Cache-aware Roofline model: Upgrading the loft

Aleksandar Ilic, Frederico Pratas, and Leonel Sousa
INESC-ID/IST, Technical University of Lisbon, Portugal
{ilic,fcpp,las}@inesc-id.pt

Abstract—The Roofline model graphically represents the attainable upper bound performance of a computer architecture. This paper analyzes the original Roofline model and proposes a novel approach to provide a more insightful performance modeling of modern architectures by introducing cache-awareness, thus significantly improving the guidelines for application optimization. The proposed model was experimentally verified for different architectures by taking advantage of built-in hardware counters with a curve fitness above 90%.

Index Terms—Multicore computer architectures, Performance modeling, Application optimization

1 Introduction

Driven by the increasing complexity of modern applications, microprocessors provide a huge diversity of computational characteristics and capabilities. While this diversity is important to fulfill the existing computational needs, it imposes challenges to fully exploit architectures' potential. A model that provides insights into the system performance capabilities is a valuable tool to assist in the development and optimization of applications and future architectures.

In order to model the interrelation between architecture capabilities and application characteristics, it is usually required to develop architecture-specific testing and simulation environments, which result in accurate but complex, difficult to develop and hard to use models. However, simpler "bound and bottleneck" approaches can provide valuable insights into the primary factors that affect the system performance, and give useful guidelines for improving applications and architectures [3]. In particular, the Roofline model [10] provides insights into inherent architectural bottlenecks and potential application optimizations. Its usefulness is patent in several works [9], both at the application [4], [6], [8] and at the architectural level [5], [7].

Modern multi-core systems can be represented as a set of central processing units (Cores) with on-chip memory hierarchy connected to a DRAM memory (see Fig. 1). Hence, the original Roofline model [10] shows the maximum attainable performance of a computer architecture as a line in the form of a roof. It relates the peak floating-point (FP) performance (F_p in $flops/s$), the peak DRAM memory bandwidth (B_D in β_D/s), and the application's *operational intensity* (I in $flops/\beta_D$), where β_D represents the data traffic in DRAM bytes-accessed. It is based on the assumption that memory transfers and computation can overlap, so the application execution time, $T(I)$, is limited by F_p or B_D:

$$T(I) = T(\frac{\phi}{\beta_D}) = \phi \times \max\left\{\frac{1}{B_D \times I}, \frac{1}{F_p}\right\} \quad (1)$$

where ϕ is the number of performed FP operations ($flops$). Thus, the original Roofline model defines the maximum attainable performance ($F_a(I)$), as:

$$F_a(I) = \frac{\phi}{T(I)} = \frac{1}{\max\left\{\frac{1}{B_D \times I}, \frac{1}{F_p}\right\}} = \min\{B_D \times I, F_p\} \quad (2)$$

Figure 2 shows the original Roofline model for the quad-core Intel 3770K processor with the characteristics presented in Tab. 1. The horizontal part of the Roofline model corresponds to the compute bound region where the F_p can be achieved. The slanted part of the model represents the memory bound region where the performance is limited by B_D. The ridge point, where the two lines meet, marks the minimum I required to achieve F_p. In general, the original Roofline modeling concept [10] ties the F_p and the theoretical bandwidth of a single memory level, with the I expressed in ϕ per accessed byte at that memory level. For example, if working sets fit into the L2 cache, this concept considers the peak L2 bandwidth (B_{L2}) and I is defined as ϕ per L2 bytes accessed (*i.e.*, between L1 and L2 caches). Hence, when improving the memory access pattern of applications, it is required to construct and simultaneously use several instances of the model (one for each memory level) in order to assess the optimization gains.

As shown in this paper, the original Roofline modeling concept is usually not sufficient to fully describe the performance of modern applications and architectures which rely on the on-chip memory hierarchy. This paper proposes the *Cache-aware Roofline model*, that is based on a different Core-centric concept where FP operations, data traffic and memory bandwidth at different levels are perceived from a consistent architectural point of view, *i.e.*, as seen by the Core. These changes introduce a fundamentally different view on the attainable performance and behavior of modern applications and architectures, thus providing more insightful guidelines for application optimization. Moreover, the proposed model is a single-plot model that translates into an increase of the maximum attainable performance when directly compared to the original DRAM roof, which can be considered as an upgrade to the loft.

2 Cache-aware Roofline Model

As stated before, in contrast to the original modeling concept that ties the F_p with the bandwidth and data traffic at a single memory hierarchy level, the proposed Cache-aware Roofline modeling concept differs in how memory traffic and bandwidth are observed. There are two fundamental key differences: *i*) we account for all memory operations including accesses to the different cache levels; and *ii*) the bandwidth depends on the accessed memory levels and it is defined relatively to the Core. Since the proposed model

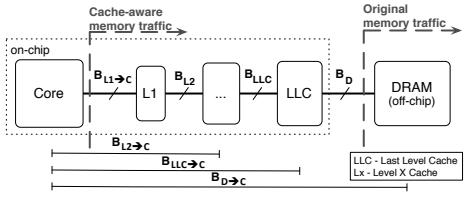


Fig. 1: General computer architecture

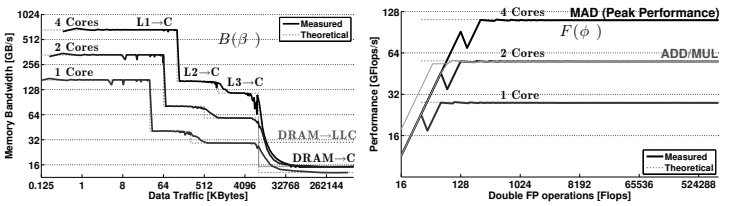


Fig. 3: Performance and bandwidth variation on the Intel 3770K

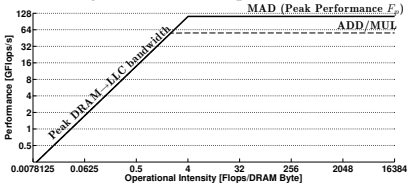


Fig. 2: Original Roofline model (3770K)

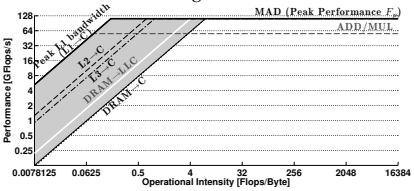


Fig. 4: Cache-aware Roofline model (3770K)

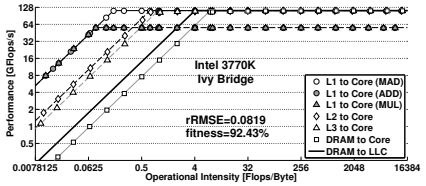
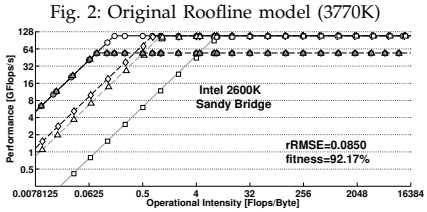
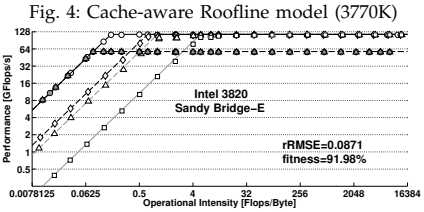


Fig. 5: Model validation for Intel 3770K





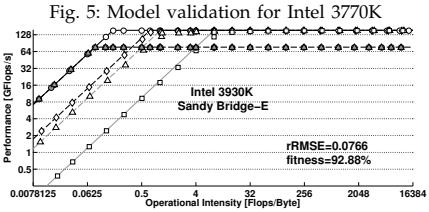


Fig. 6: Cache-aware Roofline model validation for different general-purpose machines

considers the complete volume of memory traffic, *i.e.*, the total number of transferred $bytes$ (β), the I in the Cache-aware Roofline Model is uniquely defined for all levels of the memory hierarchy, thus resulting in a single-plot model. In fact, in the proposed model, I can be the arithmetic intensity (ϕ/β). However, we use the term operational intensity to reflect the possibility of applying the proposed model for other types of operations (not necessarily arithmetic).

In order to experimentally assess the bandwidth and performance, we executed a series of tests based on Alg. 1. In particular, the bandwidth was characterized by varying the number of memory operations (test code A) to hit different memory levels by accessing contiguous and increasing memory addresses. Figure 3 shows the results for the quad-core Intel 3770K architecture[1] (see Tab. 1), where bandwidth varies with the number of transferred bytes. It also shows that the achievable peak FP performance, obtained by varying the number of $flops$ (test code B), depends on the occupation of the arithmetic units' pipeline. Figure 3 also depicts the results for a different number of cores. Thus, the accurate Roofline model must consider these variations such that:

$$F_a(I) = \frac{\phi}{T(I)} = \min\{B(\beta) \times I, F(\phi)\} \quad (3)$$

where B and F are continuous functions of β and ϕ.

In order to derive the limits of the attainable FP performance in the Cache-aware Roofline model, one must consider the peak FP performance ($F(\phi)$=F_p) for the compute bound region and the L1 peak bandwidth ($B_{L1\rightarrow C}$) for the memory bound region (the cache level closest to the Cores). Figure 4 represents both the original and the Cache-aware Roofline models as the upper bounds for the achievable FP performance. It can be observed that the memory bound region of the original model is extended in the Cache-aware Roofline model, an effect herein designated as "upgrading the loft". The new memory bound limits can be derived from Equation (3), substituting $B(\beta)$ by $B_{L1\rightarrow C}$. The latter value can be obtained from the processor specifications by multiplying the number of cores, the L1 bus width, and the clock frequency (see Tab. 1). Moreover, the insightfulness of the Cache-aware Roofline model can be further extended by introducing additional memory ceilings depending on the achievable bandwidth for each memory level (see Fig. 3 and 4). It is worth mentioning that the bandwidth from the DRAM to the Core ($B_{D\rightarrow C}$) is lower than the bandwidth considered in the original Roofline model (B_D) due to the fact that the data goes through all cache levels before arriving to the processor Cores (see Fig. 3 and 4). Overall, the proposed model reveals a previously unexplored area, thus allowing even better insights on the attainable performance.

Experimental Setup and Model Validation

In order to validate the proposed model, we performed an extensive set of experiments with micro-benchmarks based on Alg. 1 (test code C), across 3 different architectures of Intel processors in 4 computer systems (see Tab. 1). The micro-benchmarks are assembly programs designed to hit a given level of the memory hierarchy and to reach a specific operational intensity by interleaving different numbers of FP and memory operations. The proposed model does not make any direct assumptions associated to Intel processors and can be applied to any general-purpose architecture.

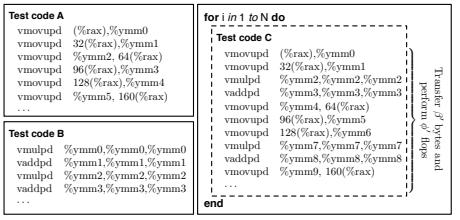
To experimentally construct the models, we have created a specialized software tool[2] that relies on built-in hardware performance counters [1]. Each obtained point represents the median of 8192 micro-benchmark repetitions to reduce the intrinsic measurement errors in real non-dedicated systems. The accuracy of the testing tool and procedures can be evidenced in Fig. 3 where the theoretical peak L1 bandwidth was achieved, although these tests are the most error prone and sensitive to the measurement method due to the very low number of memory operations.

The theoretical models, presented as lines in Fig. 5 and 6, were validated by the experimentally obtained re-

1. To fully exploit the architecture, we consider herein double-precision FP Advanced Vector Extensions (AVX) instructions.

2. The specialized software tool is available upon request by e-mail to the corresponding authors.

Algorithm 1: Assembly test micro-benchmark codes



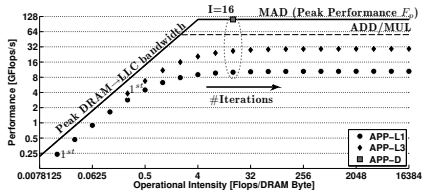


Fig. 7: Application classes in the Original Roofline model

sults (points), which resemble quite accurately the expected shapes for each tested system and memory hierarchy level. The resulting performance and operational intensity (as shown in Fig. 5 and 6) were obtained with the hardware performance counters, *i.e.*, as seen by the Core. In detail, the experimental points achieve an rRMSE of about 0.1 and a fitness higher than 90% in all cases[3]. As expected, each architecture has different rooflines and ceilings: *e.g.*, the six-core 3930K delivers the highest F_p, whereas the 2600K results in the lowest $B_{D\rightarrow C}$.

3 Analyzing the Roofline models

In order to show the practical repercussions of redefining the original Roofline model[4] for cache-awareness, we selected 3 micro-benchmarks from the ones used for model validation in the previous section (test code C in Alg. 1). The selected micro-benchmarks represent the kernels of three classes of applications (see Tab. 2), which differ in the amount of transferred $bytes$, β', and in the number of performed $flops$ per iteration, ϕ' ("for" loop in Alg. 1). Each class consists of k instances with different number of iterations $N{=}2^k$ ($1{\leq}k{\leq}20$), where in each iteration the FP operations are performed on the same data set. For each APP-D instance, all memory operations in each iteration involve accesses to the DRAM. For the remaining two classes they occur only during the first iteration, while for subsequent iterations different on-chip memory levels are accessed, L1 cache for APP-L1 and L3 cache for APP-L3. Figures 7 and 8 show the results obtained for all the classes and instances (points in the figures).

Observation 1: *the same application can be perceived differently in the two Roofline models in terms of its operational intensity when different problem sizes are considered.*

As reported in [10], in the original Roofline model, for the classes APP-L1 and APP-L3, both performance and I vary with N for each instance, as depicted in Fig. 7 (shown by the arrow). In contrast, in the Cache-aware Roofline model the instances of the three classes do not change their I with the problem size (see Fig. 8). This phenomenon arises

3. rRMSE: relative root-mean squared error; fitness=$\frac{100}{(1+rRMSE)}$

4. Due to the fact that the authors in [10] mainly focus on the DRAM variant of the original model, the results presented herein also consider this variant. However, the observations described herein are also valid for its cache level variants.

TABLE 1: General-purpose CPUs used for model evaluation

Intel processor model		2600K	3820	3930K	3770K
Architecture		Sandy Bridge	Sandy Bridge-E	Sandy Bridge-E	Ivy Bridge
#Cores		4	4	6	4
frequency [GHz]		3.4	3.6	3.2	3.5
F_p, AVX MAD (8 flops) [GFlops/s] (*)		108.8	115.2	153.6	112
L1 (per core)	size [bytes]	32K	32K	32K	32K
	bus width [bits]	384	384	384	384
	bandwidth [GB/s]	163.2	172.8	153.6	168
L2 size [bytes]		256K	256K	256K	256K
L3 size [bytes]		8M	10M	12M	8M
DRAM to LLC (DDR3)	#channels (8bytes/channel)	2	4	4	2
	bus frequency [MHz]	$2{\times}800$	$2{\times}933$	$2{\times}800$	$2{\times}933$
	bandwidth B_D [GB/s] (**)	25.6	59.7	51.2	29.9

(*) F_p=Max #flops×#Cores×frequency; (**) B_D=#channels×word size×bus frequency;

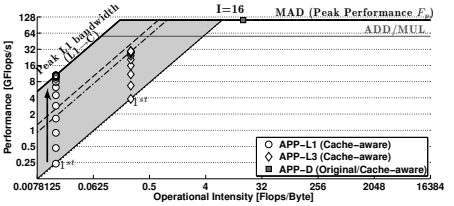


Fig. 8: Application classes in the Cache-aware Roofline model

TABLE 2: Different application classes (values per iteration)

Application class	β' [bytes]	ϕ' [flops]	I (1^{st} iteration)
APP-L1	4.5K	72	$1/64$
APP-L3	2M	524.3K	$1/4$
APP-D	61M	1023.4 M	16

from the fact that both models account for the total number of performed FP operations at the core level ($\phi'{\cdot}N$), while they perceive memory traffic differently. The original model accounts for the memory traffic to a specific memory level (*e.g.*, the DRAM), whereas the proposed model accounts for the total memory traffic as seen by the core ($\beta'{\cdot}N$).

For APP-D class, both models see the complete volume of data traffic, since all instances access the DRAM, thus they achieve the peak performance without changing their I (see the overlapping square points). However, for APP-L1 and APP-L3, only the instances with a single iteration (points marked as "1^{st}") show the same behavior in both models. For the other instances, the original and the proposed models see a different volume of data traffic, since subsequent iterations access a specific cache level (L1 for APP-L1 and L3 for APP-L3). In the original model, for these two classes, I shifts with N for each instance due to the fact that the total number of DRAM accesses becomes constant after the first iteration (β'), thus the same application can shift from a memory bound region to a compute bound region just by increasing its problem size. In contrast, in the proposed model the performance obtained for these instances shows a consistent trend towards the memory ceiling corresponding to the accessed cache level (see arrow in Fig. 8).

Since both models see the performance in the same way, the instances achieve the same performance in both, but with shifted I in the original model. Indeed, if each performance point in the original model is projected on a vertical line of fixed operational intensity (the I of the points marked as "1^{st}"), it will correspond to a point of equivalent performance in the Cache-aware model.

Observation 2: *in the original Roofline model, for a fixed I, an application achieves unexpected performance when accessing different memory levels.*

This observation corresponds to the performance variation between the instances from different classes for a fixed I in Fig. 7. For example, for $I{=}16$, the APP-L1 instance, where the computations are performed on data loaded from the L1 cache, achieves lower performance compared to the other two instances (APP-L3 and APP-D), which operate on data transferred from significantly "slower" memory levels. Thus, the optimization guidelines deduced from the original

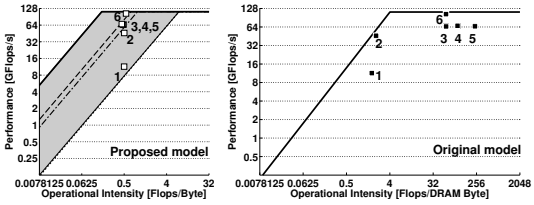


Fig. 9: Matrix multiplication optimization steps (Intel 3770K)

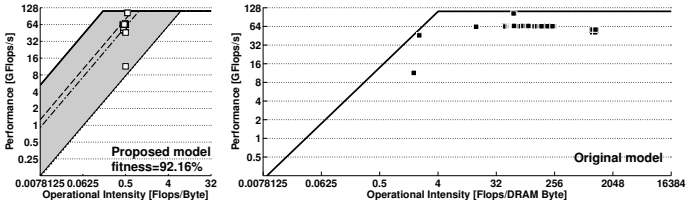


Fig. 10: Optimizing matrix multiplication with about 100 techniques (Intel 3770K)

model suggest that there is a better opportunity for further performance improvements (*i.e.*, space to achieve F_p) for instances from APP-L1 than for APP-L3 and APP-D.

This effect is not observed in the Cache-aware Roofline model, since the instances do not change their I. Actually, the points in the proposed model that correspond to the points of I=16 in the original model reveal that the instances from APP-L1 and APP-L3 are memory bound, and never compute bound. Thus, there are no optimization techniques that can be applied to the APP-L1 instances to provide performance improvements as suggested by the original model, *i.e.*, to achieve the peak performance in the compute bound region. In fact, for the instance with the maximum number of iterations (N=2^{20}) no further optimizations can improve its performance since it achieves the peak theoretical bandwidth of the L1 cache. The same contradictory optimization guidelines are also observed for the APP-L3 instances, although in this case the proposed model suggests that the data access pattern can actually be improved to hit cache levels closer to the core (up to F_p).

Observation 3: *in the original Roofline model, an application in the memory bound region is never able to achieve the model's maximum attainable performance.*

Let us consider the instances from each class with a single iteration, namely, I=1/64 for APP-L1, I=1/4 for APP-L3, and I=16 for APP-D (marked with "1^{st}"), where all memory operations access the DRAM. Although the three instances belong to different classes, they represent a set of basic tests with different I that can hit the roof of the original model in both compute and memory bound regions. However, as evidenced in Fig. 7, only the APP-D instance is able to achieve the $F_a(I)$, while the other two instances are clearly below the roof. This is due to the fact that the original model mainly considers the peak bandwidth between the DRAM and the last level cache (LLC) (B_D in Fig. 1). This is not realistic, since the memory operations must traverse the whole memory hierarchy between the Core and the DRAM, *i.e.*, the bandwidth from the DRAM to the Core ($B_{D\rightarrow C}$ in Fig. 1) should be considered. For these reasons, the points obtained for the same instances in the Cache-aware model are clearly in line with the modeled attainable performance in the memory bound region (see Fig. 8).

4 Application Optimization

In order to evidence the insightfulness of the proposed and original models for a real-world application, Fig. 9 shows experimental results obtained when optimizing the dense matrix multiplication of size N=M=K=2304. The numbers in Fig 9 reflect different optimization techniques applied, namely: basic implementation (1); improved access pattern by transposition (2); blocking the data to fit in L3 (3), L2 (4), and L1 (5); and MKL implementation (6).

The proposed model places the basic implementation (1) in the compute bound region suggesting that, although limited by DRAM, it can be improved to hit F_p. Applying the subsequent techniques gives performance improvements towards the expected cache level ceilings, without changing I, until reaching the suggested F_p with the highly optimized MKL implementation. In contrast, the original model places the basic implementation in the memory bound region, suggesting that F_p cannot be achieved. The predicted memory ceiling was reached with (2), thus the original model suggests that no optimization can further improve performance. When plotting the other optimizations there is a sudden shift of I towards the compute bound region. However, those improvements would never be reached by following the initial suggestions of the original model.

Figure 10 shows results obtained for about 100 optimization techniques with different access patterns and block sizes. While the proposed model shows that different optimization techniques result in coherent performance improvements with fixed I, the optimization process in the original model is more difficult due to a high variation of I.

5 Conclusion

In this paper we propose the Cache-aware Roofline model that introduces a new perspective to the original Roofline modeling concept [10] by revealing architectural details that were previously disregarded, and by significantly improving the guidelines for application optimization. Highly accurate results (rRMSE$\approx$0.1, fitness$>$90%) were obtained for different architectures when verifying the proposed model by relying on hardware performance counters. We envision that the importance of the proposed model will be higher in future architectures, *e.g.*, GPUs and Xeon Phi [2].

Acknowledgments

This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under projects PTDC/EEI-ELC/3152/2012, PEst-OE/EEI/LA0021/2011, and PTDC/EEA-ELC/117329/2010. F. Pratas also acknowledges the FCT scholarship SFRH/BPD/87734/2012.

References

[1] "Intel 64 and IA-32 Architectures Software Developer's Manual," 2012.
[2] "The Intel Xeon Phi Coprocessor 5110P: Highly-parallel Processing for Unparalleled Discovery," 2012.
[3] M. Hill and M. Marty, "Amdahl's Law in the Multicore Era," *IEEE Computer*, vol. 46, 2008.
[4] S. Kamil *et al.*, "An Auto-Tuning Framework for Parallel Multicore Stencil Computations," in *IPDPS*, 2010, pp. 1–12.
[5] J. Lorenzo *et al.*, "Study of Performance Issues on a SMP-NUMA System using the Roofline Model," *PDPTA*, 2011.
[6] C. Nugteren and H. Corporaal, "The boat hull model," in *CF*, 2012, pp. 203–112.
[7] Y. Sato *et al.*, "Performance tuning and analysis of future vector processors based on the roofline model," in *MEDEA*, 2009.
[8] R. Steinmann, "Applying the Roofline Model," *MSc Thesis, Advanced Computing Laboratory, ETH Zürich*, 2012.
[9] S. Williams, "Auto-tuning Performance on Multicore Computers," *PhD Thesis, University of California at Berkley*, 2008.
[10] S. Williams *et al.*, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," *CACM*, vol. 52, no. 4, pp. 65–76, 2009.