GPGPU Programming

Donato D'Ambrosio

Department of Mathematics and Computer Science
Cubo 30B, University of Calabria, Rende 87036, Italy

Lecturer's email: donato.dambrosio [at] unical.it
Lecturer's homepage: http://www.mat.unical.it/~donato

Course's homepage: https://www.mat.unical.it/~donato/gpgpu.html
Course team page: https://bit.ly/3CU9xiR
Code to join the team: **3d98qzr**

Academic Year 2021/22

Table of contents

The Roofline Performance Model

The Roofline Performance Model

The Roofline Performance Model

- Roofline[1] is a model widely used to assess both computational and memory performance of algorithms.
- A compute kernel can be characterized by a its Arithmetic Intensity (or Operational Intensity) defined as the number of floating-point operations per unit of data (similar to the compute-to-global-memory-access ratio dfined in slides 4): $AI\left[\frac{FLOP}{byte}\right]$
- A computational device can be characterize by:
  - Theoretical computational peak performance, i.e. the number of floating-point operations per second: $FLOPS\left[\frac{FLOP}{s}\right]$
  - Memory or bandwidth peak performance, i.e. the number of bytes that can be fetched/stored per second: $BW\left[\frac{byte}{s}\right]$

[1] S. Williams, A. Waterman, D. Patterson (2009). Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM.* 52 (4): 65–76. https://people.eecs.berkeley.edu/~kubitron/cs252/handouts/papers/RooflineVyNoYellow.pdf

The Roofline Performance Model

- Let us consider the vector addition kernel:

  ```
  __global__ void vecAddKernel(float* A, float* B, float* C, int n)
  {
    int i = blockDim.x*blockIdx.x + threadIdx.x;

    if (i < n)
      C[i] = A[i] + B[i];
  }
  ```

  We have:
  - 1 ADD
  - 2 (4 bytes) loads
  - 1 (4 bytes) store

  Resulting in:

  $$AI = 1/(2*4+4) = 1/12$$

- The `int i = blockDim.x*blockIdx.x + threadIdx.x;` statement was not taken into account since it does not involve neither the Global or the Shared Memory.

The Roofline Performance Model

- The above assessment of AI can be difficult for complex kernels.
- Moreover, it does not take into account the actual data load/store operations performed by the GPU, the transaction size (32 bytes), and the involved memory subsystems.
- Nevertheless, the `nvprof` Nvidia profiler can be used. Please, refer to the following paper (co-authored by S. Williams) to properly assess the CUDA kernels' AI:
  - C. Yang, T. Kurth, S. Williams (2020). Hierarchical Roofline analysis for GPUs: Accelerating performance optimization for the NERSC-9 Perlmutter system. *Concurrency Computat Pract Exper.* 2020;32:e5547. https://onlinelibrary.wiley.com/doi/10.1002/cpe.5547
- You can also refer to the following link for an overview of the metrics provided by `nvprof`:
  - CUDA profiler metrics: https://docs.nvidia.com/cuda/profiler-users-guide/index.html#metrics-reference

The Roofline Performance Model

- Since[2] $FLOPS \left[\frac{FLOP}{s}\right]$ can be expressed as the product of $AI \left[\frac{FLOP}{byte}\right]$ and $BW \left[\frac{byte}{s}\right]$:

  $$\frac{FLOP}{s} = \frac{FLOP}{byte} \cdot \frac{byte}{s}$$

  If we consider the log, we obtain:

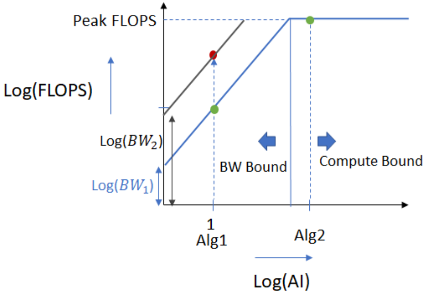
  $$\log(FLOPS) = \log(AI) + \log(BW)$$

- If we interpret the above formula as the equation of a line of the form $y = mx + c$, we have:
  - $y = \log(FLOPS)$
  - $m = 1$
  - $c = \log(BW)$

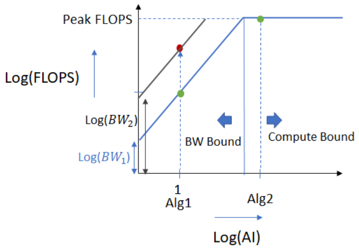
[2]See the article about the explanation of the Roofline plot at following url: https://www.telesens.co/2018/07/26/understanding-roofline-charts/

The Roofline Performance Model

- In the Roofline plot, the peak FLOPS performance is a point on the vertical axis, while the peak memory bandwidth is the intersect point with the vertical axis of the above line of slope 1.
- The following is an example of Roofline plot for two kernels Alg1 and Alg2, being BW_1 and BW_2 the Global and Shared Memory peak bandwidths.

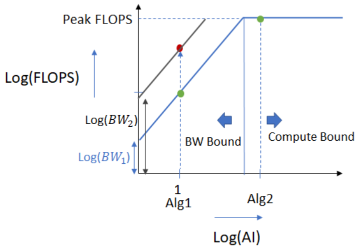


The Roofline Performance Model



- Alg1 lies on the sloping part of the roofline. The low memory bandwidth prevents it from achieving the maximum performance the hardware permits (bandwidth or memory bound). We can:
  - Try to increase its AI (with more operations per byte), moving it along the right along the x axis.
  - Increase the memory bandwidth (e.g., by making more efficient use of the Shared Memory, which has an higher bandwidth peack - see the red line).

The Roofline Performance Model



- Alg2 has a higher arithmetic intensity and lies on the flat part of the roofline plot. It uses more computation per unit data transferred and is already using all available compute resources and achieving peak FLOPS (compute bound).
- To improve the performance of this algorithm, we'd need to invest in hardware with faster and/or larger number of computing resources.

The Roofline Performance Model

- The GTX 980 has:
  - Theoretical performance: 4,612 GFLOP/s (device specs)
  - Theoretical Global Memory bandwidth: 224.32 GB/s (device specs)
  - Shared Memory: to be measured by using microbenchmarks.
- You can use the *gpumembench* microbenchmark[3] to evaluate the Shared Memory throughput of the GTX 980 GPU.
  - GitHub page: https://github.com/ekondis/gpumembench
  - CUDA path in JPDM2 (to be updated in the top level Makefile): `/opt/cuda`
  - CUDA Makefiles options: Compile for the Compute Capability (CC) 5.2 only (previous CC do not compile with CUDA 11 on JPDM2 - you need to change the Makefiles).

[3] E. Konstantinidis, Y. Cotronis, (2016). A quantitative performance evaluation of fast on-chip memories of GPUs, 24th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Heraklion, Crete, Greece, pp. 448-455. (See the Files section of the Lectures channel.)

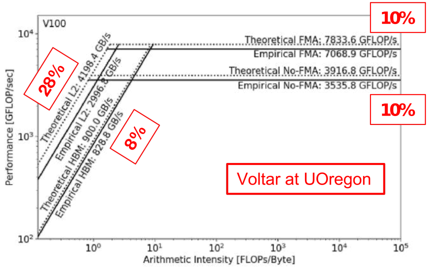
The Roofline Performance Model

- Alternatively, you can use the tools mentioned in Yang et al. (2020)[4] to assess the device performance.
- It is important to know that peak performance can only be obtained by using special CISC instruction modern device relies on, such as:
  - FMA (Fused Multiply Add) z=a*x+y, being z,x,y vectors or scalars
  - 4FMA (quad FMA) z=A*x+z, being A a FP32 matrix and x,z vectors
  - HMMA (Tensor Core): Z=AB+C, being Z,A,B,C FP16 matrices
  - ...
- Such CISC instructions are 32bit or (worse) reduced precision instructions. It is not surprising that scientific compute kernel does not reach device's peak computational performance.

[4] C. Yang, T. Kurth, S. Williams (2020). Hierarchical Roofline analysis for GPUs: Accelerating performance optimization for the NERSC-9 Perlmutter system. *Concurrency Computat Pract Exper.* 2020;32:e5547. https://onlinelibrary.wiley.com/doi/10.1002/cpe.5547

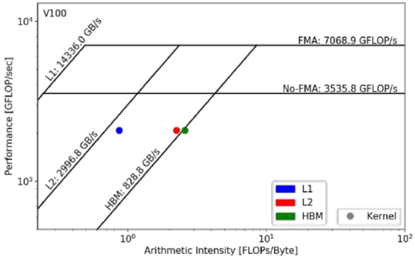
The Roofline Performance Model

- It is important to refer to the proper performance ceiling. For instance, it would be necessary to assess device's FP64 No-FMA peak if your kernel do not use CISC instructions but FP64 instructions only.
- Similarly, it is necessary to assess the bandwidth of different memory subsystems, if used by your kernel.



The Roofline Performance Model

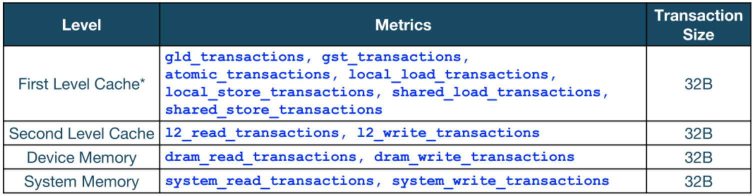
- Here is an example relying on the ERT (Empirical Roofline Toolkit)[5] in which different AIs have been computed for a single kernel for each exploited memory subsystem. Note that the kernel computational performance is always the same.



[5]from: C. Yang, T. Kurth, S. Williams (2020). Hierarchical Roofline analysis for GPUs: Accelerating performance optimization for the NERSC-9 Perlmutter system. *Concurrency Computat Pract Exper.* 2020;32:e5547. https://onlinelibrary.wiley.com/doi/10.1002/cpe.5547

The Roofline Performance Model

- For computing the AI, the following metrics can used:



Level	Metrics	Transaction Size
First Level Cache*	`gld_transactions`, `gst_transactions`, `atomic_transactions`, `local_load_transactions`, `local_store_transactions`, `shared_load_transactions`, `shared_store_transactions`	32B
Second Level Cache	`l2_read_transactions`, `l2_write_transactions`	32B
Device Memory	`dram_read_transactions`, `dram_write_transactions`	32B
System Memory	`system_read_transactions`, `system_write_transactions`	32B

- The command is:

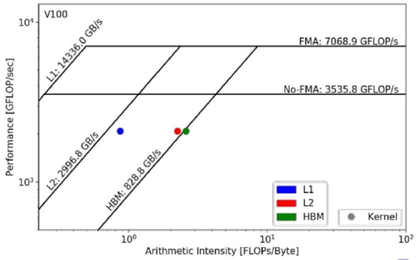
```
nvprof –kernels "name_of_the_kernel_to_be_profiled" –metrics flop_count_dp –metrics gld_transactions –metrics
gst_transactions –metrics l2_read_transactions –metrics l2_write_transactions –metrics dram_read_transactions
–metrics dram_write_transactions –metrics sysmem_read_bytes –metrics sysmem_write_bytes
./name_of_the_cuda_program
```

- Do not forget to multiply the obtained values by 32, since the metrics actually refer to transactions.

The Roofline Performance Model

- Do not forget to multiply the obtained values by 32, since the metrics actually refer to transactions.
- Eventually, to compute kernel's AI and performance, you can refer to the following:

$$\underset{\text{(GFLOP/s)}}{\text{Performance}} = \frac{\mathit{nvprof}\ \text{FLOPs}}{\text{Runtime}}, \quad \underset{\text{(FLOPs/Byte)}}{\text{Arithmetic Intensity}} = \frac{\mathit{nvprof}\ \text{FLOPs}}{\mathit{nvprof}\ \text{Data Movement}}$$


V100
L1: 14336.0 GB/s
L2: 2996.8 GB/s
HBM: 828.8 GB/s
FMA: 7068.9 GFLOP/s
No-FMA: 3535.8 GFLOP/s
L1
L2
HBM
Kernel
Performance [GFLOP/sec]
Arithmetic Intensity [FLOPs/Byte]
10^4
10^3
10^0
10^1
10^2