Roofline Model applied to NUMA and Heterogeneous Memories

Nicolas Denoyelle and Aleksandar Ilić

Inria - France – INESC-ID – Portugal
Nicolas.Denoyelle@inria.fr, ilic@sips.inesc-id.pt

Abstract. The ever growing complexity of high performance computing systems, has made impossible for a single expert to master them entirely. . Recently, the Roofline Model has earned a great popularity modeling hardware and software, thanks to its simplicity and yet his efficiency. In this short paper we present an extended use of the traditional model, taking into account heterogeneous and NUMA memories in addition to the main DRAM memory.

1 Introduction

Emerging memory technologies (e.g non-volatile memory, heterogeneous memory architectures, on-package memory), are required to address applications' needs and improve the performance at the cost of a growing hardware complexity.

The memory wall makes data locality increasingly important, especially in this context, to achieve good performance. We think we can leverage the Roofline Model to detect some data locality issues related to memory bandwidth. For instance, a data allocated on low-bandwidth memory, may lead the application to become bandwidth bound. Our extension to heterogeneous and NUMA memory shows that in such sytems, memory bandwidth may vary from one pair (source, destination) to another. Thus bad data locality can be spotted by observing those bandwidth bounds reached by an application.

The remainder of this paper is organized as follow:

The section 2 will describe the model and the most interesting existing improvements. Finally section 3 will briefly describe our implementation to measure memory bandwidths, and the validation of the model.

2 The Roofline Model Then and Now

The original paper [7] depict a machine with two subsystems: a memory and a compute unit, running the instructions dispatched from a common instruction channel.

This idea is drawn on figure 1. Depending on the operational intensity (i.e. the ratio of compute instructions over memory instructions), one unit, the other or both may be saturated with instructions. The width of the memory channel is the bandwidth and the width of the compute channel is the fpeak (floating point

peak) performance. The model assumes there can be no dependency between instructions (they can overlap perfectly) and instructions' lattency is hidden.

On figure 3 is shown the graphical representation of the model. The operational intensity stands on abscissa and the performance stands on the ordinate axis. Top horizontal lines show the fpeak performance for different type of compute instructions and oblique lines show the memory bandwidth for different types of memories. On this representation one can see that: the roofline model ($min(bandwidth * operational\_intensity, fpeak)$) shows whether a pattern of compute/memory instructions interleaving is either compute or memory bound.

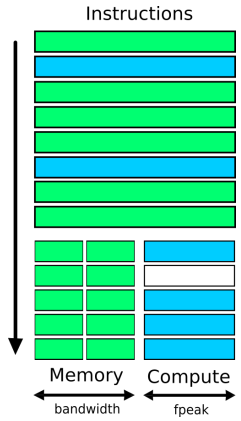

Fig. 1: Roofline model draw

The model has been successfully used in several [4] [6] [5] application optimizations, whether to prove bottlenecks, or measure achievable (or achieved) improvements.

It has been applied to other memory subsystems [3], energy [2], and abstract runtime systems.

Based on the Cache Aware Roofline Model [3] methodology, we applied and validated the model to heterogeneous memory subsystems.

3 Contribution

In our contribution, we developped a tool able to benchmark rooflines and validate them with micro kernels, compatible with intel chips.

Each benchmark (rooflines, and validation) is made of assembly code using best architectures instructions available at compile time.

In brief, the fpeak benchmark repeats compute instructions and outputs $\frac{n*iFlops}{time}$ where n is the number of compute instructions and $iFlops$ is the number of floating point operations computed per instruction.

The bandwidth benchmark repeats coallesced memory instruction on a private[1] buffer of several sizes fitting the memory to benchmark and outputs $\frac{n*iBytes}{time}$ where n is the number of memory instructions and $iBytes$ is the number of Bytes transferred per instruction.

And the validation kernels interleave previously defined benchmarks instructions and output its performance and operational intensity.

We can find several arithmetic floating point operation peaks (addition, multiplication, overlapping multiply add, and fuse multiply add), as well as several bandwidth types (load, store, non temporal load, non temporal store, interleaving of 2 loads and one store), and for several memory levels (L1, L2, L3, (caches ...), NUMA memories, MCDRAM ...) detectable with hwloc [1] library.

[1] Each thread have its own buffer's chunk

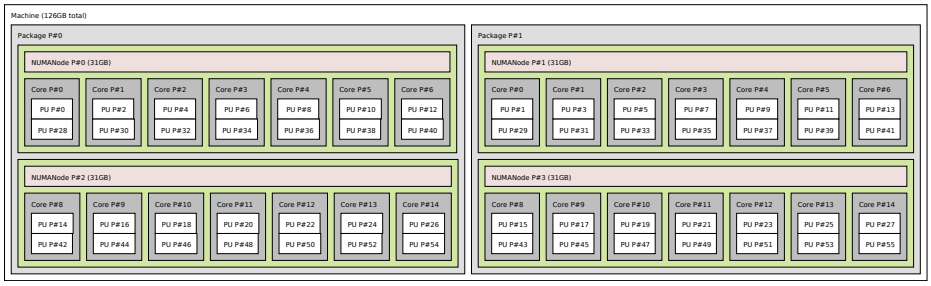

Fig. 2: Machine topology, with hidden caches

Figure 2 shows a representation of such a machine given by hwloc, in the form of nested boxes representing hierarchical inclusiveness of ressources in the machine. Our benchmark can work both in sequential and in parallel. In the former, one benchamrk thread is spawn on first Core (left most) of the machine. In the latter, one benchmark thread per Core of the first NUMA memory node is spawn, and threads measure (Flops and Bytes) are accumulated. The parallel model representation obtained from machine 2 is shown on figure 3. In this model, the sum of each thread load bandwidth on its (non-shared) L1 cache yields a throughput of 761GB/s, whereas the shared local memory (NUMANode:0) yields a throughput of 36GB/s. Remote memory bandwidth are obtained by running the same benchmark as local memory, but explicitly allocating the data on remote memories. On this figure, the lines shows the top measures, and one can notice, on L3 a wider line with less opacity showing the measured bandwidth deviation. We can also notice that for L1 (black line), the points does not stick very well to lines around the ridge. Actually the points does not measure the bandwidths, but instead validate rooflines. They consist of micro kernels of different known operational intensities, interleaving memory and compute instruction to reach rooflines, and show how realistic the model is with those metrics, by measuring each kernel's performance.

4 Conclusion and future work

In this short paper we prestented the Roofline Model and an extension to heterogneous memory's sytem using the Cache Aware Roofline Model methodology. Later on we will validate the interest of this representation with applications.

References

1. Goglin, B.: Managing the topology of heterogeneous cluster nodes with hardware locality (hwloc). In: High Performance Computing Simulation (HPCS), 2014 International Conference on. pp. 74–81 (July 2014)
2. Ilic, A., Pratas, F., Sousa, L.: Beyond the roofline: Cache-aware power and energy-efficiency modeling for multi-cores. IEEE Transactions on Computers PP(99), 1–1 (2016)

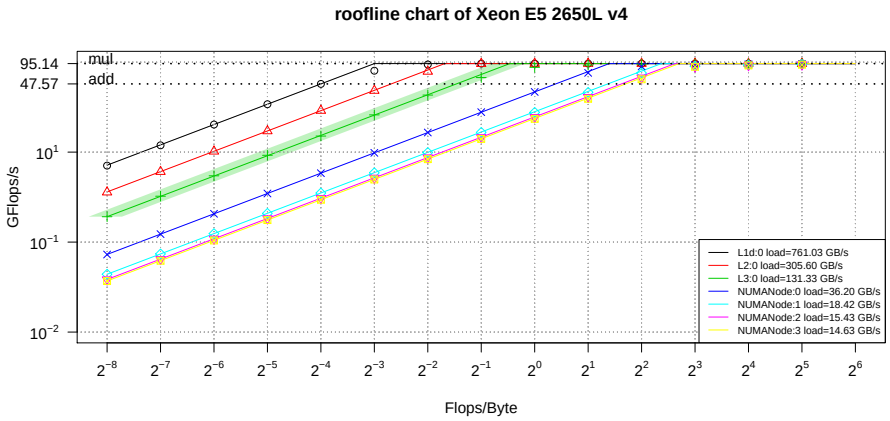

Fig. 3: Example of the roofline chart with several bounds. Units are not instructions but rather, more standard unit, e.g flops for floating point operations (maybe several per instructions) and bytes transferred from memory (several per instruction).

3. Ilic, A., Pratas, F., Sousa, L.: Cache-aware roofline model: Upgrading the loft. IEEE Computer Architecture Letters 13(1), 21–24 (2014)
4. Kim, K.H., Kim, K., Park, Q.H.: Performance analysis and optimization of three-dimensional {FDTD} on {GPU} using roofline model. Computer Physics Communications 182(6), 1201 – 1207 (2011), http://www.sciencedirect.com/science/article/pii/S0010465511000452
5. van Nieuwpoort, R.V., Romein, J.W.: Using many-core hardware to correlate radio astronomy signals. In: Proceedings of the 23rd International Conference on Supercomputing. pp. 440–449. ICS '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1542275.1542337
6. Rossinelli, D., Conti, C., Koumoutsakos, P.: Mesh–particle interpolations on graphics processing units and multicore central processing units. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 369(1944), 2164–2175 (2011), http://rsta.royalsocietypublishing.org/content/369/1944/2164
7. Williams, S., Waterman, A., Patterson, D.: Roofline: An insightful visual performance model for multicore architectures. Commun. ACM 52(4), 65–76 (Apr 2009), http://doi.acm.org/10.1145/1498765.1498785