Bounded Performance: The Roofline Model Uncovered

Kambiz Homayounfar

July 2023

Is your algorithm computation-bound or memory-bound?

In 2009, a seminal paper emerged from computer scientists at University of California, Berkeley.[1]

Their pioneering work introduced the Roofline model, a visually intuitive tool designed to navigate the labyrinth of performance optimization in contemporary computer architectures. The model applies to neural networks, but more on that later.

[1] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009

First, a few words about what motivated the roofline model. Towards the latter part of the 2000s, the computational landscape shifted significantly. Increasingly, the performance of computer systems was not confined by their computational capacity but rather by their memory bandwidth. This was predominantly a result of the widening chasm between processor speed and memory speed — a formidable hurdle colloquially referred to as the "memory wall." Regrettably, the performance models of the era failed to sufficiently encapsulate this evolution.

Against this backdrop, the Roofline model was born. An elegant synthesis of simplicity and intuition, the model clearly depicts the delicate interplay between computational capability and memory bandwidth. It allows users to graphically visualize the zenith of achievable performance as a function of operational intensity, effortlessly discern whether a given piece of code is bounded by computation or memory, and how closely it approaches peak performance.

Furthermore, the Roofline model promotes acute performance optimization techniques like loop unrolling, blocking, and vectorization. Such methods can enhance a kernel's operational intensity, inching it closer to the elusive "roofline." Therefore, the model serves as a diagnostic tool and a beacon guiding the pursuit of performance optimization in the complex landscape of modern computer architectures.

Let's now build intuition for Roofline model. Imagine trying to empty a large swimming pool with a bucket. Your speed would depend on two factors: how fast you can fill and empty the bucket (your "bucket speed") and how much water your bucket can hold (its "size").

Similarly, the Roofline model provides a way to visualize the performance of a computer system, with computational capability analogous to the bucket speed and memory bandwidth comparable to the bucket size.

Bucket speed	Computation capability
Bucket size	Memory bandwidth

On the horizontal axis of the Roofline model, we plot the "Operational Intensity" - the computational equivalent of how much of the bucket you can fill each time.[2] On the vertical axis, we plot "Performance," the computational equivalent of how quickly you can fill and empty the bucket.

[2] If you can only fill it halfway, you're not utilizing its full size.

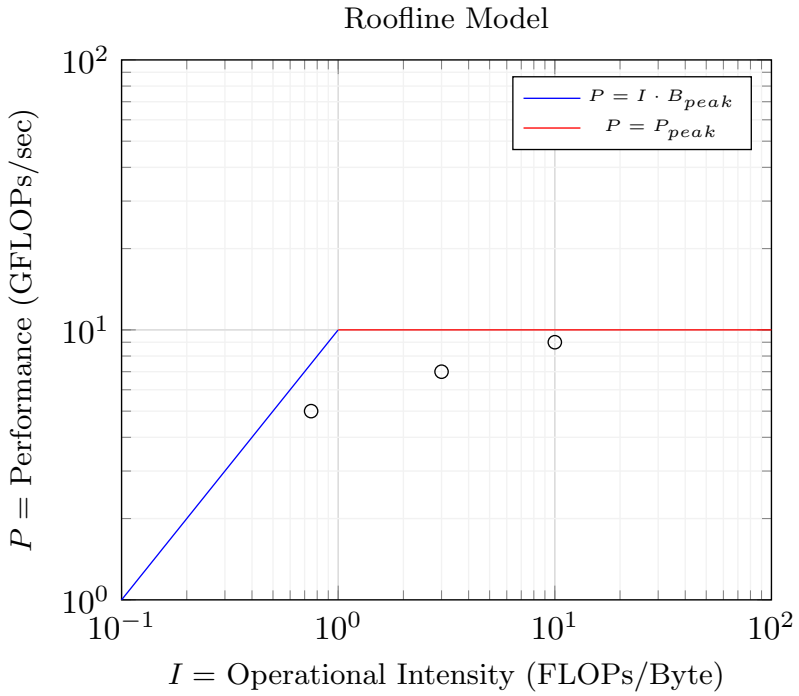

Figure 1: Roofline model illustrating performance limits and operational intensities of various computational kernels marked by ○. The blue line represents the memory-bound region, and the red line represents the computation-bound region.

The roofline itself is shaped like the profile of a roof. The ascending portion represents the increasing performance as we better use our "bucket" (the memory bandwidth), filling it more each time. However, no matter how efficiently we fill the bucket, there's a limit to how quickly we can move — our "bucket speed" (the computational limit). Regarding the pool analogy, if you have a small bucket but move very quickly, you'll hit the point where getting a bigger bucket would help more than moving faster. This is the turning point on the roof, where increasing the bucket size (memory bandwidth) no longer improves performance, and the limiting factor becomes how quickly you can move (computational capability).

For example, consider a computer system that can perform a maximum of 20 billion calculations per second and transfer 5 billion bytes of data per second. It's like having a bucket that can only hold 5 liters and a bucket speed of 20 liters per second. If you can only fill the bucket to 2 liters each time, you're limited by your filling capacity (memory bandwidth) rather than your speed. You're effectively only moving 10 liters per second. This is equivalent to a piece of code (or "kernel") with an operational intensity of 2 calculations per byte. On the other hand, if you can fill the bucket to its full capacity of 5 liters each time, your speed becomes the limiting factor. Even though your bucket is fully utilized, you can only move up to 20 liters per second. This is like a kernel with 10 calculations per byte operational intensity.

In this way, the Roofline model provides a clear, intuitive way to understand and visualize the balance between computational capability

and memory bandwidth in a computer system. It serves as a valuable tool for identifying bottlenecks and opportunities for performance improvement, much like how understanding the balance between bucket capacity and bucket speed can help optimize the task of emptying a swimming pool.

TWO PRIMARY ELEMENTS guide this model: the system's peak computational performance, denoted by P_{peak}, and the system's peak memory bandwidth, symbolized by B_{peak} (Figure 1).

The Roofline model demarcates two predominant regions of performance:

Memory-bound region: Performance, or P, in this area is primarily confined by the memory bandwidth. Here, the operational intensity, symbolized by I, the ratio of floating-point operations to bytes transferred, is often found to be low. This performance within this territory can be expressed via the equation: $P = I \cdot B_{peak}$.

Compute-bound region: In this region, the computational performance of the system, rather than the memory bandwidth, chiefly limits the performance. This typically occurs when the operational intensity soars. The performance within this region adheres to the equation: $P = P_{peak}$.

The shift from the memory-bound region to the compute-bound region is marked when $I \cdot B_{peak} = P_{peak}$ or, in other words, when the operational intensity, I, equals P_{peak}/B_{peak}. Beyond this juncture, a surge in operational intensity fails to elevate performance, which is instead restrained by the system's peak computational performance, P_{peak}.

The term 'roofline' in the model is formed by these two regions, signifying the pinnacle of attainable performance for a given operational intensity. By plotting different computational tasks, or 'kernels', on this model, one can discern whether the task is memory-bound or compute-bound and gauge its proximity to the theoretical peak performance of the system.

DEEP LEARNING methodologies rely heavily on of matrix computations, often inducing substantial data exchanges between the processor and memory. These interactions can be depicted using the Roofline model, where the computational performance, typically quantified in Floating-Point Operations[3] per second (FLOPs/sec), is graphically elucidated as a function of the operational intensity — defined as the ratio of FLOPs to the volume of data transferred (in bytes).

Consider, for instance, the application of the Roofline model to the performance characterization of a convolutional neural network (CNN), a commonly employed structure in image classification tasks. Each convolutional layer within the CNN entails a multitude of matrix multiplications, each of which can be symbolized as a distinct point within the Roofline model. If the operational intensity of these

[3] Often expressed in mega or giga units: MFLOPs and GFLOPs respectively.

calculations is on the lower end, the resultant performance is predominantly memory-bandwidth-bound, manifesting in the 'memory-bound' segment of the Roofline model. Conversely, higher operational intensity calculations encounter performance constraints from the system's peak computational performance, as depicted in the 'compute-bound' region of the model.

By applying the Roofline model, one can discern and capitalize on opportunities for performance optimization within deep learning algorithms. If the computations within a CNN exhibit a memory-bound trend, strategies such as loop blocking or data reuse could be implemented to escalate operational intensity and consequently augment performance. Alternatively, if the computations are identified as compute-bound, approaches such as algorithmic enhancements or the deployment of hardware accelerators could potentially elevate the system's peak computational performance.