace, "How not to lie with statistics : The correct way to summarize benchmark results," *Commun. ACM*, vol. 29, no. 3, pp. 218–221, Mar. 1986.

[3] M. Gomaa, M. D. Powell, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system," in *ASPLOS*, 2004.

[4] M. F. Iqbal and L. K. John, "Confusion by all means," in *Workshop on Unique Chips and Systems (UCAS)*, 2010.

[5] L. K. John, "More on finding a single number to indicate overall performance of a benchmark suite," *ACM SIGARCH Comput. Archit. News*, vol. 32, no. 1, Mar. 2004.

[6] K. Luo, J. Gummaraju, and M. Franklin, "Balancing throughput and fairness in SMT processors," in *ISPASS*, 2001.

[7] J. R. Mashey, "War of the benchmark means : Time for a truce," *ACM SIGARCH Comput. Archit. News*, vol. 32, no. 4, Sep. 2004.

[8] Y. Sazeides and T. Juan, "How to compare the performance of two SMT microarchitectures," in *ISPASS*, 2001.

[9] J. E. Smith, "Characterizing computer performance with a single number," *Commun. ACM*, vol. 31, no. 10, pp. 1202–1206, Oct. 1988.

[10] A. Snavely and D. M. Tullsen, "Symbiotic jobscheduling for a simultaneous multithreading architecture," in *ASPLOS*, 2000.

[11] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multithreading : Maximizing on-chip parallelism," in *ISCA*, 1995.

[12] H. Vandierendonck and A. Seznec, "Fairness metrics for multi-threaded processors," *IEEE CAL*, vol. 10, no. 1, Jan. 2011.