# GPGPU Programming

Donato D'Ambrosio

Department of Mathematics and Computer Science
Cubo 30B, University of Calabria, Rende 87036, Italy
mailto: donato.dambrosio@unical.it
homepage: http://www.mat.unical.it/~donato

Academic Year 2020/21

# 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.
- From the computational point of view, a 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: $BW\ \left[\frac{byte}{s}\right]$
- A kernel (a CUDA kernel for instance) can be characterized by a quantity called Arithmetic Intensity (aka Operational Intensity) defined as the number of floating-point operations per unit of data: $AI\ \left[\frac{FLOP}{byte}\right]$

[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

- In the case of complex kernels, it can be difficult to assess the Arithmetic Intensity.
- Nevertheless, the `nvprof` Nvidia profiler can be used in the case of CUDA kernels.
- You can refer the following links:
  - CUDA profiler metrics: `https://docs.nvidia.com/cuda/profiler-users-guide/index.html#metrics-reference`
  - An interesting thread on StackOverflow: `https://stackoverflow.com/questions/37732735/nvprof-option-for-bandwidth`

# 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:

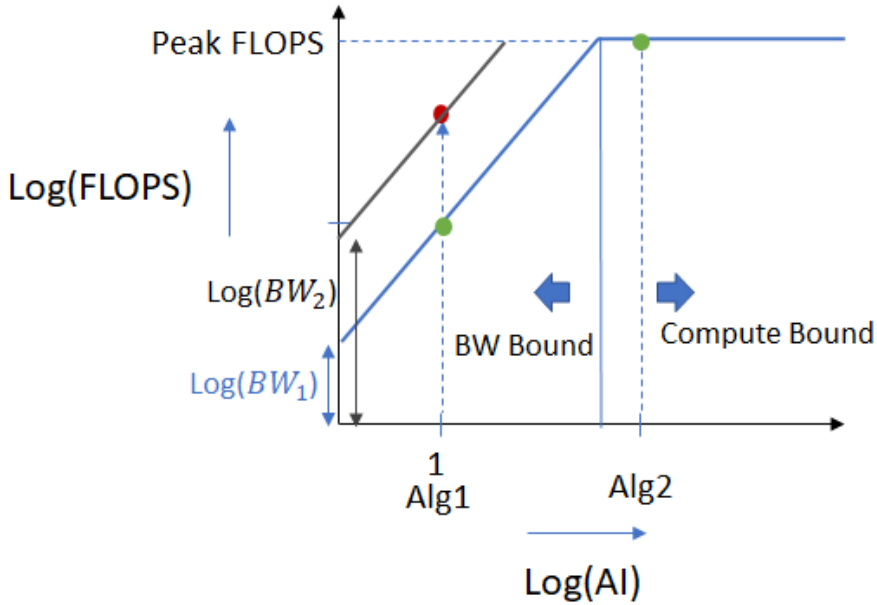
$$\log(FLOPS) = \log(AI) + \log(BW)$$

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

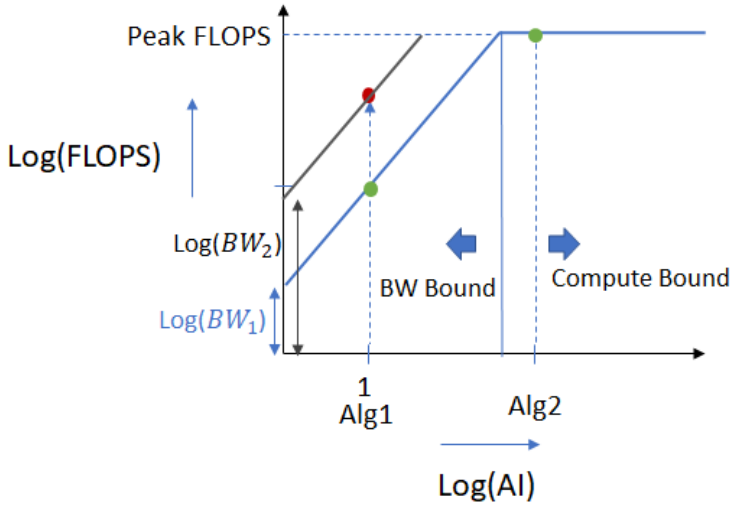
[2] See the article about the explanation of the Roofline plot at following url: `https://www.telesens.co/2018/07/26/understanding-roofline-charts/`

# 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.
- If $BW_1$ and $BW_2$ are the Global and Shared Memory peak bandwidths, respectively, the following is a possible example of the Roofline plot for two kernels Alg1 and Alg2.

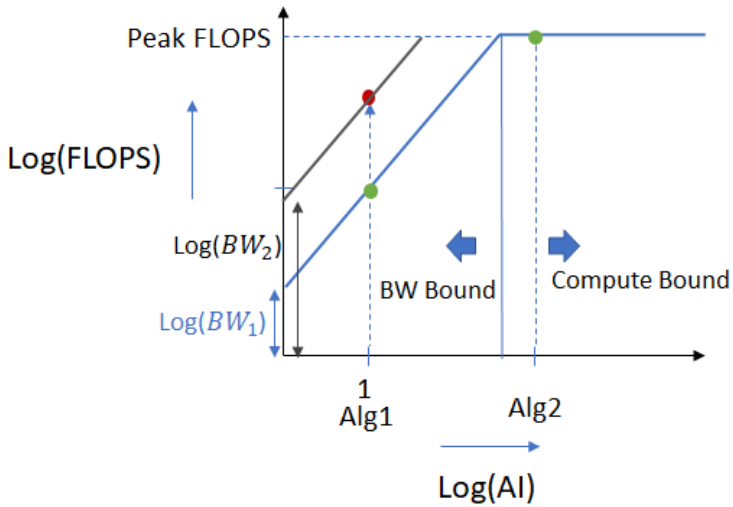


# The Roofline Performance Model




- 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 - seed 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.)