Restating the Case for Weighted-IPC Metrics to Evaluate Multiprogram Workload Performance

Stijn Eyerman Lieven Eeckhout
Ghent University, Belgium

Abstract—Weighted speedup is nowadays the most commonly used multiprogram workload performance metric. Weighted speedup is a weighted-IPC metric, i.e., the multiprogram IPC of each program is first weighted with its isolated IPC. Recently, Michaud questions the validity of weighted-IPC metrics by arguing that they are inconsistent and that weighted speedup favors unfairness [4]. Instead, he advocates using the arithmetic or harmonic mean of the raw IPC values of the programs in the multiprogram workload. We show that weighted-IPC metrics are not inconsistent, and that weighted speedup is fair in giving equal importance to each program. We argue that, in contrast to raw-IPC metrics, weighted-IPC metrics have a system-level meaning, and that raw-IPC metrics are affected by the inherent behavior of the programs. We also show that the choice of a metric may adversely affect the conclusions from an experiment. We suggest to use two weighted-IPC metrics—system throughput (STP) and average normalized turnaround time (ANTT)—for evaluating multiprogram workload performance, and to avoid raw-IPC metrics.

1 Introduction

Metrics for evaluating the performance of multiprogram workloads on multicore or multithreaded processors have been subject of a long and still lasting debate. Early studies used the sum of the IPCs of each program [6], but Snavely and Tullsen [5] argued that this metric favors high-IPC programs and proposed weighted speedup instead. Later, Luo et al. [3] proposed the harmonic mean of the speedups instead of the sum or the arithmetic mean. Because the harmonic mean tends to be lower when there is much variance, it incorporates a notion of fairness. Both metrics are weighted-IPC metrics, i.e., the IPC of each program in the multiprogram experiment is first divided by the IPC of that program when run in isolation.

Since then, there seems to have risen some agreement on the use of weighted-IPC metrics. Today, about 75% of the papers that evaluate multiprogram performance use weighted speedup (sample from the HPCA, ASPLOS, ISCA and MICRO conferences in 2012). About 20% also evaluate the harmonic mean of speedups. 20% still use the sum of IPCs, but always in combination with the weighted speedup metric.

Recently, Michaud [4] questions the consistency and validity of weighted-IPC metrics. He argues that speedup metrics (including weighted speedup and harmonic mean of speedups) are inconsistent and should therefore not be used. Instead, he advocates the use of raw IPC values, instead of weighted IPC values, and he proposes to aggregate the raw IPCs using either the arithmetic or the harmonic mean. He claims that these metrics are a stronger *indicator of throughput increase* than the weighted metrics. In this paper, we challenge this claim, and restate the case for the weighted-IPC metrics.

The next section gives an overview of the commonly used multiprogram workload performance metrics. Sections 3 and 4 challenge Michaud's claims that the weighted-IPC metrics lack consistency and favor unfairness. Section 5 discusses the intuition of weighted-IPC metrics compared to raw-IPC metrics. Section 6 shows results from real experiments and concludes that the choice of the performance metric does matter in practical situations. Finally, we conclude in Section 7.

. Manuscript submitted: 01-Mar-2013. Manuscript accepted: 31-Mar-2013. Final manuscript received: 09-Apr-2013.
Date of publication 05 May. 2013; date of current version 31 Dec. 2014.
Digital Object Identifier 10.1109/L-CA.2013.9

2 Multiprogram performance metrics

There are two categories of multiprogram workload performance metrics: the raw-IPC metrics, which use the raw IPCs of the benchmarks in the multiprogram experiment, versus weighted-IPC metrics, which divide the multiprogram IPCs by the isolated IPCs.

Let IPC_i^{MP} be the IPC of program i in the multiprogram experiment. The raw-IPC metrics are defined as

$$\text{perf} = f(IPC_i^{MP}), \tag{1}$$

where f can be the sum, arithmetic mean, the harmonic mean or the geometric mean.

Weighted-IPC metrics on the other hand first weight the IPC of a program with its isolated IPC, i.e., its IPC when run alone on the system or its single-program IPC, IPC_i^{SP}:

$$\text{perf} = f(\frac{IPC_i^{MP}}{IPC_i^{SP}}). \tag{2}$$

The function f can again be the sum, arithmetic mean, the harmonic mean or the geometric mean (although the latter is not common). Weighted speedup [5] uses the sum; Luo et al. [3] propose to use the harmonic mean.

In our own prior work [2], we argued that weighted speedup can be interpreted as a measure for system throughput (STP), i.e., the number of jobs finished per unit of time. The harmonic mean of weighted IPCs is the reciprocal of the average normalized turnaround time (ANTT), i.e., the average slowdown of each program in the multiprogram workload versus isolated execution. We also make the case that IPC_i^{SP} should be the IPC over the same instructions as executed in the multiprogram experiment. For example, if a program only executes half of its instructions in the multiprogram experiment, the isolated IPC is the IPC of the first half of the instructions in isolated execution.

TABLE 1
Multiprogram speedup metrics (from [4]).

	single program	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
weighted speedup		1	1.25
H-mean of speedups		0.5	0.4
STP		1	1.25
ANTT		2	2.5
A-mean of IPCs		0.75	0.75
H-mean of IPCs		0.66	0.66

3 Weighted-IPC metrics are consistent

Michaud's first objection against weighted-IPC metrics is the lack of consistency [4]. He defines a consistent metric as a metric for which the result remains the same when the IPCs of the various benchmarks are permuted in the metric's formula. This is a justifiable definition and intuitively makes sense: all programs should be treated equally, and none should be (dis)favored. However, all of the metrics defined in the previous section comply with that definition, since summation and multiplication are commutative.

Table 1 shows Michaud's example to support his claim. Machine X runs both benchmarks at half their original speed, while machine Y runs one benchmark at its original speed, and one at a quarter of its speed. Michaud points to the fact that weighted speedup and the harmonic mean of speedups lead to different conclusions: weighted speedup suggests that machine Y performs better than machine X, while the harmonic mean indicates the opposite conclusion. The raw-IPC metrics, on the other hand, indicate that both machines perform equally well.

However, as discussed in [2], STP (weighted speedup) and ANTT (reciprocal of hmean of speedups) both measure a different aspect of multiprogram performance. Machine X executes each benchmark at half its original speed, so its total throughput is equal to a single-threaded machine that executes the benchmarks sequentially. Machine Y executes one program at its original speed, and one at a quarter, so its total throughput is 1.25 times higher than a single-threaded machine. This is reflected in the STP numbers. However, for machine X, each of the programs has a two-fold slowdown, while for machine Y, one has no slowdown and one executes four times slower. The average slowdown for X is 2, and 2.5 for Y, which are the ANTT numbers.

We can therefore conclude that machine Y has a higher throughput but also a larger average turnaround time than machine X. There is no inconsistency. There is also no objective reason to conclude that both machines perform equally well, as the raw-IPC metrics suggest. Machines X and Y behave differently, which is reflected in the STP and ANTT metric (Y has a higher throughput at the cost of a higher average turnaround time compared to X).

4 Weighted-IPC metrics treat programs equally fair

Michaud's second objection against the weighted speedup metric is that it favors unfairness. A configuration can show an increase in weighted speedup (or STP), while fairness is decreased. This is not in contradiction with the system-level meaning of weighted speedup as explained in [2]. Weighted speedup or STP quantifies the system-level throughput of the processor, which is the average number of jobs finished per unit of time. This is irrespective of how fair the throughput is divided among the programs. As shown in Table 1, throughput can be increased at the expense of fairness: machine Y favors benchmark A at the expense of benchmark B, so it is less fair, but it achieves a higher throughput.

This might seem surprising, because weighted speedup was introduced in [5] as a metric that is more fair compared to the sum of raw IPCs, which had been used prior to [5]. In fact, weighted speedup treats the programs in a more fair way than the sum of raw IPCs: a low-IPC program has the same weight as a high-IPC program, and no program is (dis)favored because of its inherent behavior. For example, assume a program with an isolated IPC of 0.2. A reduction of 0.1 in IPC will result in a slowdown of a factor of 2, as observed by the user. On the other hand, a program with an isolated IPC of 2 will experience a slowdown of only 5% if its IPC is reduced by 0.1. Raw-IPC metrics consider both slowdowns equally important, while weighted-IPC metrics capture the user and system perceived performance relative to isolated execution.

So, weighted speedup is more fair than raw-IPC metrics in the way the programs are treated, but it does not capture the variation in the amount of performance degradation between the programs. To quantify that kind of fairness, ANTT can be used or a specific fairness metric [2].

5 Weighted-IPC metrics are more intuitive

In the previous section, we already touched upon the fact that weighted-IPC metrics are more intuitive than raw-IPC metrics in the sense that they capture the user's and system's perception of performance. In a multiprogram context (multiple single-threaded programs), individual programs will almost always have the same or lower performance compared to isolated execution. There is usually no positive interference, where a program speeds up the execution of another program; there is only negative interference in the shared resources. On the other hand, there is an increase in system throughput, because multiple programs run concurrently. To evaluate multiprogram workload performance, it is therefore important to evaluate both components: individual program performance degradation and total system throughput increase, which is exactly what ANTT and STP measure.

Put differently, the programs in a multiprogram workload have an inherent workload behavior, which is determined by how the program is constructed (instruction mix, dependencies, memory access behavior, branch behavior, etc.). The effect of the inherent behavior on the IPC can be measured by an isolated run. A low-IPC program will remain low-IPC in a multiprogram context, while a high-IPC program is more likely to remain relatively high-IPC, if the system is somewhat fair. As described in the previous section, it is not fair to give equal weight to an absolute IPC reduction for low- and high-IPC programs. Calculating relative performance reductions is more intuitive and has a system-level meaning.

In order to further illustrate the intuition behind weighted-IPC metrics versus raw-IPC metrics, we elaborate some artificial examples. These contain numbers that are

TABLE 2
Example 1. The multithreaded processor consistently halves the performance for each of the programs.

		single program	multi program
exp. 1	benchmark A	IPC = 1	IPC = 0.5
	benchmark B	IPC = 2	IPC = 1
exp. 2	benchmark A′	IPC = 0.5	IPC = 0.25
	benchmark B	IPC = 2	IPC = 1

metric	exp. 1	exp. 2
STP	1	1
ANTT	2	2
A-mean IPCs	0.75	0.625
H-mean IPCs	0.67	0.4
G-mean IPCs	0.71	0.5

TABLE 3
Example 2. Both multi-threaded processors halve the performance of one program, the other one runs at full speed. Machine X favors the high-IPC program, while machine Y favors the low-IPC program.

		single program	multi program
machine X	benchmark A	IPC = 1	IPC = 0.5
	benchmark B	IPC = 2	IPC = 2
machine Y	benchmark A	IPC = 1	IPC = 1
	benchmark B	IPC = 2	IPC = 1

metric	machine X	machine Y
STP	1.5	1.5
ANTT	1.5	1.5
A-mean IPCs	1.25	1
H-mean IPCs	0.8	1
G-mean IPCs	1	1

made up, in order to illustrate certain cases. Section 6 contains results from real simulation experiments.

Assume we have a two-threaded processor that consistently halves the performance of each program (e.g., a poorly designed SMT processor). Table 2 shows two experiments on this processor. The difference between the experiments is one benchmark (A in the first experiment, A′ in the second; benchmark B is the same in both experiments).

STP and ANTT are consistent for both experiments: STP=1 means that the processor has the same throughput as a single-threaded processor, and ANTT=2 means that each program takes twice as long to execute compared to isolated execution. This is what can be intuitively expected for this processor. The raw-IPC metrics on the other hand are not consistent: the performance numbers are lower for the second experiment compared to the first. The raw IPC numbers depend on the inherent characteristics of the considered benchmarks (i.e., their single-program IPC). We can conclude that STP and ANTT quantify system performance, independent of the benchmark selection, while raw-IPC metrics are affected by the inherent characteristics of the individual benchmarks, which leads to less general conclusions.

In our next example (see Table 3), we compare two different machines X and Y using the same benchmarks. Both machines halve the performance of one program, while the other program runs at full speed. For machine X, the high-IPC program (benchmark B) has no performance degradation, while machine Y favors the low-IPC program (benchmark A).

The STP and ANTT metrics are again consistent: the throughput is 1.5 times higher than the single-threaded machine, and turnaround time is increased by a factor of 1.5. The raw-IPC metrics are inconsistent: the arithmetic mean favors the machine that is best for high-IPC programs, while the harmonic mean picks the machine that favors low-IPC programs. The geometric mean provides the correct conclusion that both processors perform equally well in this case, but it has no physical meaning.

6 The Choice of Multiprogram Metrics Matters for Real Experiments

The examples in the previous sections all consist of numbers that were made up in order to illustrate the (un)intuitiveness of the metrics under discussion. In most cases, these numbers are fairly extreme, to better clarify the case. One may ask the question whether the choice of metric makes a difference in real design studies and experiments. Real numbers tend to be less extreme, and in many cases the difference between two design alternatives is so clear that every metric that makes some sense provides the same qualitative conclusion (although the quantitative numbers may differ).

To evaluate the importance of metric choice, we set up the following study. We looked up the results from an experiment that compares multiple fetch policies for SMT processors executing multiprogram workloads (details are in [1]). We selected two fetch policy comparisons, one where the difference in performance is large (more than 10% difference in STP and ANTT; ICOUNT versus MLP-aware FLUSH in this case), and one where the difference in performance is less clear (less than 5% difference in STP and ANTT; MLP-aware FLUSH versus DCRA in this case). We picked the numbers for 30 four-program combinations running on a four-way SMT processor.

Tables 4 and 5 show the number of workloads (out of the 30 four-program combinations) for which different metrics result in contradictory conclusions. We say that two metrics result in an contradictory conclusion if one metric suggests an improvement in performance while the other one suggests a performance degradation when comparing the two fetch policies for the same workload. For example, in Table 4, the fourth number in the first row of numbers indicates that for 11 of the workloads, the conclusion (performance improvement or degradation) for H-mean of IPCs is opposite to that of STP. The average percentage of difference over all workloads is also reported for every metric.

For the pair of fetch policies where the difference in performance is clear (Table 4), STP and ANTT are in agreement for every workload (no contradictory conclusions). Although a disagreement in STP and ANTT is not a problem, as explained before, the second configuration has both higher throughput (better STP) *and* lower turnaround times (lower ANTT) in this experiment. The raw-IPC metrics agree relatively well with that conclusion, except for the harmonic mean of IPCs. Approximately one third of the workloads (9 to 11 out of 30) has a different conclusion

TABLE 4
Number of contradictory conclusions for every pair of metrics, and average percentage of difference, for two fetch policies that have a clear performance difference. The total number of workloads is 30.

	STP	ANTT	A-mean of IPCs	H-mean of IPCs	G-mean of IPCs
STP	0	0	1	11	0
ANTT	0	0	1	10	0
A-mean	1	1	0	11	1
H-mean	11	10	11	0	9
G-mean	1	0	1	9	0
avg	16%	-12%	22%	3%	13%

TABLE 5
Number of contradictory conclusions for every pair of metrics, and average percentage of difference, for two fetch policies that have a small performance difference. The total number of workloads is 30.

	STP	ANTT	A-mean of IPCs	H-mean of IPCs	G-mean of IPCs
STP	0	11	1	13	5
ANTT	11	0	9	7	2
A-mean	1	9	0	12	5
H-mean	13	7	12	0	4
G-mean	5	2	5	4	0
avg	0.9%	2.7%	0.8%	-5.5%	-1.7%

for this metric compared to the other metrics. In these cases, the harmonic mean of IPCs indicates a performance degradation, while the other metrics show an improvement. This is the case when one of the IPCs is low in the first configuration, and slightly lower in the second one, while the IPCs of the other jobs are higher and increase. The small decrease in a low number has a much higher penalty in the harmonic mean than the gains due to higher increases in high numbers. Since the IPC for this job is already low in isolated execution, STP and ANTT account for that. We can conclude that in case of a clear performance difference, all metrics provide the same conclusion, except for the harmonic mean of IPCs. However, the latter metric is the one that Michaud advocates in his paper.

Table 5 shows the results for the case where the performance difference is small between the two fetch policies. The STP and ANTT metrics are now contradictory for one third of the workloads. As discussed before, this is not a problem: the averages show that throughput is increased, but there is also an increase in turnaround time. Or in other words, the one fetch policy (DCRA in this case) yields higher system throughput compared to the other fetch policy (MLP-aware FLUSH in this case) at the cost of increased average job turnaround time. The arithmetic mean of IPCs seems to correlate better to STP, while the geometric mean has a better correlation with ANTT. However, none of these metrics provide the same conclusion as the weighted-IPC metrics for all of the workloads. The harmonic mean again shows the largest disagreement with the other metrics.

Note that this particular study does not show that one or another metric makes more sense. Due to the absence of a golden reference metric, we can only say that different metrics can lead to different conclusions. The previous sections were meant to show that STP and ANTT make more sense, are more intuitive and have a system-level meaning, in contrast to raw-IPC metrics. This section shows that using raw-IPC metrics can lead to different conclusions versus using weighted-IPC metrics, and therefore, raw-IPC metrics may result in incorrect conclusions.

7 Conclusion

In this paper, we restated the case for using weighted-IPC metrics for multiprogram workloads, and we showed that raw-IPC metrics, as proposed by Michaud [4], can lead to wrong conclusions. The main insight is that raw-IPC metrics are affected by the inherent behavior of programs in the workload, and that they do not capture user and system perceived performance. Weighted-IPC metrics on the other hand, weight multi-program performance of a program against its isolated performance, which leads to insightful metrics with a system-level meaning.

We argue that the weighted-IPC metrics do not suffer from inconsistency, in contrast to what Michaud claims. In fact, the examples in Section 5 show that the raw-IPC metrics are less consistent from the viewpoint of system-level performance. Fairness is also not a concern, because a pure throughput metric as STP is not supposed to account for fairness. Fairness can be calculated separataly, but is also incorporated in the ANTT metric.

We also show that the choice of the performance metric does affect conclusions in real experiments, especially when the performance difference between the design alternatives is small. Picking the wrong metrics can therefore lead to incorrect conclusions.

We conclude that for multiprogram workload evaluations, at least STP and ANTT should be reported, possibly complemented with a fairness metric. Raw-IPC metrics should be avoided at all times.

Acknowledgements

Stijn Eyerman is supported through a postdoctoral fellowship by the Research Foundation – Flanders (FWO). Additional support is provided by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC Grant agreement no. 259295.

References

[1] S. Eyerman and L. Eeckhout, "A memory-level parallelism aware fetch policy for SMT processors," in *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2007, pp. 240–249.
[2] ——, "System-level performance metrics for multi-program workloads," *IEEE Micro*, vol. 28, no. 3, pp. 42–53, May/June 2008.
[3] K. Luo, J. Gummaraju, and M. Franklin, "Balancing throughput and fairness in SMT processors," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Nov. 2001, pp. 164–171.
[4] P. Michaud, "Demystifying multicore throughput metrics," *Computer Architecture Letters (In Preprint)*, 2012.
[5] A. Snavely and D. M. Tullsen, "Symbiotic jobscheduling for simultaneous multithreading processor," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Nov. 2000, pp. 234–244.
[6] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multithreading: Maximizing on-chip parallelism," in *Proceedings of the 22nd Annual International Symposium on Computer Architecture (ISCA)*, Jun. 1995, pp. 392–403.