CSCI-GA-3033-085

ImageNet Analysis using Roofline Model

Abhishek Verma

April 8, 2022

Abstract

The arrival of GPU has led Artificial Intelligence to a new era because of the computation power that it accompanies. As with any new hardware, performance benchmarking is an important metric. It helps application engineers understand bottlenecks in their applications and increase development efficiency. It can also help infrastructure engineers in bringing the cost down. In this project, we use the Roofline model - an easy-to-understand, visual performance model that offers insights to programmers and architects on improving parallel software and hardware for floating-point computations[1]. Specifically, we compare the performance of various neural networks on different GPUs.

1 Introduction

We use the term operational intensity(OI) to mean operations per byte of DRAM traffic. We define total bytes accessed as those that go to the main memory after they have been filtered by the cache hierarchy. That is, we measure traffic between the caches and memory rather than between the processor and the caches. Thus, OI suggests the DRAM bandwidth needed by a kernel on a particular computer[1].

The Roofline model ties floating-point performance, operational intensity, and memory performance together in a two-dimensional graph. Peak floating-point performance can be found using the hardware specifications or microbenchmarks. A neural network is a point on the two-dimensional graph and this point indicates the ability of this net to exploit the underlying kernel[1].

Figure 1 provides a summary of the Roofline model. The graph can be divided into regions as illustrated in Figure 2.

In this project, we measure the performance of three neural networks - Resnet18, Resnet34 and Resnet50 on two GPUs - Nvidia Tesla V100 and Nvidia A100 Tensor Core GPU. We compute the theoretical values followed by experimental values and make a comparison among them.

2 Theoretical Computations

The GPU model can be found out using the command:

```
nvidia-smi --query-gpu=gpu_name,gpu_bus_id,vbios_version --format=csv
```

Since the release of Ampere GPUs, Pytorch has been using TF32 by default[8]. Hence, TF32 peak FLOP must be used instead of FP32 peak FLOP. The OI computation for each GPU based on the data provided by the manufacturer[6][7] is shown in Table 1.

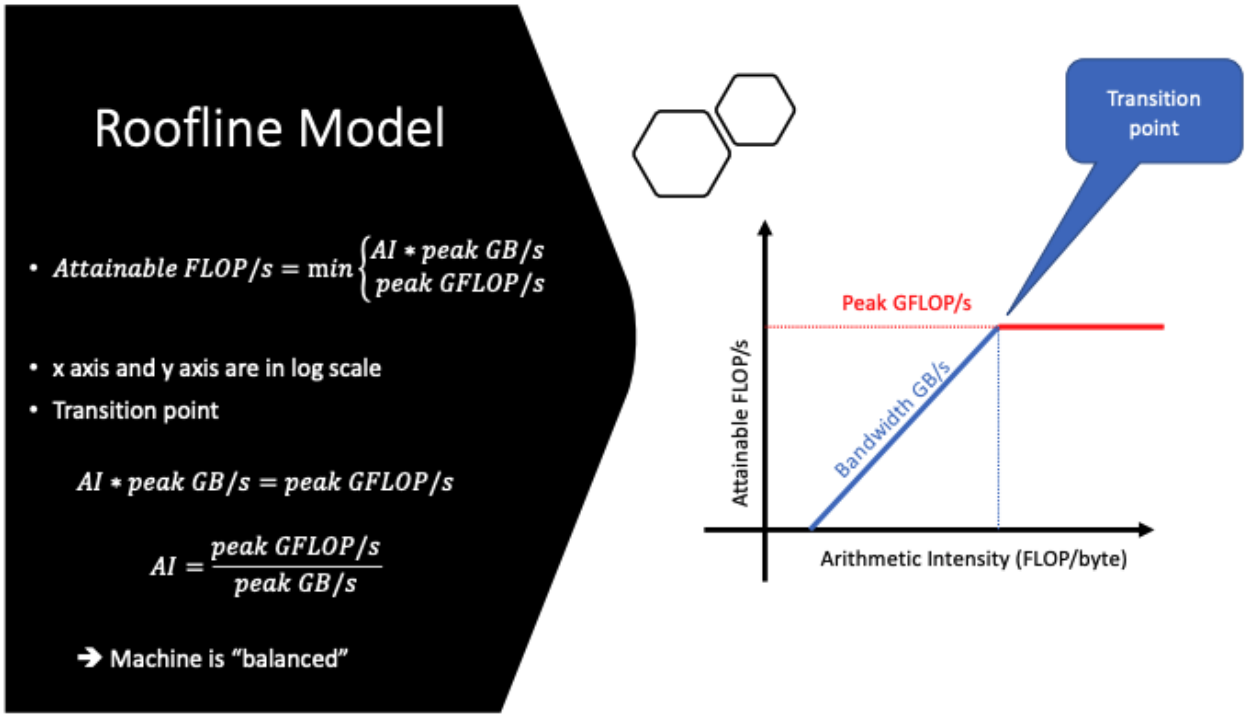

Figure 1: Roofline Model

Table 1: Theoretical Operational Intensity of GPUs

	V100	**A100**
Peak attainable TFLOP/s	125	312
Peak memory bandwidth (TB/sec)	0.900	1.555
Operational Intensity (FLOP/byte)	138.89	200.64
Memory (GB)	4	4

The input image dimension is 224 x 224 x 3 (Assuming 32 bit, size = 224x224x3x4 bytes = 0.6 MB).

The FLOP needed by a neural network in a single forward pass is dominated by Convolution layers. FLOP in a single Conv2D layer is given by

```
[(Ci * Kw * Kh) + (Ci * Kw * Kh - 1) + 1] * (Co * Ho * Wo)
where,
Ci : Input Channels
Co : Output Channels
Kw : Convolution Kernel Width
Kh : Convolution Kernel Height
Ho : Output Height
Wo : Output Width
```

Computing the total FLOP of a neural network using the above formula is no easy task. There are many layers in a single network. A python library - pthflops[2] was used for this. For a

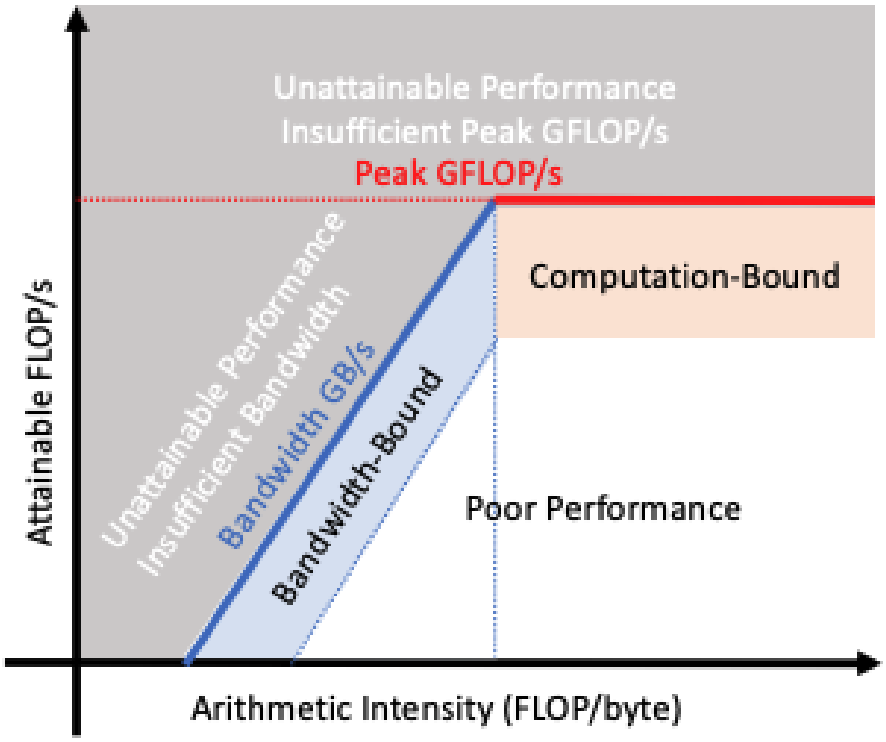

Figure 2: Roofline Regions

detailed report, please refer to the file `theory/theory_flops.log`. The related code can be found in `theory/theoretical_flops.py`.

In order to compute the OI of a neural network, we need the total theoretical memory usage as well. This can be done by adding the size of all intermediate input, outputs and model parameters. This is a tedious task to be done manually. This was accomplished by using the python script - `script/ai.py`. Please refer to the file `theory/theoretical_ai.log` for the runtime logs of this script.

The FLOP for each model along with the corresponding theoretical OI is shown in Table 2.

Table 2: Theoretical OI of different neural networks

	FLOP	**Memory (bytes)**	**Theoretical OI**
Resnet18	1,826,793,960	59509888	30.70
Resnet34	3,682,754,024	1.0995e+08	33.50
Resnet50	4,143,375,336	1.7728e+08	23.37

3 Hypothesis

- On V100 as well as A100 GPUs, Resnet34 will have the highest OI, followed by Resnet18. Resnet50 will be the worst performer. The reason is that theoretical OI of these models follows the same order and the theoretical OI of all the models are greater than that of both the GPUs.
- In both the GPUs, all three neural networks will be memory bandwidth bounded. This is due to the high peak computation capacity and low memory bandwidth of both the GPUs.

4 Experimental Setup

1. Clone [3] repository to get access to the Imagenet model training code. This code is based on the Pytorch framework.
2. Make the following modifications to the code:
   - Instead of running the training on full dataset, randomly sample 2-3 training examples and use these for training.
   - There is no need to validate the model, thus, comment out the validation code in the train() function.
3. Use singularity to run the experiments as it lets us install the required python libraries and provides us an isolated environment.
4. To collect data about the FLOP and memory usage per model, ncu profiling tool is used.
   (a) For the FLOP computation, following metrics were used:

```
Floating precision FLOP =
smsp__sass_thread_inst_executed_op_fadd_pred_on.sum
+ smsp__sass_thread_inst_executed_op_fmul_pred_on.sum
+ 2*smsp__sass_thread_inst_executed_op_ffma_pred_on.sum

Double precision FLOP =
smsp__sass_thread_inst_executed_op_dadd_pred_on.sum
+ smsp__sass_thread_inst_executed_op_dmul_pred_on.sum
+ 2*smsp__sass_thread_inst_executed_op_dfma_pred_on.sum

Half precision FLOP =
smsp__sass_thread_inst_executed_op_hadd_pred_on.sum
+ smsp__sass_thread_inst_executed_op_hmul_pred_on.sum
+ 2*smsp__sass_thread_inst_executed_op_hfma_pred_on.sum

Total FLOP =
Floating precision FLOP
+ Double precision FLOP
+ Half precision FLOP
```

   (b) For the byte computation, following metrics were used:

```
Total bytes = dram__bytes_read.sum + dram__bytes_write.sum
```

5. OI can be computed using:
   `OI = Total FLOP / Total bytes`

6. To collect data about the total GPU runtime, nsys profiling tool is used.

7. Experimental FLOP/s of a model on a given GPU can be computed using:
   `FLOPS (FLOP/s) = Total FLOP / Total GPU runtime`

8. Attainable FLOPS can be computed using:
   `Attainable FLOP/s =`
   `min {OI * system peak memory bandwidth , system peak attainable FLOP/s}`

Note: A step-by-step guide to run the profiler can be found in the README file.

5 Results

- Results of different models on V100 GPU are illustrated in Table 3 and Figure 3.
- Results of different models on V100 GPU are illustrated in Table 4 and Figure 4.

Table 3: V100 - Experimental Results

Model	**Experimental OI** (flop/byte)	**Experimental FLOP/s** (GFLOP/s)	**Attainable FLOP/s** (GFLOP/s)
Resnet18	16.54	4991	14886
Resnet34	16.76	4340	15084
Resnet50	12.16	5001	10944

Table 4: A100 - Experimental Results

Model	**Experimental OI** (flop/byte)	**Experimental FLOP/s** (GFLOP/s)	**Attainable FLOP/s** (GFLOP/s)
Resnet18	17.65	5469	27445
Resnet34	18.33	4796	28503
Resnet50	13.62	6311	21179

6 Observations

We observe the behavior of OI, Experimental FLOP/s and Attainable FLOP/s.

6.1 Operational Intensity

- The OI of Resnet18 and Resnet34 are quite close but it is lower for Resnet50 - in both the GPUs. This is in accordance with our theoretical calculations. It is difficult to reason out this behavior without analyzing the underneath neural network architecture layer-by-layer. Intuitively, since Resnet50 has a lot more hidden layers than Resnet18 and Resnet34, at some

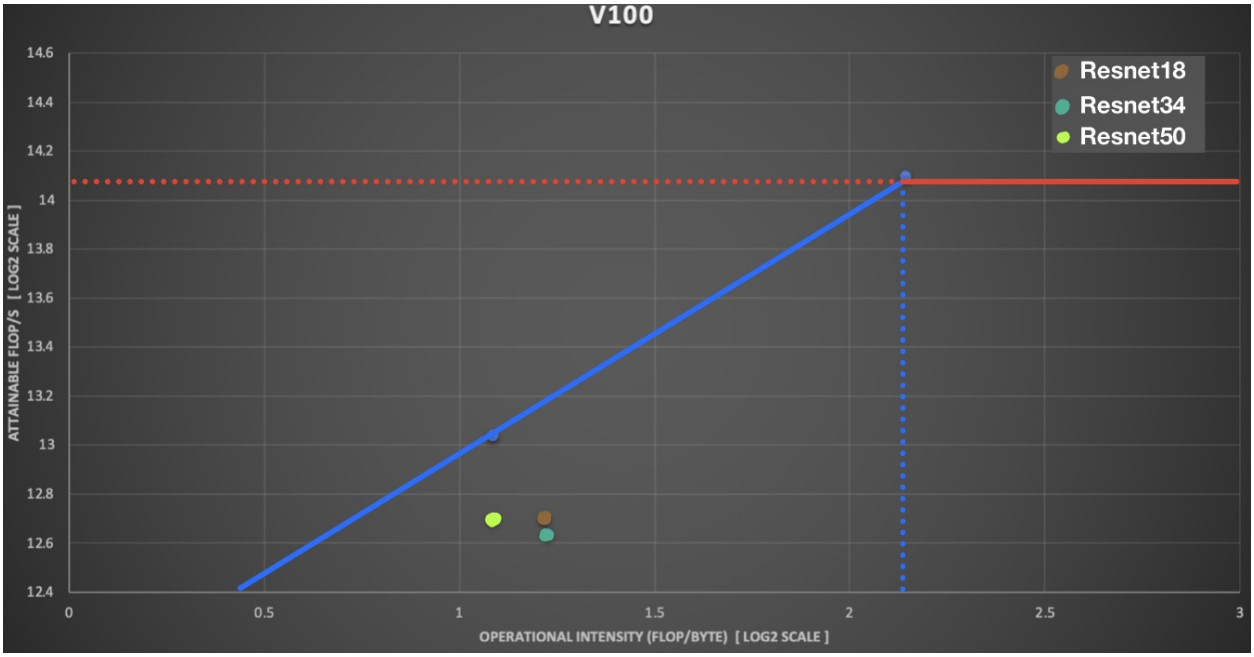

Figure 3: V100 - Roofline Model

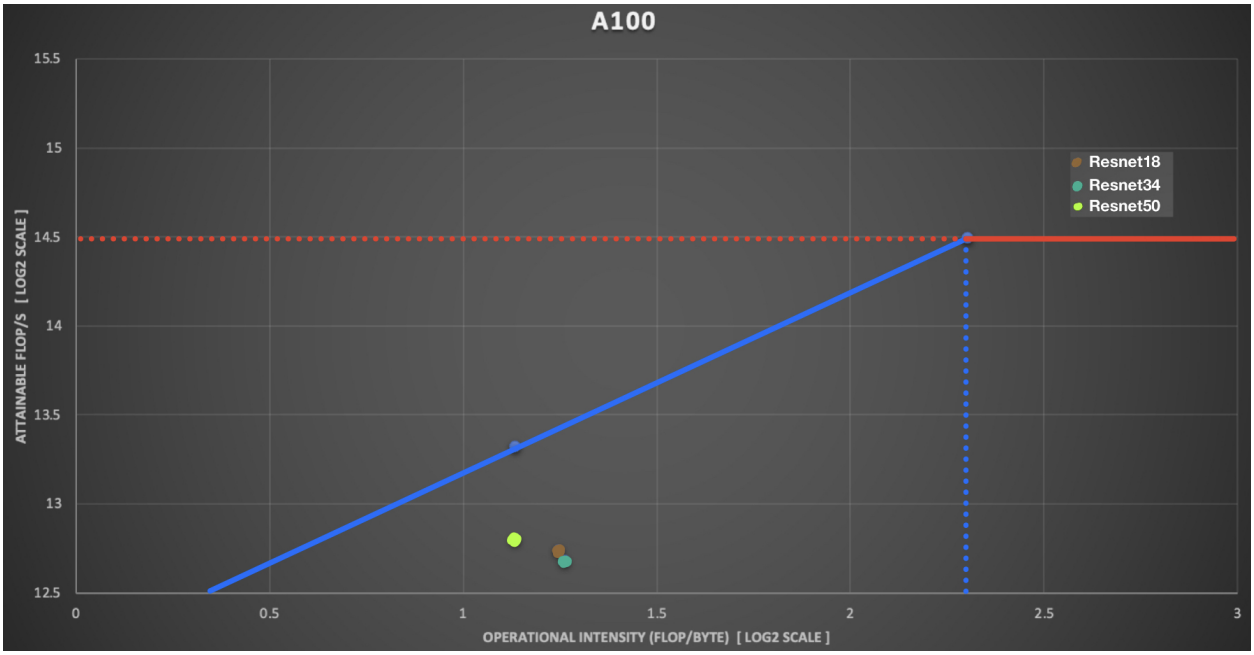

Figure 4: A100 - Roofline Regions

point the number of numerical computations is bound to overshadow the size of input and output of each layer.

- A100 has greater OI compared to V100 (on all the models). In theoretical calculation, we saw that the peak OI of V100 is greater than A100. This shows that all three models utilize A100 in a more efficient manner. Also, all these models fail to utilize the peak performance

of V100.

6.2 Attainable FLOP/s

- On both the GPUs, the attainable FLOP/s are a result of memory bandwidth (and not peak FLOP/s of the GPU). Resnet34 has the highest Attainable FLOP/s because of the underlying high OI. This shows that the performance of all the models is bandwidth-bounded. This makes sense because the both the GPUs have a low memory bandwidth but high peak computational power.

- The Attainable FLOP/s is higher on A100 - in agreement with the theoretical values provided by the manufacturers. This is a result of a higher FLOP/s and memory bandwidth of A100 compared to V100. This also means that all three neural networks are better off running on A100.

6.3 Actual FLOP/s

- In all the cases, actual FLOP/s are lower than attainable FLOP/s. This means that the models are incapable of utilizing the full capacity and performance of the GPUs. This implies that there is a scope for improvement in the computation kernels underneath as the current ones are not fully capable of utilizing the underlying hardware.

- Actual FLOP/s is most for Resnet50. This might be a result of some optimizations in the kernel being used underneath. Resnet50 makes use of many special convolutions (like 1x1) - convolutions like this are easy to optimize compared to bigger convolutions.

7 Conclusion

The experiments solidified the confidence in the hypotheses. Moreover, the experimental performance agrees with all the theoretical relative performances. The three models have different performances because of the underlying limitations on peak FLOP/s and peak memory bandwidth of the GPUs. This is in agreement with the presence of the horizontal line and the ridge point in the Roofline model.

We also identified the gap between actual FLOP/s and attainable FLOP/s. These gaps exist because of software not being able to utilize the underlying hardware to its full potential. We ran the experiments on the Resnet family. This helps in comparing the performance across different Resnet with increased hidden layers. It might be the case that this gap would be lesser for other neural networks. It is near impossible to optimize the kernels for a specific model. Hence, this gap is bound to exist. But with more development in the field, we might see a significant decrease in this gap.

With all the good things in the Roofline model, there are some limitations to it as well. Comparing the peak performance might not be the best way to judge the performance. In actuality, there are a lot of things that go underneath neural network training like GPU parallelization and caching. Performance can be increased by the presence of bigger and better caches.

References

[1] Samuel Williams, Andrew Waterman, and David Patterson, "Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures", 2008

[2] 1adrianb/pytorch-estimate-flops

[3] ImageNet training in PyTorch

[4] Charlene Yang, "Hierarchical Roofline Analysis: How to Collect Data using Performance Tools on Intel CPUs and NVIDIA GPUs", 2020

[5] Kernel Profiling Guide :: Nsight Compute Documentation

[6] nvidia v100 - tensor core gpu

[7] NVIDIA A100 — Tensor Core GPU

[8] CUDA semantics — PyTorch 1.11.0 documentation