A roofline model based on working set size for embedded systems

Seong Jin Cho, Seung Hyun Yun, and Jae Wook Jeon^{a)}

*The College of Information and Communication Engineering,
Sungkyunkwan University, Suwon, 440746, Korea*

a) jwjeon@yurim.skku.ac.kr

Abstract: A “roofline model” is a system performance and optimization guide for programmers and system engineers to apply in the design of future architectures. We review a conventional roofline model and propose a working set size (WSS)-based roofline model. Because of the recent increase in the performance of embedded systems, we investigated embedded systems based on an ARM architecture. Our proposed scheme presents practical guidelines for a multi-core architecture when the program uses a small WSS.

Keywords: roofline model, embedded system, working set size

Classification: Electron devices, circuits, and systems

References

- [1] M. Gries: VLSI J. **38** (2004) 131. DOI:10.1016/j.vlsi.2004.06.001
- [2] K. Furlinger, M. Gerndt and J. Dongarra: ICCS (2007) 815.
- [3] J. Paul, D. Thomas and A. Bobrek: IEEE Tran. VLSI **14** (2006) 868. DOI:10.1109/TVLSI.2006.878474
- [4] S. W. Williams, A. Waterman and D. A. Patterson: Commun. ACM **52** (2009) 65. DOI:10.1145/1498765.1498785
- [5] H. Jia, Y. Zhang, G. Long, J. Xu and Y. Li: Euro-Par (2012) 920.
- [6] C. Nugteren, G. Braak and H. Corporaal: ADAPT (2014) 8.
- [7] R. Steinmann: Master thesis University of ETH Zurich, Swiss (2012).
- [8] M. Spierings and R. Voort: Master thesis University of Eindhoven, Nederland (2011).
- [9] A. Ilic, F. Pratas and L. Sousa: IEEE Comput. Archit. Lett. **99** (2013). DOI:10.1109/L-CA.2013.6
- [10] STREAM: Sustainable Memory Bandwidth in High Performance Computers, <http://www.cs.virginia.edu/stream/>.

1 Introduction

The latest embedded systems, such as in smart phones, and tablet PCs, with performance comparable to that of a desktop PC can serve consumers who want to operate highly complex applications. Many mobile companies have now released recent smart phones with state-of-the-art applications processors (APs), consisting of a multi-core, a high performance GPU, and many

dedicated IPs. Usually, an embedded system has millions of design possibilities [1].

To design the embedded system appropriately and to evaluate its performance, many benchmark consortiums have released benchmark suites, such as the “SPEC CPU” [2]. Furthermore, scenario-based benchmark programs that can consider the consumer are being researched [3]. Another method, modeling, can analyze system performance and suggests future architectures. In particular, Williams et al. [4] proposed the “roofline model,” which provides insights on system performance and can optimize the performance of the kernel of a system. This model can evaluate the floating-point performance of embedded systems in a heterogeneous parallel computer architecture.

The roofline model has been studied by many researchers. Jia et al. [5] and Nugteren et al. [6] proposed the GPU roofline and DVFS algorithm on the GPU. For easy construction of the roofline model, Steinmann [7] developed and distributed a tool that automatically records the kernel *operation intensity* on a x86 architecture. Spierings et al. [8] presented the overall embedded roofline model. However, they focused mainly on the Intel architecture and only briefly explained the ARM architecture. Ilic et al. [9] explored the cache roofline above the memory bandwidth. However, the cache-aware roofline model considers the peak performance of the cache only and it does not consider the WSS. Usually, when the kernel has a small WSS, expecting peak performance is difficult to scale using a maximized core.

In this letter, we introduce a roofline model based on WSS and evaluate ARM-based embedded systems. We then compare the conventional roofline model with our proposed roofline model and explain the optimization of the kernel for the embedded system.

2 Roofline model based on WSS

2.1 Original roofline model

The original roofline model of Williams et al. [4] provides insights on the floating-point performance from the DRAM to CPU and suggests a guideline for optimizing the performance of a multi-core architecture. The model is determined by two flows. First, it defines the attainable peak performance of the memory and the CPU bound for the system. Second, it establishes the performance ceilings below each bound for optimization.

Let F denote the total number of floating-point operations executed in each core, and M represent the total number of memory operations on the DRAM (Fig. 1(a)). When a kernel is executed on this system, it can be characterized by floating-point operation per byte access (F/M), called the *operation intensity* (I), and the peak performance ($P_F = F/s$), where the bandwidth of the kernel that accesses DRAM, B_m , represents the peak performance per *operation intensity*. So, the original roofline model for peak performance is determined by:

$$P_o(I) = \min \left\{ B_m \times I, \sum_{i=0}^n F_n \right\} \quad (1)$$

Thus, the solid line in Fig. 1(b) shows the peak performance of the memory and the CPU of the original roofline model. The slope line of the roofline model corresponds to the memory bound and the top line of the model corresponds to the CPU bound.

The performance ceilings are obtained by calculating the lowest ceiling first. If the kernel has enough parallelization for each core, then an instruction-level parallelism (ILP) ceiling is defined by $P(ILP) = C_f \times C_{num}$, where C_f , and C_{num} are the frequency of the core, and the number of cores, respectively. The next lowest ceiling, single-instruction-multiple-data (SIMD), is calculated from S_w (SIMD throughput), as $P(SIMD) = P(ILP) \times S_w$. Finally, the complex operation fused-multiply-add (FMA), if supported by the core, can be defined by $P(FMA) = P(SIMD) \times FMA \text{ operation/cycle}$. Usually, the lower ceiling can be optimized more readily than the higher ceiling.

2.2 Cache roofline model

A limitation of the original roofline model, given a fixed *operation intensity*, is that it cannot increase its performance when the kernel is bound to the peak performance of the memory (Fig. 1(b)). Additionally, it is difficult to modify the *operation intensity* of kernel X to $\tilde{X}$. Thus, Ilic et al. [9] suggested that the cache roofline model exploits kernel X so that it goes to $\tilde{X}$, the upper bound of the memory, without modifying the *operation intensity*.

Likewise with the original roofline model, the kernel accesses the cache instead of DRAM, so the bandwidth of the cache represents peak performance per *operation intensity* in the cache region. Thus, the cache roofline model for peak performance of each cache is determined by:

$$P_c(I) = \begin{cases} \min\{\sum_{i=0}^n B_{cnL1} \times I, \sum_{i=0}^n F_n\}, & \text{if regions} = B_{L1} \\ \min\{B_{LLC} \times I, \sum_{i=0}^n F_n\}, & \text{if regions} = B_{LLC} \end{cases} \quad (2)$$

where B_{L1} represents the bandwidth between the core and L1 cache and B_{LLC} represents the bandwidth between the L1 cache and the Last Level Cache (Fig. 1(a)). If system has more cache levels, then the higher bandwidth of the cache will be added to the cache roofline model. Note that this model is added to original roofline model P_o . Likewise, with original roofline model, it is created once per system, not once per kernel.

2.3 Roofline model based on WSS for embedded systems

Although the cache roofline model can increase the performance of the kernel in the cache regions for the kernel, it can only consider the ideal peak performance of the system. Fig. 2(a) and 2(c) show the peak bandwidth of the embedded system obtained from a micro-benchmark program. This benchmark program consisting of assembly code can evaluate peak performances, such as simple reading, and writing functions, in each cache region. The cache roofline model, shown in Fig. 1(b) was established based on the peak performance obtained from a micro benchmark program.

Fig. 1. (a) Multicore architecture, (b) Original roofline model (Intel i5-720)

However, the purpose of the roofline model was to provide guidelines for performance optimization. With real-world applications, the cache cannot achieve peak performance because of the loop overhead, communication among the cores, dependency, and other reasons. Thus, we focused on the WSS of real-world applications and the effects on the bandwidth of a multi-core embedded system. Fig. 2(b) and 2(d) depict the average bandwidth and scalable point on each system using benchmark program. Usually, the L1 cache bandwidth is faster than other level caches or main memory. However, if WSS is smaller than scalable point, a multi-core model has *lower* problem-solving performance due to communication overhead among the cores.

To evaluate performance, we used a stream benchmark [10] based on OpenMP that provides a convenient parallel program for application developers. To exploit the cache regions, we modified a stream benchmark that supports measuring memory bandwidth. The performance of real-world application can attainable for the results of the modified stream benchmark, not the micro benchmark that executes simple read/write functions. Thus, we introduce a WSS-based roofline model and determined its performance:

$$P(I) = \min \left\{ I \times \max \{ B_{single}, B_{multi} \}, \sum_{i=0}^n F_n \right\} \quad (3)$$

$$B_{single} = \{ B_{c0L1}, B_{LLC} \}, B_{multi} = \left\{ \sum_{i=0}^n B_{cnL1}, B_{LLC} \right\} \quad (4)$$

where B_{single} and B_{multi} are the maximum bandwidth between single-core and multi-core, respectively. The proposed model suggests a scalable point between single-core and multi-core due to considering the maximum bandwidth of WSS.

Fig. 2. (a) Peak (Odroid-X), (b) Average (Odroid-X), (c) Peak (Arndale-5250), (d) Average (Arndale-5250)

3 Evaluation

3.1 Experiment environment

The roofline model based on WSS was evaluated under the experiment environment given in Table I. In Algorithm 1, we describe the roofline model based on WSS. To measure the bandwidth and the number of floating-point operations, we made a kernel device driver for the performance-monitoring unit (PMU), which can best monitor the overall exact status of the system.

Table I. Experiment environment

System name	Odroid-X	Arndale-5250
Architecture	ARMv7 Cortex-A9	ARMv7 Cortex-A15
Total Cores	Quad @ 1.4 GHz	Dual @ 1.7 GHz
L1/L2 cache	32 KB-D, 32 KB-I/1 MB	32 KB-D, 32 KB-I/1 MB
DRAM	LP-DDR2 1 GB (400 Mhz)	LP-DDR3 2 GB (800 Mhz)
FP Architecture	VFPv3	VFPv4
Peak DRAM (GB/s)	6.4	12.8
Peak Gflops/s (DP, SP)	5.6, 22.4	6.8, 13.6

Algorithm 1 The roofline model based on WSS

```

1: procedure BANDWIDTH (c: the number of core, b[ ]: the bandwidth
   of each WSS, w[ ]: WSS)
2:   if c == single then
3:     all cores  $\leftarrow$  core 0;
4:   end if
5:   repeat
6:     all cores: flush();
7:     start measure();
8:     core 0: job forked();
9:     all cores: access the data;
10:    all cores: execute the memory and cpu operations;
11:    all cores: wait for other cores;
12:    core 0: join();
13:    stop measure();
14:     $bw = \text{total bytes from Memory} / \text{time}$ ;
15:     $b[ ] = \max(b[ ], bw)$ ;
16:  until sizeof(w[ ])
17: end procedure
18: procedure PEAKPERFCPU
19:   start measure();
20:   all cores: execute the operations;
21:   stop measure();
22:   return (SIMD * 2 + VFP) / time;
23: end procedure
24: procedure WSSROOFLINE (w[ ], max OI)
25:    $b[ ], f = \text{NULL}$ ;
26:   BandWidth(single, b[ ], w[ ]);
27:   BandWidth(multi, b[ ], w[ ]);
28:    $f = \text{PeakPerfCPU}()$ ;
29:   repeat
30:     repeat
31:        $P = \min(b[ ] * i, f)$ ;
32:     until  $i < \text{max OI}$ 
33:   until sizeof(b[ ])
34: end procedure

```

3.2 Results and discussion

Fig. 3(a) and 3(b) depict the roofline model based on the WSS of the embedded system. To obtain better scalability performance at the convex point than at other points, we recommend using a convex point WSS in multi-core programming. In the Odroid-X case, the regions that included the L2 cache through DRAM showed worse performance than the Arndale-5250 because of the poor L2 prefetching performance in the Odroid-X. To maximize the scalability of the multi-core, the Odroid-X uses a WSS that includes 32 KB through 512 KB, and Arndale-5250 uses a WSS that includes 128 KB through 1 MB.

Fig. 3. Roofline model based on WSS (a) Odroid-X, (b) Arndale-5250

Fig. 4 shows optimization of the kernel that carries out matrix multiplication. The *operation intensity* of the kernel O is $1/16$, and it has a 16 KB WSS. In this case, the conventional roofline model states that it must fit the L1 bandwidth using the maximum threads for peak performance. However, it is difficult work for a real-world program and does not meet the performance because WSS is less than the scalable point. Thus, using a single thread on kernel O guarantees faster performance than the multi-core thread. On the other hand, kernel X that has the 512 KB WSS can reach kernel $\hat{X}$ using multi threads. In the CPU bound, the ARM processor does not yet support the SIMD function for double-precision operation, although its performance has been increased due to consumer demand. Thus, it would be premature to use the ARM processor in scientific applications.

Fig. 4. Optimization on the roofline model

4 Conclusions

We propose a roofline model based on WSS and evaluated it for embedded systems. We also compared the conventional roofline model and the WSS-based roofline model. The proposed model using a single thread shows better performance in WSS regions below the scalable point. On the other hand, it guarantees scalability at the convex regions in the model when operating with a maximized core.

Acknowledgments

This research was supported by Basic Science Research Program through NRF of Korea, funded by MOE (NRF-2010-0020210).