Extending Amdahl's Law for Multicores with Turbo Boost

Uri Verner, Avi Mendelson, and Assaf Schuster
Dept. of Computer Science, Technion, Israel
{uriv,avi.mendelson,assaf}@cs.technion.ac.il

Abstract—Rewriting sequential programs to make use of multiple cores requires considerable effort. For many years, Amdahl's law has served as a guideline to assess the performance benefits of parallel programs over sequential ones, but recent advances in multicore design introduced variability in the performance of the cores and motivated the reexamination of the underlying model. This paper extends Amdahl's law for multicore processors with built-in dynamic frequency scaling mechanisms such as Intel's Turbo Boost. Using a model that captures performance dependencies between cores, we present tighter upper bounds for the speedup and reduction in energy consumption of a parallel program over a sequential one on a given multicore processor and validate them on Haswell and Sandy Bridge Intel CPUs. Previous studies have shown that from a processor design perspective, Turbo Boost mitigates the speedup limitations obtained under Amdahl's law by providing higher performance for the same energy budget. However, our new model and evaluation show that from a software development perspective, Turbo Boost aggravates these limitations by making parallelization of sequential codes less profitable.

Index Terms—performance modeling, multicore, Turbo Boost, code parallelization, Amdahl's law

1 Introduction

Multithreading is known to provide speedup and reduced energy consumption. However, writing and validating parallel codes requires significant programming effort and expertise. Thus, given a sequential code, it is important to have a preliminary estimate of the expected benefits of its parallelization. Such estimates are commonly computed using Amdahl's law [2] and its extension for energy [12], which provide easy-to-compute formulas for upper bounds for the corresponding parameters. These formulas are based on several simplifying assumptions, such as the processors being identical and their speeds fixed, and they do not account for external factors that may affect the performance of the cores.

Most modern processors are limited by maximum power and thermal constraints. Therefore, in multicore processors the nominal frequency is set according to the power consumption and temperature when all the cores are active. However, when some of the cores are idle, the spare power budget can be utilized to increase the frequency of the active cores; most modern multicore processors implement such a mechanism. For example, Intel CPUs with Turbo Boost technology [10] are equipped with a frequency governor that aims to maximize core frequencies such that the power consumption, current consumption, and temperature stay below certain limits; AMD processors use a similar technology called Turbo Core. Hereafter, we refer to these mechanisms as Turbo Boost. These technological advances encouraged us to revisit Amdahl's law for multicore processors.

Using analysis and experimental evaluation, we show that, from a software design perspective, Turbo Boost aggravates the speedup limitations obtained by Amdahl's law by making parallelization of sequential codes less profitable. This is in contrast to previous studies showing that from a processor design perspective, frequency scaling mechanisms mitigate these limitations by providing higher performance for the same energy budget [4], [5]. We extend the model used in Amdahl's law to account for frequency changes. Using this extended model, we propose refined formulas for the maximum expected speedup and energy scaling, and validate them experimentally.

2 Background

2.1 Amdahl's Law

Amdahl's law states that having multiple processors to accelerate a program has limited effect on the speedup unless the processing rate of its sequential component is increased by nearly the same magnitude. Gene Amdahl illustrated this "law" using the following simplified model. The program has two parts: a parallel part of proportionate size f, which is equally split without scheduling overhead between N identical processors running in parallel, and a sequential part of size $1-f$ that does not overlap the parallel part. The speedup in this case is computed using the following formula:

$$speedup_{Amdahl} = \frac{1}{(1-f)+\frac{f}{N}} \ . \qquad (1)$$

Using this model, Amdahl advocated for the continuation of using single-processor machines (in 1967) [2], [3]; the formula was, in fact, derived from his original paper by others.

Four decades later, when the industry turned to multicore processors, a series of works extended Amdahl's law to take into account additional characteristics of these processors (see the survey in [1]). The assumptions in Amdahl's law were refined for multicores in two main directions:

- **Architecture**: use of various chip designs, including symmetric, asymmetric, dynamic, distributed, and heterogeneous multicores, and dynamic frequency scaling.
- **Shared resources and scalability**: cache contention, limited memory latency/bandwidth, communication between cores, synchronization, context switching, and computation scaling.

The extended models provided more accurate estimation of performance and energy consumption, and showed architectural tradeoffs in multicore design. Hill and Marty [9] compared different multicore designs under a fixed resource budget (e.g., chip area) and provided important insights for chip architects. Their designs consist of a number of basic cores and at most one more powerful core. These models generally cannot represent dynamic voltage and frequency scaling (DVFS), where the power of more than one core can be increased or decreased dynamically. Optimizing time and energy in multicores using DVFS was examined in [6], [7], [11], [4], [8]. Turbo Boost implements many such optimizations in an embedded automatic power management mechanism.

This paper considers a different aspect of Amdahl's law – estimating the improvement ratio in time and energy between parallel and serial codes on an *existing* multicore processor with Turbo Boost. Unlike previous works, in our model the sequential program runs on one core of the multicore processor.

2.2 Energy Scaling

The energy consumed by a program is equal to $P \times T$, where P is the average power consumption and T is the execution time. Woo and Lee [12] extended Amdahl's law for energy efficiency. In their augmented model, each processor consumes one unit of power when it is in active state and a fraction π when it is in idle state ($0 \leq \pi \leq 1$). The energy consumption of the parallel program is given by the following formula:

$$\begin{aligned} E &= \underbrace{(1-f)\cdot(1+(N-1)\pi)}_{serial} + \underbrace{\frac{f}{N}\cdot N}_{parallel} \\ &= 1+(N-1)\pi(1-f)\ . \end{aligned} \quad (2)$$

The energy consumption of the sequential program, which is also executed on the multicore processor, is equivalent to the energy consumption of a parallel program with $f = 0$. Therefore, the energy scaling factor is as follows:

$$E^{+} = \frac{1+(N-1)\pi}{1+(N-1)\pi(1-f)}\ . \quad (3)$$

Normally, program parallelization reduces energy consumption, as the numerator is greater than the denominator.

3 Extended Amdahl's Law

3.1 System Model

A compute-intensive sequential program is executed on a symmetric multiprocessor with N cores and Turbo Boost. An alternative implementation is considered, where a part of proportionate size f is split equally between the N cores without overhead; we call this implementation the parallel program. Our aim is to compute the maximum expected speedup and energy improvement of the parallel program over the sequential one. Our model extends Amdahl's law to account for multiprocessors where the performance of each processor depends on the number of active (non-idle) processors. Concretely, we focus on multicore CPUs with Turbo Boost, which operate at higher frequency when fewer cores are active. For example, Tables 1 and 2 show that the maximum CPU frequency scales inversely with the number of active cores in two Xeon series CPUs. We denote the effective speed (performance) of every active core when n cores are active by $s(n)$. Note that $s(N)$ is higher than the base frequency. This is partly because the chip's temperature depends on other factors besides the number of active cores.

3.2 Extended Speedup Model

In the extended model, the core speeds are not fixed. For simplicity, we normalize all the speeds by $s(1)$. Thus, when only one core is active, its normalized speed is 1, and when N cores are active, the normalized speed of each core is $s(N)/s(1)$. We modify the speedup formula to match the extended model by introducing a correction factor for the parallel part:

$$speedup_{multicore} = \frac{1}{(1-f)+\frac{f}{N}\cdot\frac{s(1)}{s(N)}}\ . \quad (4)$$

TABLE 1: Frequency specification: 8-core Intel Xeon E5-2690

Active cores	1	2	3	4	5	6	7	8	Base
Frequency [GHz]	3.8	3.6	3.6	3.4	3.4	3.3	3.3	3.3	2.9

TABLE 2: Frequency specification: 12-core Intel Xeon E5-2658 v3

Active cores	1	2	3	4	5 – 12	Base
Frequency [GHz]	2.9	2.9	2.7	2.6	2.5	2.2

Normally, the factor $s(1)/s(N)$ is greater than 1 because higher frequency is achieved when some cores are idle. Therefore, the above formula provides a tighter upper bound on the speedup than does Amdahl's law.

3.3 Extended Energy Scaling Model

The energy model of Woo and Lee [12] assumes that active cores always have the same power consumption. However, in multicores with frequency scaling mechanisms, active cores consume more power when some cores are idle, because they operate at higher frequency. We augment the model we used for the speedup formula with information about power to compute the energy improvement factor.

Let $P(n)$ denote the average power consumption of the CPU when n cores are active ($1 \leq n \leq N$). Then, the energy consumption during a period of length t where n cores are active is $P(n)\cdot t$. The energy scaling factor is hence described by the following formula:

$$\frac{1\cdot P(1)}{(1-f)\cdot P(1)+\frac{f}{N}\cdot\frac{s(1)}{s(N)}\cdot P(N)}\ .$$

We rewrite this formula in a slightly more compact form:

$$E^{+}_{refined} = \frac{1}{(1-f)+\frac{f}{N}\cdot\left(\frac{P(N)}{s(N)}\Big/\frac{P(1)}{s(1)}\right)}\ . \quad (5)$$

4 Evaluation

In this section, we evaluate the accuracy of the traditional and refined formulas for speedup and energy scaling. We start with an experimental evaluation on two Intel multicore CPUs with Turbo Boost. Then, we perform a theoretical analysis to identify how different problem parameters will affect accuracy, assuming our model is correct, and examine the potential for improvement in a series of Intel Xeon CPU models, based on their specification.

4.1 Experimental Setup

We used two evaluation platforms that were located in an air-conditioned server room.

- A Sandy Bridge platform composed of two eight-core Intel Xeon E5-2690 CPUs with 128GB RAM that runs Linux kernel 3.13.0.
- A Haswell platform composed of two twelve-core Intel Xeon E5-2658 v3 CPUs with 64GB RAM that runs Linux kernel 4.0.9.

We disabled hyperthreading, CPU power limitations, and the CPU in Socket 1, and configured the OS frequency scaling governor to run the CPUs at maximum frequency. The base frequency and the maximum Turbo frequencies for the two processor types are shown in Tables 1 and 2.

A worker thread was attached to each core and executed a given function in a loop, for a given number of iterations. Each thread measured its execution time, and thread 0 also measured the energy consumed by the cores and by the entire processor

TABLE 3: Execution time [sec]

	Xeon E5-2690				Xeon E5-2658 v3	
	INT		AES		AES	
f	Turbo	No Turbo	Turbo	No Turbo	Turbo	No Turbo
0	15.8	20.7	16.0	20.7	20.8	27.7
0.2	13.1	17.1	13.2	17.1	17.0	22.3
0.4	10.4	13.4	10.4	13.5	13.3	17.3
0.6	7.7	9.8	7.7	9.9	9.5	12.3
0.8	5.0	6.2	5.1	6.2	5.8	7.3
1	2.3	2.6	2.3	2.7	2.0	2.3

TABLE 4: Package energy consumption $[J]$

	Xeon E5-2690				Xeon E5-2658 v3	
	INT		AES		AES	
f	Turbo	No Turbo	Turbo	No Turbo	Turbo	No Turbo
0	712.8	639.6	757.1	662.0	876.4	1097.3
0.2	608.7	548.0	646.8	581.5	728.5	918.6
0.4	509.4	457.5	544.9	479.7	597.7	734.9
0.6	407.3	366.3	440.8	388.3	449.7	548.6
0.8	307.9	274.6	340.4	295.2	310.6	357.2
1	209.4	182.8	229.6	202.6	166.4	168.8

package, using processor counters `MSR_PKG_ENERGY_STATUS` and `MSR_PP0_ENERGY_STATUS`. The process was run with maximum priority.

We measured the accuracy of speedup and energy improvement formulas in the following three configurations:

INT-SB:	a synthetic sequence of integer MUL and DIV operations on the Sandy Bridge platform.
AES-SB:	an AES-CBC 128-bit encryption function from the Crypto++ 5.6.2 library on the Sandy Bridge platform.
AES-HW:	the AES-CBC function on the Haswell platform.

For each configuration, a program with a configurable parallel portion $f \in [0, 1]$ and a sequential portion $(1 - f)$ was executed, and its execution time and energy consumption were measured. The experiments were run with Turbo Boost enabled and again with it disabled.

4.2 Speedup Measurements

The time measurement results are listed in Table 3. For each value of f, we computed the speedup by dividing the execution time of the sequential program ($f = 0$) by that of the parallel program, and used it to compute the error of each formula.

Figure 1 shows the estimation errors of the traditional and refined speedup formulas for the AES-HW configuration, with Turbo Boost enabled. The traditional formula overestimated the speedup by 16% for $f = 1$, while the refined formula accurately estimated the speedup with a maximum error of 0.3% and as low as 0.04% for $f = 1$. These results clearly show that the refined formula significantly improves the accuracy of the speedup estimation, indicating that the upper-bound on speedup given by the refined formula is achievable for real-world applications. For INT-SB, the error similarly increased, up to 15.1% for $f = 1$, while the refined formula was very accurate with 0.05% maximum error. The function here was designed for maximum utilization of the ALU; hence these results verify that the core frequencies are as expected. For AES-SB, the results were similar; the maximum error of the original formula was 16.2%, while for the refined formula it was 0.9%.

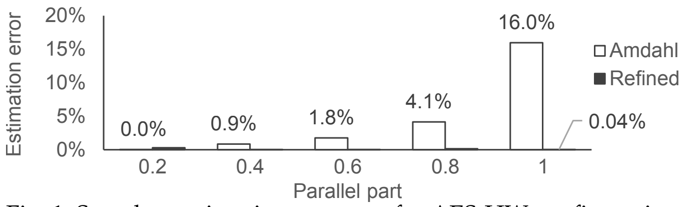

Fig. 1: Speedup estimation accuracy for AES-HW configuration

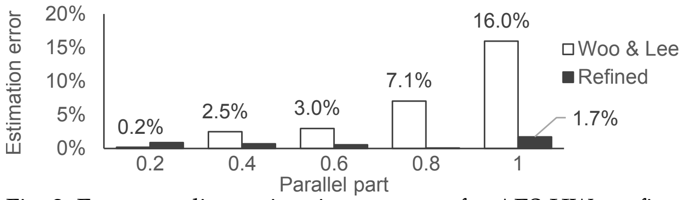

Fig. 2: Energy scaling estimation accuracy for AES-HW config.

4.3 Energy Consumption Measurements

Figure 2 shows the estimation errors of the traditional and refined energy scaling factor formulas for the AES-HW configuration, with Turbo Boost enabled. The refined formula significantly improved the accuracy, though its error was not as low as for the speedup. Notably, the maximum error of the original speedup and energy improvement formulas is the same at 16%. We show that this result is expected in our theoretical analysis in Section 4.5. As for speedup, the results indicate that the estimated improvement in energy consumption is achievable in real-world applications. For INT-SB, the trend was similar. The original formula had errors 5.0% and 15.1% for $f = 0.8$ and $f = 1$ in the original formula, while the refined formula had a maximum error of 0.8%, for $f = 0.6$, and an error of 0.03% for $f = 1$. For AES-SB, the errors were higher for both formulas; for the original formula they were 10% and 16.2%, for $f = 0.8$ and $f = 1$, while for the refined formula they were 2.1% and 0.9%, respectively.

The refined formula uses the average power consumption parameters $P(1)$ and $P(N)$. We measured $P(n)$ for every $1 \leq n \leq N$ active cores as follows: a fixed-size workload was split equally between the n cores, and the power was computed by dividing the CPU package energy consumption by the execution time. On the Haswell platform with Turbo Boost enabled, for the AES-CBC workload, the power scaling was close to linear, $P^T_{pkg}(n) \approx 3.7n + 37.9$ [W], but the curve was not smooth. With Turbo Boost disabled, it was also close to linear and the curve was smoother; $P^D_{pkg}(n) \approx 2.8n + 28.4$ [W].

4.4 Accuracy With Turbo Boost Disabled

Our refined formulas are based on the assumption that Amdahl's law accurately models multicores with fixed frequency; hence, we expect the original formulas to be accurate with Turbo Boost disabled. The results of our experiments confirm this assumption. For INT-SB, the maximum error of the original speedup function was only 0.1%. For AES-HW, the error was 1.2% for all $f = 0.2 - 1.0$. Interestingly, for AES-SB, the error for $f = 0.2 - 0.8$ was at most 0.4%, but for $f = 1$ it was 4.4%. Our temperature measurements indicate that this was not caused by thermal issues. Note that in all the experiments, the parallel part is executed by *all* the threads, so any delay caused by using all the cores (such as frequency throttling) should have an effect for any $f > 0$. We ascribe this error to microarchitectural issues.

For energy scaling, the formula based on Woo and Lee's model was also quite accurate. For INT-SB, the maximum error was 0.3%; for AES-HW it was 1.3%; and for AES-SB it was again 4.4%. Note that Woo and Lee's formula yields different estimates when Turbo Boost is enabled and when it is disabled because the idle state power parameter π is different in the two cases, which is why there is a difference in the estimation error.

4.5 Theoretical Analysis

Assuming that our extended model is correct, the error of the original formula is $abs(\frac{o-r}{r})$ where o and r are the results of the original and refined formulas, respectively. After rearranging the terms, the expected speedup error is

$$err_{speedup} = \frac{1}{N \cdot \left(\frac{1}{f} - 1\right) + 1} \cdot \left(\frac{s(1)}{s(N)} - 1\right) . \quad (6)$$

The formula shows that the factors that can increase the error include increasing the parallel part f, changing to a CPU with a larger range of Turbo frequencies, or changing to a CPU with fewer cores. We used this formula to find that the average improvement potential for 36 Haswell-EP CPU models with 4-18 cores is 15.5%, and the maximum is 38%.

To compute the energy improvement error, we first need to bring the two formulas to use the same representation of power. π is defined as the fraction of the power of an active core consumed by an idle core; consequently, it can be represented as follows: $\pi = \frac{N}{N-1} \cdot \frac{P(1)}{P(N)} - \frac{1}{N-1}$. The expected error in energy improvement is

$$err_{energy} = \frac{\left(\frac{s(1)}{s(N)} - 1\right) f}{N \cdot \frac{P(1)}{P(N)} \cdot (1-f) + f} . \quad (7)$$

The error can increase by the factors mentioned for speedup, and additionally by changing to a CPU where idle cores consume less power.

For $f = 1$ we get $err_{speedup} = err_{energy} = \frac{s(1)}{s(N)} - 1$. This explains why we got the same maximum error for speedup and energy in our experiments; e.g., for AES-HW, this error is $\frac{2.9}{2.5} - 1 = 16\%$, as we observed.

5 Conclusions and Discussion

Turbo Boost and similar frequency scaling technologies such as Turbo Core have changed the scaling curve of parallel programs. One might expect that without any parallelization overhead, the speedup would be linear, but with Turbo Boost the performance scales sub-linearly.

While the use of multiple core processors has become ubiquitous in modern computer systems, much of the legacy software remains sequential due to the significant cost and effort required for its parallelization. Therefore, simple tools for initial evaluation of the speedup and energy savings that could be gained from the parallelization are needed.

We have shown that the speedup limitations obtained under Amdahl's law are worse on multicore processors with Turbo Boost, where the use of multiple cores is accompanied by lower frequency. Previous works [4], [5] conversely claimed that Turbo Boost mitigates these limitations. The difference in the conclusions results from a different choice of reference points. The prior works showed that multicore processors with scaling mechanisms can complete the program in shorter time than their fixed-performance counterparts that have the same power consumption. This work, on the other hand, evaluated the speedup factor achieved from parallelizing a sequential program such that it makes use of multiple CPU cores.

We presented a refined formula for Amdahl's law that computes a tight upper bound on the speedup for CPUs with Turbo Boost while having a minimal impact on the simplicity of the model. In our experiments with AES-CBC 128-bit encryption and a synthetic compute-intensive function on an eight- and twelve-core Intel Xeon series CPUs with Turbo boost, the estimated speedup using Amdahl's law was up to 16% too high. In contrast, our refined formula had an estimation error of less than 1%. The changes in core frequency with Turbo Boost affect the power consumption of the CPU, so the extension of Amdahl's law for energy scaling also needs to be revised. We presented a refined formula for the energy improvement factor that can be achieved from parallelization on a CPU with Turbo Boost and showed that it significantly reduces the estimation error. Using theoretical analysis, we showed that increasing the parallel part, changing to a CPU with a larger range of Turbo frequencies, or changing to a CPU with fewer cores can increase the necessary correction factor. For 36 Haswell-EP CPU models with 4-18 cores, we found the average correction factor to be 15.5%, and the maximum 38%. For energy, changing to a CPU with more power-efficient idle cores can also increase the correction factor. The analysis also shows that with a fully parallelizable program, the correction factor for speedup and energy is the same.

Normally, the frequency of a core is highest when all other cores are idle; however, this is not guaranteed by the CPU specification. Theoretically, it is possible that for some CPU and workload, the sequential program would run only at base frequency (e.g., 2.2GHz), while the parallel program would run at a Turbo frequency (e.g., 2.5GHz). In this case, the speedup would be higher than the upper bound given by Amdahl's law. For example, suppose the sequential program starts running at maximum Turbo frequency and shortly after that, the core overheats and its frequency drops to base; in the parallel program, however, each core runs at a lower Turbo frequency and does not overheat.

References

[1] B. M. Al-Babtain, F. J. Al-kanderi, M. F. Al-fahad, and I. Ahmad. A Survey on Amdahl's Law Extension in Multicore Architectures. *International Journal of New Computer Architectures and their Applications (IJNCAA)*, 3(3):30–46, 2013.

[2] G. M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings*, 30:483–485, 1967.

[3] G. M. Amdahl. Computer architecture and Amdahl's law. *Computer*, 46(12):38–46, 2013.

[4] M. Annavaram, E. Grochowski, and J. Shen. Mitigating Amdahl's law through EPI throttling. *32nd International Symposium on Computer Architecture (ISCA'05)*, 2005.

[5] J. Charles, P. Jassi, N. S. Ananth, A. Sadat, and A. Fedorova. Evaluation of the Intel Core i7 Turbo Boost feature. *IEEE International Symposium on Workload Characterization (IISWC)*, pages 188–197, Oct. 2009.

[6] S. Cho and R. G. Melhem. Corollaries to Amdahl's law for energy. *IEEE Computer Architecture Letters*, 7(1):25–28, 2008.

[7] S. Cho and R. G. Melhem. On the Interplay of Parallelization, Program Performance, and Energy Consumption. *IEEE Transactions on Parallel and Distributed Systems*, 21(3):342–353, 2010.

[8] U. Gupta and U. Ogras. Constrained energy optimization in heterogeneous platforms using generalized scaling models. *Computer Architecture Letters*, 14(1):21–25, Jan 2015.

[9] M. D. Hill and M. R. Marty. Amdahl's Law in the Multicore Era. *Computer*, 41(July):33–38, 2008.

[10] Intel Corporation. Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors. November 2008.

[11] S. Londono and J. de Gyvez. Extending Amdahl's law for energy-efficiency. In *International Conference on Energy Aware Computing (ICEAC)*, pages 1–4, Dec 2010.

[12] D. H. Woo and H.-H. S. Lee. Extending Amdahl's law for energy-efficient computing in the many-core era. *Computer*, 41:24–31, 2008.