ROOFLINE MODEL

Henry R. Moncada

March 16, 2022

Contents

1 Introduction **1**

2 Concepts or Chararcteristics **1**
2.1 Kernel . . . 1
2.2 proxy applications . . . 2
2.3 Load balancing . . . 2
2.4 Prefixes for representing orders of magnitude . . . 2
2.5 Arithmetic Intensity (AI) . . . 2
2.6 Performance . . . 3

3 DRAM Memory **3**
3.1 Capacity . . . 3
3.2 Bandwidth . . . 3
3.3 Latency . . . 3

4 Computational Loops **3**

5 Roofline Model **4**
5.1 Background . . . 4
5.2 Roofline Model . . . 4

6 Examples **6**

Appendices **10**

1 Introduction

The Roofline performance model is a tool to understand the kernel/hardware limitation.It provides a relatively simple way for performance estimates based on the computational kernel and hardware characteristics. It provides a visually-intuitive way for users to identify performance bottlenecks and motivate kernel/code optimization strategies. The Roofline is a throughput-oriented performance model centered around the interplay between computational capabilities (e.g. peak GFLOP/s), memory bandwidth (e.g. STREAM GB/s), and data locality (i.e. reuse of data once it is loaded from memory). Data locality is commonly expressed as arithmetic intensity which is the ratio of floating-point operations performed to data movement (FLOPs/Bytes).

2 Concepts or Chararcteristics

2.1 Kernel

A kernel is a fundamental component of an software package. A software package can contain multiple kernels (microkernels). A microkernel can be program of a source code, block section of source code, a layer of source code, etc. These microkernels deals only with critical activities in software package.

In the case of operating system (OS), kernels handles many fundamental processes at a basic level, communicating with hardware and managing resources, such as RAM and the CPU. On most systems, the kernel is one of the programs loaded at the beginning of the boot sequence when a computer starts up. It handles the rest of startup as well as memory, peripherals, and input/output (I/O) requests from software, translating them into data-processing instructions for the CPU. The kernel performs a system check and recognizes components, such as the processor, GPU, and memory.

2.2 proxy applications

In high performance computing (HPC), proxy applications (`proxy apps`) are small, simplified codes that allow application developers to share important features of large applications without forcing collaborators to assimilate large and complex code bases. Proxy apps are often used as models for performance-critical computations, but proxy apps can do more than just represent algorithms or computational characteristics of apps. They also capture programming methods and styles that drive requirements for compilers and other elements of the tool chain.

2.3 Load balancing

Load balancing in distributed memory systems refers to the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient and improve performance, typically by moving work from overloaded resources to underloaded resources. Load balancing techniques can optimize the response time for each task, avoiding unevenly overloading compute nodes while other compute nodes are left idle. In short, a load balancer acts as the traffic cop sitting in front of your system manage the requests across all nodes capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one node is overworked, which could degrade performance.

2.4 Prefixes for representing orders of magnitude

Orders of magnitude (in base 10) are expressed using standard metric prefixes, which are abbreviated to single characters when prepended to other abbreviations, such as FLOPS and B (for byte):

Prefix	Abbreviation	Order of magnitude (as a factor of 10)	Computer performance	Storage capacity
giga-	G	10^9	gigaFLOPS (GFLOPS)	gigabyte (GB)
tera-	T	10^{12}	teraFLOPS (TFLOPS)	terabyte (TB)
peta-	P	10^{15}	petaFLOPS (PFLOPS)	petabyte (PB)
exa-	E	10^{18}	exaFLOPS (EFLOPS)	exabyte (EB)
zetta-	Z	10^{21}	zettaFLOPS (ZFLOPS)	zettabyte (ZB)
yotta-	Y	10^{24}	yottaFLOPS (YFLOPS)	yottabyte (YB)

2.5 Arithmetic Intensity (AI)

Arithmetic Intensity (AI), also referred to as Operational Intensity, Computational Intensity is a measure of (Work) floating-point operations (`FLOPs`) performed by a given code (or code section) relative to the (memory traffic) amount of memory accesses (`Bytes`) that are required to support those operations. It is most often defined as a `FLOPs` per `Bytes` ratio of floating-point operations performed to data movement (`FLOP/Byte`). For a given kernel (code, or code section), we can find a point on the `X-axis` based on its Arithmetic intensity (AI).

2.6 Performance

A point in the Y-axis represents the measured performance GFLOP/s. This performance number can be compared against the bounds set by the peak compute performance (Peak GFLOPs) and the memory bandwidth of the system (Peak GB/s) to determine what is limiting performance: memory or compute[1].

3 DRAM Memory

3.1 Capacity

3.2 Bandwidth

3.3 Latency

4 Computational Loops

LOOPS	FLOPS	BYTES	AI = $\frac{FLOPS}{BYTES}$

for (i = 0; i <N; ++i){
  z[i] = x[i]
}
	0 add
	0 mult
		1(8 bytes) loads
		1(8 bytes) write (or data transfer)
			$\frac{0}{1*8+1*8} = \frac{0}{16}$

for (i = 0; i <N; ++i){
  z[i] = x[i]+y[i]
}
	1 add	2(8 bytes) loads
		1(8 bytes) write (or data transfer)
			$\frac{1}{2*8+1*8} = \frac{1}{24}$

for (i = 0; i <N; ++i){
  z[i] = x[i]+y[i]*x[i]
}
	1 add
	1 mult
		2(8 bytes) loads
		1(8 bytes) write (or data transfer)
			$\frac{2}{2*8+1*8} = \frac{1}{12}$

double s = 0.0;
for (i = 0; i <N; ++i){
  s += a[i]* a[i]
}
	1 add
	1 mult
		1(8 bytes) loads	$\frac{2}{1*8} = \frac{2}{8}$

for (i = 0; i <N; ++i){
 I1 = A_offset[i];
 I2 = A_offset[i+1];
 sum = 0.0;
 for (j = 0; j <(I2-I1); ++j)
  sum += A[I1+j]*x[col_index[I2+j]];
 y[i] = sum;
}
	1 add
	1 mult
		2(8 bytes) + 1 (4 bytes) loads
		1(8 bytes) write (or data transfer)
			$\frac{2}{(2*8+1*4)+1*8} = \frac{1}{14}$

for (i = 0; i <N; ++i){
  a[i] = buffer[i]+b[i];
  c[i] = buffer[i]+d[i];
}
	2 add	3(8 bytes) loads
		2(8 bytes) write (or data transfer)
			$\frac{2}{3*8+2*8} = \frac{2}{40}$

for (i = 0; i <N; ++i){
  a[i] = buffer[i]+b[i];
}
for (i = 0; i <N; ++i){
  c[i] = buffer[i]+d[i];
}
	2 add	4(8 bytes) loads
		2(8 bytes) write (or data transfer)
			$\frac{2}{4*8+2*8} = \frac{2}{48}$

5 Roofline Model

5.1 Background

- x-axis - Computational Intensity or Aritmetic Intensity $[FLOP/Bytes]$
- y-axis - Attainable Peak Performance $[GFLOP/s] = [(FLOP \times 10^9)/s]$
- m slope - Bandwidth $[GFLOP/s]/FLOP/Bytes]] = [GBytes/s]$
- $y = mx + b$, where $b = 0$,

$$\text{Attainable Peak Performance} = \text{Bandwidth} \times \text{Computational Intensity} + b$$

  $b = 0$, Since Attainable Peak Performance is equal zero when Computational Intensity is equal to zero

- Knee point: It is interception between the Computational Intensity and Bandwidth ($y = mx \Rightarrow x = y/m$)

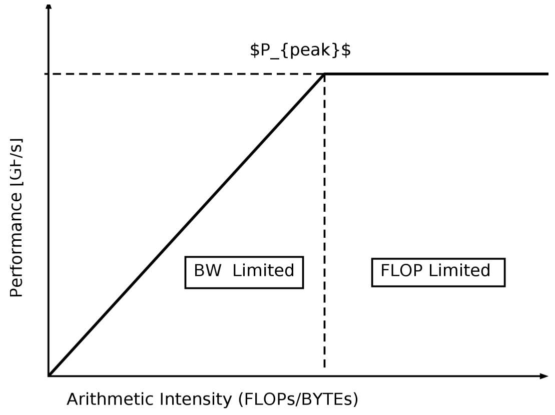
$$\begin{aligned} y_{\text{Attainable Peak Performance}} &= m_{\text{Bandwidth}} \cdot y_{\text{Computational Intensity}} \\ x_{\text{Computational Intensity}} &= \frac{y_{\text{Attainable Peak Performance}}}{m_{\text{Bandwidth}}} \cdot \frac{(GFLOP/s)}{(GBytes/s)} \end{aligned}$$

- BandWidth slope line can be build an array(x, y)

$$\begin{aligned} x &= 0 : x_{\text{Computational Intensity}} \\ y &= m_{\text{Bandwidth}} \cdot x \end{aligned}$$

5.2 Roofline Model

The Roofline model characterizes a kernel's performance in GigaFLOPs per second `(GFLOP/s)` as a function of its Arithmetic Intensity (AI) a ratio of floating-point operations performed to data movement `(FLOP/Byte)`, as described as Equation 5.2



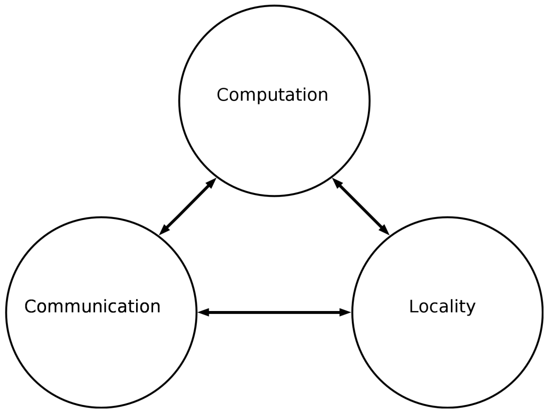
Roofline Performance Model

$$GFLOP/s \leq \min \begin{cases} Peak\ GFLOP/s \\ Peak\ GB/s \times Aritmetic\ Intensity \end{cases}$$

Performance can be estimated from hardware and kernel characteristics

- Kernels can be **compute bounded (DGEMM)** or **communication bounded (DAXPY)** (kernels are rarely well balanced)

Maximum processing capability	Peak performance	$P_{peak}\left[\frac{FLOPs}{s}\right]$
Rate of revolving door	Memory bandwidth	$b_S\left[\frac{Bytes}{s}\right]$
Workload per customer	Arithmetic Intensity (AI) or Computational Intensity	$I\left[\frac{FLOPs}{s}\right]$



Roofline model - Principal Components to Performance

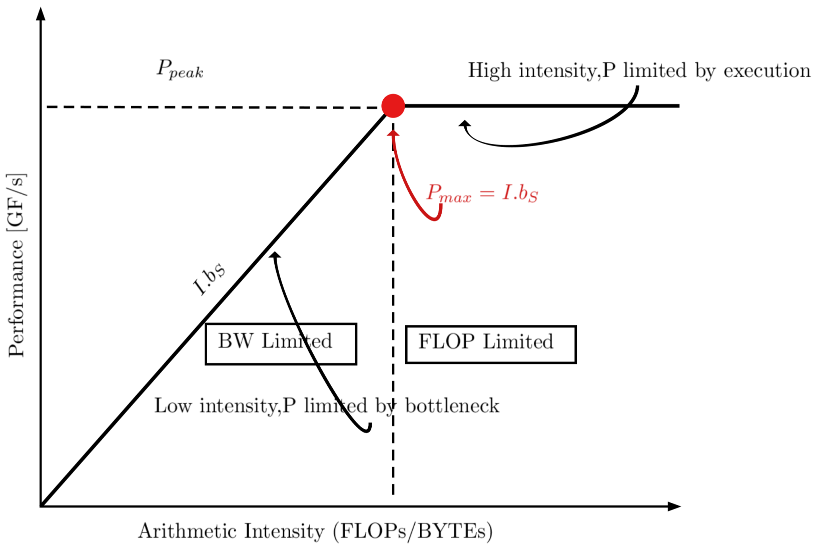
- Mapping kernel characteristics to hardware characteristics (or vice-versa) lead to performance
- Some hardware is more
  - Communication oriented than another (High memory BW)
  - Computation oriented than another (High FLOPs)
- Peak performance
  - High intensity (FLOP Limited): Performance limited by execution
  - Low intensity (BandWidth Limited): Performance limited by bottleneck
  - $P_{max} = I \cdot b_S$ (at the red inflexion point)

The Roofline Model – refined

- P_{max} = Applicable peak performance of a loop, assuming that data comes from the level 1 cache (this is not necessarily P_{peak}), $P_{max} = 176$ GFlop/s.
- I = Computational intensity (work per byte transferred) over the slowest data path utilized (code balance $B_C = I^{-1}$), $I = 0.167$ Flop/Byte, $B_C = 6$ Byte/Flop.
- b_S = Applicable (saturated) peak bandwidth of the slowest data path utilized, $b_S = 56$ GByte/s.

Expected performance:

$$P = \min\left(P_{max}, I \cdot b_S\right) = \min\left(P_{max}, \frac{b_S}{B_C}\right)$$


P_{peak}
High intensity,P limited by execution
$P_{max} = I.b_S$
$I.b_S$
BW Limited
FLOP Limited
Low intensity,P limited by bottleneck
Performance [GF/s]
Arithmetic Intensity (FLOPs/BYTEs)

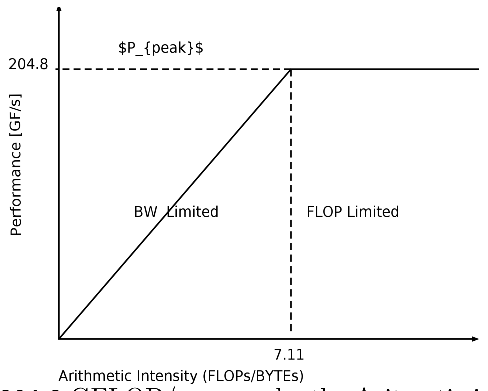
Roofline model - Performance Upper Bounded

6 Examples

1. Consider DAXPY

```
for (i = 0; i < N; ++i) {
    y[i] = a*x[i]+y[i];
}
```

Execution on BlueGene/Q (Peak 204.8 GFLOP/node).

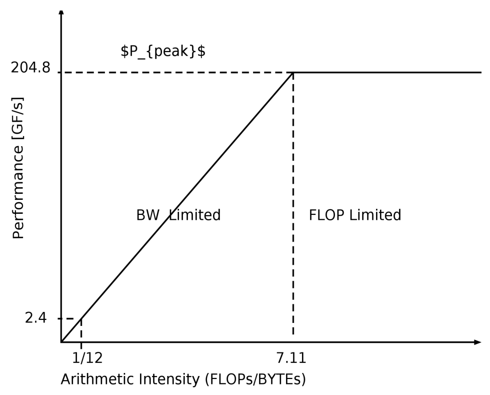


Solution : From the graps, the Peak Performance (P_{max}) is 204.8 GFLOP/s per node, the Aritmetic intensity (AI) is 7.11 FLOPs/BYTEs, and from the loop the Aritmetic Intensity (AI) can be estimated.

LOOPS	FLOPS	BYTES	AI $= \frac{FLOPS}{BYTES}$
for (i = 0; i <N; ++i){ y[i] = a*x[i]+y[i] }	1 add 1 mult	2(8 bytes) loads 1(8 bytes) write (or data transfer)	$\frac{2}{2*8+1*8} = \frac{1}{12}$

Then, the loop Peak performance

$$\begin{array}{lccc} \text{Arithmetic Intensity (AI)} & \frac{1}{12} & \longleftrightarrow & 7.11 \\ \text{Peak performance } (P_{peak}) & x & \longleftrightarrow & 204.8 \end{array}$$

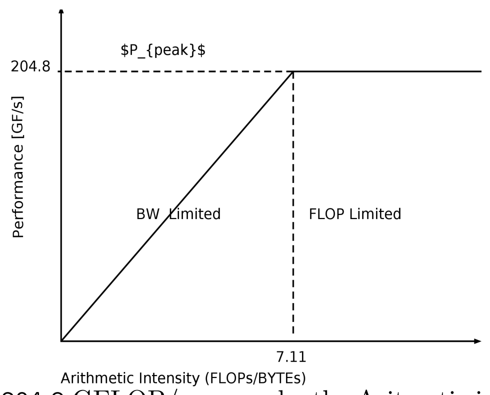


The loop AI is in the memory bandwidth (BW) limited area

$$x = \frac{(204.8)\ (1/12)}{(7.11)} = 2.4\ \frac{GF}{s}$$

.

2. Consider DAXPY



```
for (i = 0; i < N; ++i) {
    y[i] = a*x[i]+y[i]+x[i]*x[i];
}
```

Execution on BlueGene/Q (Peak 204.8 GFLOP/node).

Solution : From the graps, the Peak Performance (P_{max}) is 204.8 GFLOP/s per node, the Aritmetic intensity (AI) is 7.11 FLOPs/BYTEs, and from the loop the Aritmetic Intensity (AI) can be estimated. Performance

LOOPS	FLOPS	BYTES	AI $= \frac{FLOPS}{BYTES}$
for (i = 0; i <N; ++i){ y[i] = a*x[i]+y[i]+x[i]*x[i] }	2 add 2 mult	2(8 bytes) loads 1(8 bytes) write (or data transfer)	$\frac{4}{2*8+1*8} = \frac{1}{6}$

estimates

Arithmetic Intensity	$\frac{1}{6}$	$\longleftrightarrow$	7.11
Peak performance	x	$\longleftrightarrow$	204.8

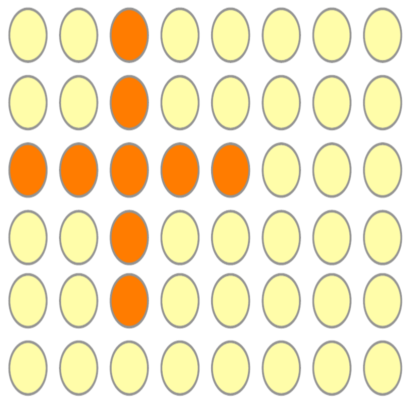
The loop AI is in the memory bandwidth (BW) limited area

$$x = \frac{(204.8)\ (1/6)}{(7.11)} = 4.8\ \frac{GF}{s}$$

3. Consider two arrays A, and B, both have dimension of $N \times N$

```
for (i = 0; i < N; ++i) {
    for (j = 0; j < N; ++j) {
        B[i][j] = A[i-2][j] + A[i-1][j] + C*A[i][j] +
                  A[i+1][j] + A[i+2][j] + A[i][j-2] +
                  A[i][j-1] + A[i][j+1] + A[i][j+2];
}
```

Execution on BlueGene/Q (Peak 204.8 GFLOP/node).



Solution : From the graps, the Peak Performance (P_{max}) is 204.8 GFLOP/s per node, the Aritmetic intensity (AI) is 7.11 FLOPs/BYTEs, and from the loop the Aritmetic Intensity (AI) can be estimated.

LOOPS	FLOPS	BYTES	AI $= \frac{FLOPS}{BYTES}$

for (i = 0; i <N; ++i){
for (j = 0; j <N; ++j){
B[i][j] = A[i-2][j]+A[i-1][j]+A[i][j]
A[i+1][j]+A[i+2][j]+A[i][j-2]
A[i][j-1]+A[i][j+1]+A[i][j+2]
}
	8 add
	1 mult
		1(8 bytes) loads
		1(8 bytes) write (or data transfer)
			$\frac{9}{1*8+1*8} = \frac{1}{2}$

Performance estimates

Arithmetic Intensity	$\frac{1}{2}$	$\longleftrightarrow$	7.11
Peak performance	x	$\longleftrightarrow$	204.8

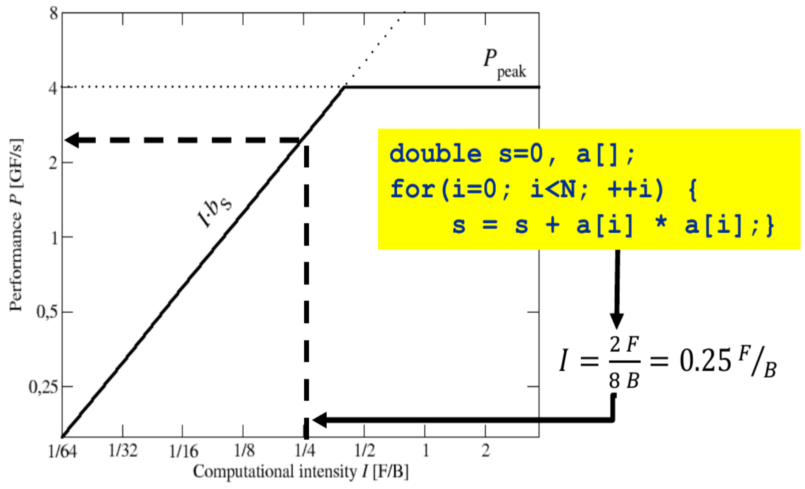
The loop AI is in the memory bandwidth (BW) limited area

$$x = \frac{(204.8)\ (1/2)}{(7.11)} = 14.4\ \frac{GF}{s}$$

4. Apply this example to performance of compute devices

```
double s = 0;
double a [];
for  (i = 0; i < N; ++i) {
    s = s + a[i]  * a[i];
}
```

$$P_{peak} = 4\frac{GFLOPs}{s}$$
$$b_S = 10\frac{GBytes}{s}$$



```
double s=0, a[];
for(i=0; i<N; ++i) {
    s = s + a[i] * a[i];}
```

$$I = \frac{2\,F}{8\,B} = 0.25\ {}^{F}/_{B}$$

LOOPS	FLOPS	BYTES	AI $= \frac{FLOPS}{BYTES}$

double s = 0;
double a[];
for (i = 0; i <N; ++i){
s = s + a[i]* a[i]
}
	1 add
	1 mult
		1(8 bytes) loads	$\frac{2}{1*8} = \frac{2}{8}$

5. Estimate P_{max} of vector triad on Haswell

```
double *A, *B, *C, *D;
for  (int  i=0; i<N; i++) {
  A[i]  = B[i]  + C[i]  * D[i];
}
```

$$P_{peak} = 4\frac{GFLOPs}{s}$$
$$b_S = 10\frac{GBytes}{s}$$

LOOPS	FLOPS	BYTES	AI = $\frac{FLOPS}{BYTES}$

double $^*A$, $^*B$, $^*C$, $^*D$;
for (i = 0; i <N; ++i){
A[i] = B[i] + C[i] * D[i]
}
	1 add
	1 mult
		3(8 bytes) loads
		1(8 bytes) write (or data transfer)
			$\frac{2}{3*8+1*8} = \frac{2}{32}$

Minimum number of cycles to process one AVX-vectorized iteration (one core)?

Equivalent to 4 scalar iterations

Cycle 1: LOAD + LOAD + STORE

Cycle 2: LOAD + LOAD + **FMA** + FMA

Cycle 3: LOAD + LOAD + STORE

Answer: 1.5 cycles

What is the performance in GFlops/s per core and the bandwidth in GBytes/s?

One AVX iteration (1.5 cycles) does $4 \times 2 = 8$ flops:

$$\frac{2.3 \times 10^9\ cy/s}{1.5\ cy} \times 4\ updates\ \frac{2\ flops}{update} = 12.27\ \frac{Gflops}{s}$$

$$6.13 \times 10^9\ \frac{updates}{s} \times 32\ \frac{bytes}{update} = 196\ \frac{Gbyte}{s}$$

Vector triad $A(:) = B(:) + C(:) * D(:)$ on a 2.3 GHz 14-core Haswell chip (AVX vectorized)

- Peak memory bandwidth: $b_S = 50\ \frac{GB}{s}$
- Memory code balance:
  - $B_c = \frac{(4+1)\ Words}{2\ Flops} = 20\ \frac{B}{F}$ (including write allocate)
  - $I = 0.05\ \frac{F}{B}$
  - $I \cdot b_S = 0.05\ \frac{F}{B} \times 50\ \frac{GB}{s} = 2.5\ \frac{GF}{s}$ (0.5% of peak performance)

- $P_{peak} = 515.2\ \frac{Gflop}{s}$ $(14\ cores \times (8+8)\frac{Flops}{cy} \times 2.3\ GHz)$
- $P_{max} = 14 \times 12.27\ \frac{Gflop}{s} = 172\ \frac{Gflop}{s}$ (33% peak)

$$P = \min\left(P_{max}, I \cdot b_S\right) = \min\left(172, 2.5\right)\ GFlop/s = 2.5\ GFlop/s \tag{1}$$

6. NVIDIA's latest V100 GPU :

- Each GV100 Streaming Multiprocessor (SM) consists of four processing blocks (warp schedulers)
- Each warp scheduler can dispatch one instruction per cycle.

As such, the theoretical maximum (warp-based) `instruction/s` is

$$80(\text{SM}) \times 4(\text{warp scheduler}) times 1(\text{instruction/cycle}) \times 1.53(\text{GHz}) = 489.6 GIPS \tag{2}$$

- Memory access is coalesced into transactions.
  - The transaction size for global/local memory, the L2 cache, and HBM are 32 bytes.
  - The shared memory transaction size is 128 bytes.

  - In practice, a warp-level load may generate anywhere from 1 to 32 transactions depending on memory patterns.
- This makes the transaction the natural unit when analyzing memory access.
- We leverage Yang et al.'s methodology for measuring GPU bandwidths but rescale into billions of transactions per second (GTXN/s) based on the transaction size.

shows the resultant Instruction Roofline ceilings for the GV100. L1, L2, and HBM bandwidths of 14000, 2996, and 828 GB/s sustain 437, 93.6, and 25.9 GTXN/s when normalized to the typical 32-byte transaction size.

$$GIPS \leq \min \begin{cases} Peak\ GIPS \\ Peak\ GT \times N/s \times Instruction\ Intensity \end{cases} \tag{3}$$

7. (NERSC-Cori Haswell/KNL) Xeon Phi 7250 (HLRN Cray TDS), advertised with 3.05 TFLOPS peak DP (peak double precision performance).Intel Xeon Phi Processor 7250 68C
   - Processor speed (Clock rate) : 1.4GHz
   - Architecture : x86-64
   - Physical Cores Per Node : 68
   - Theoretical peak : 3 TFlops/node
   - 8 SIMD (Single Instruction/Multiple Data)
   - 2 FMS (Fused Multiply-Add instructions) is an extension to the 128 and 256-bit Streaming SIMD Extensions instructions in the x86 microprocessor instruction set to perform fused multiply–add (FMA) operations.
   - VPUs : Vector processing units (VPUs) perform the actual work of computing a SIMD vector operation in parallel.

```
1.4 GHz x 68 core x 8 SIMD x 2 VPUs x 2 FMA = 3046.4 GFLOPS
```

```
AVX frequency is only 1.2 GHz and might throttle down under heavy load => actual peak: 2611.2 GFLOPS
```

References

[1] N. Ding and S. Williams. An instruction roofline model for gpus. In *2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pages 7–18, 2019.