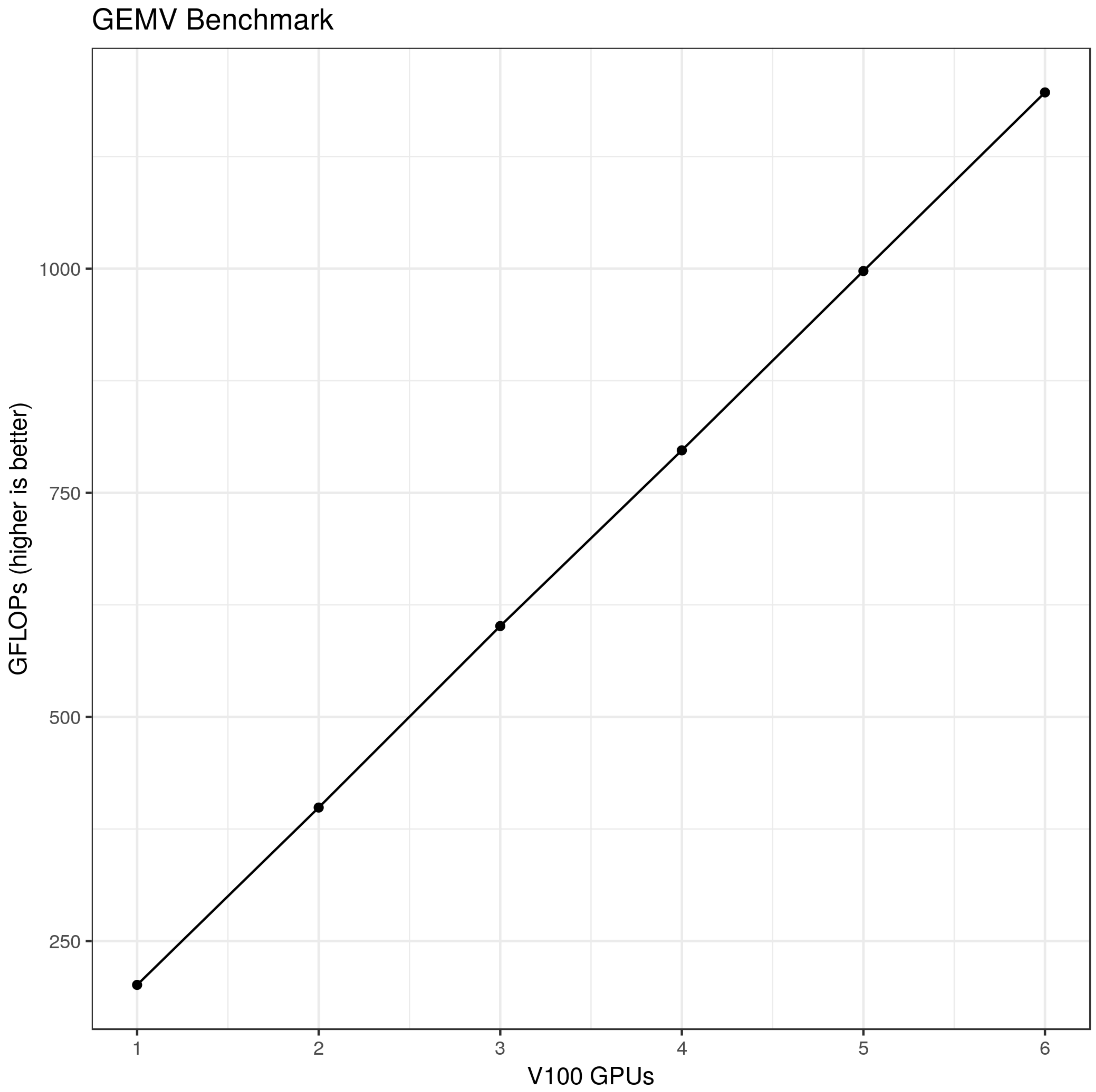
Model Driven Code Optimization - I: Roofline Model

Piyush Sao
Computer Science & Mathematics Division, ORNL
With inputs from Rich Vuduc (GT), Jee Choi (UOregon), Aparna Chandramowlisharan (UCI)
Sam Williams (LBNL)

Oak Ridge National Laboratory

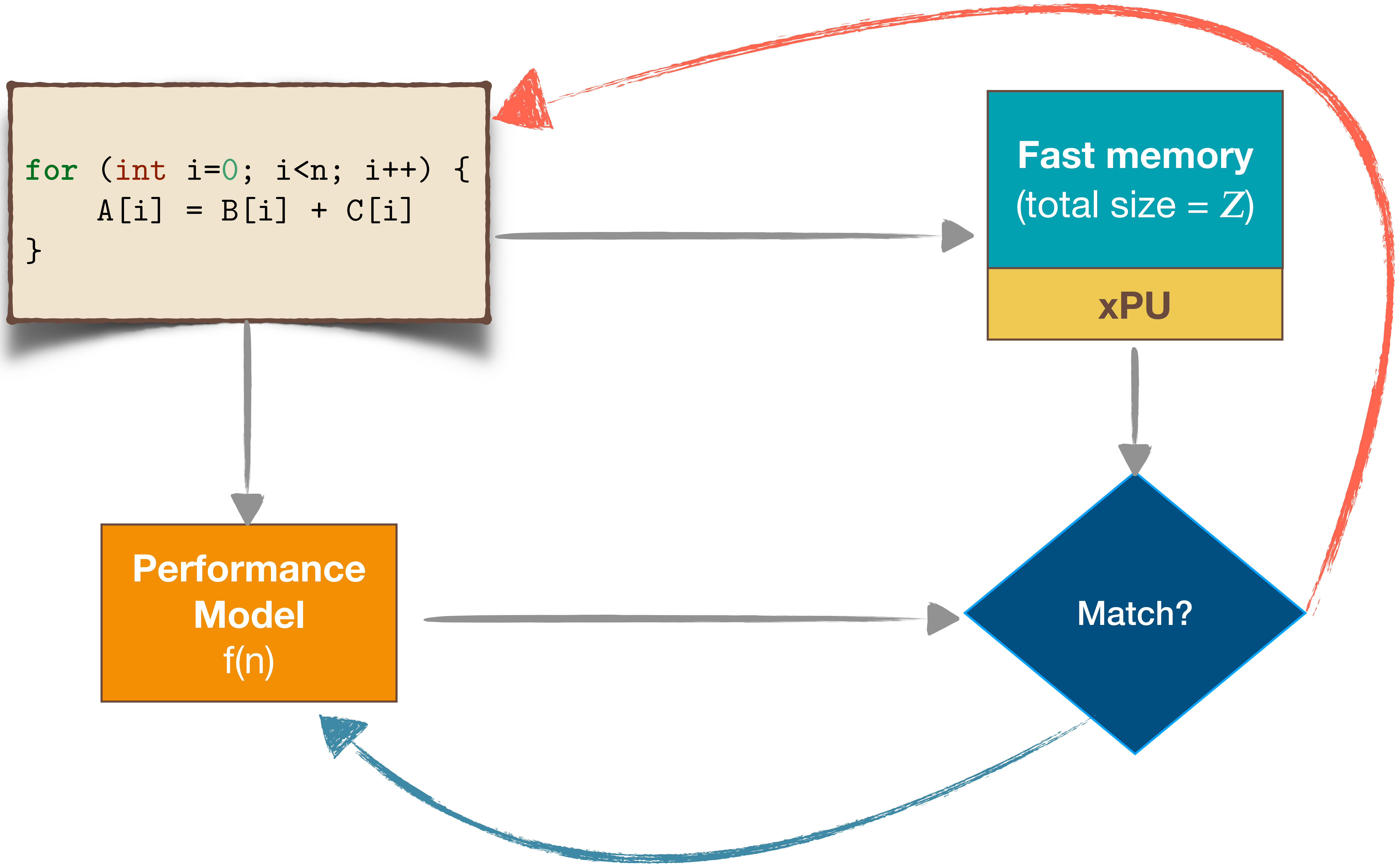


GeMV on a Summit Node

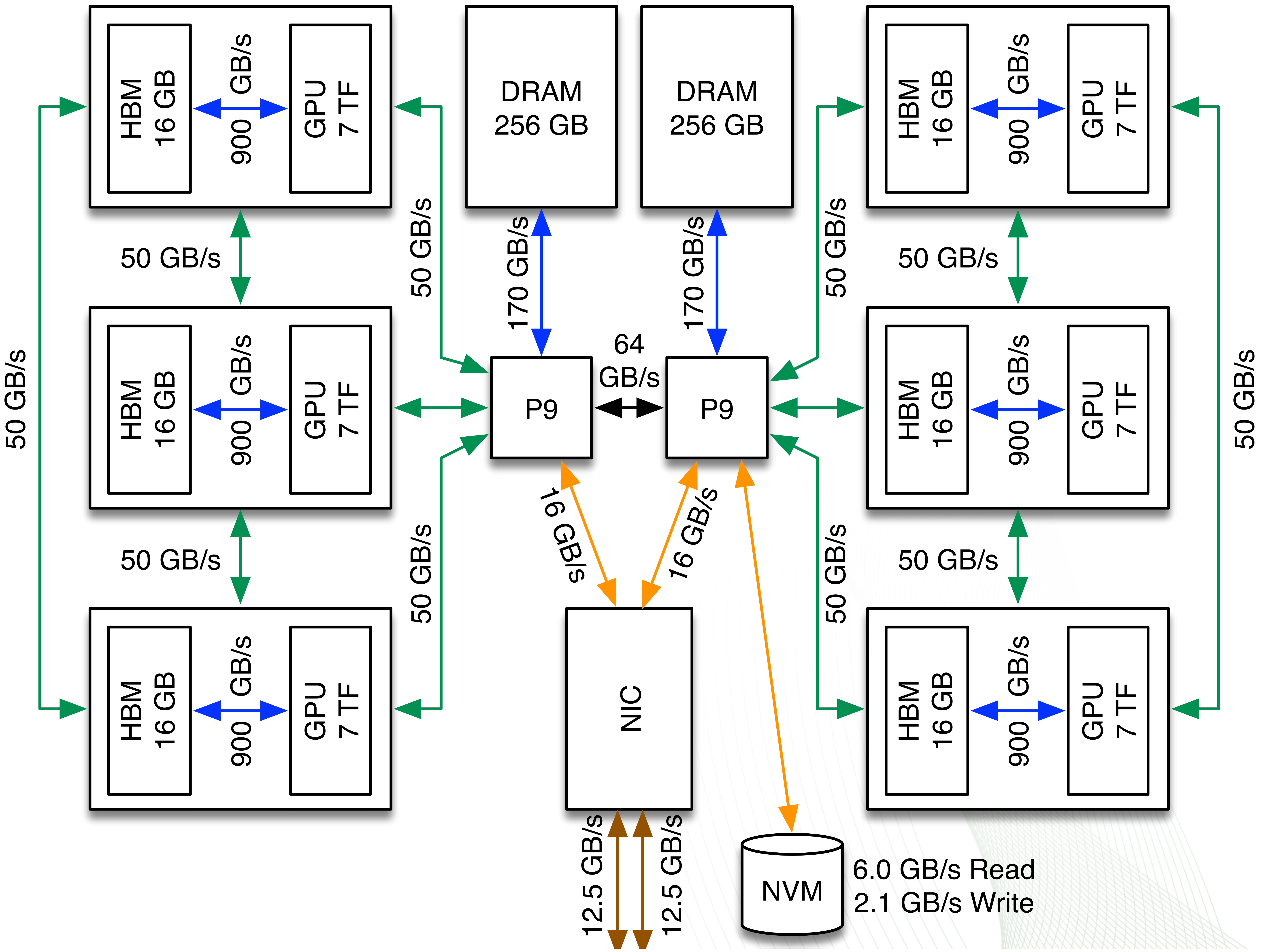
- Double Precision Peak Per V100 is **7 TF/sec**
- But my friend is seeing only *200 GF/sec* GPU?

```
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
    C[i] += A[i][j]*B[j]
```

How do I make my code run faster?



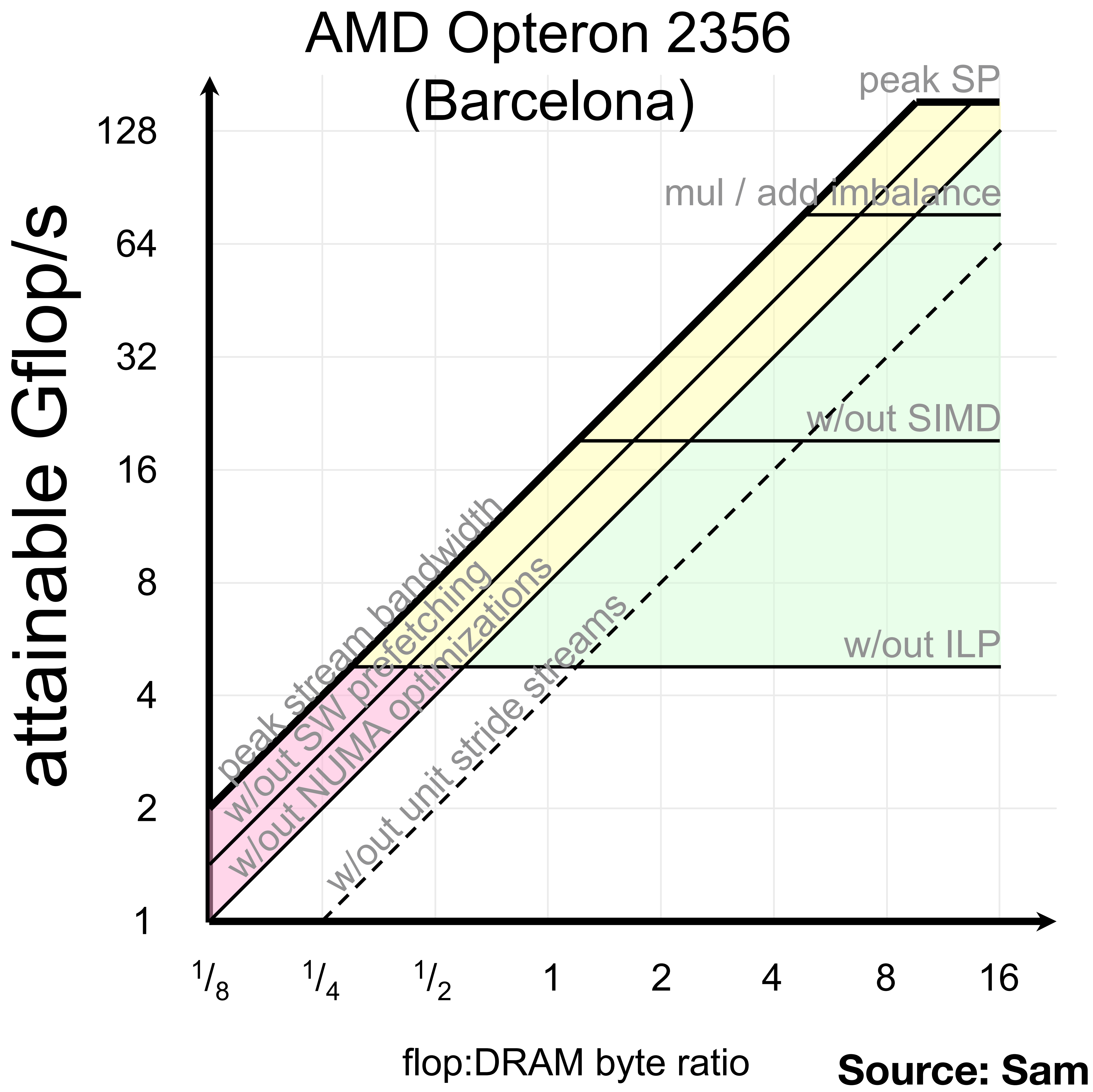
Performance Modeling is Hard



Fast Code

- Mapping code to architecture
- Fat nodes = too many architectural features and parameters

Roofline Model



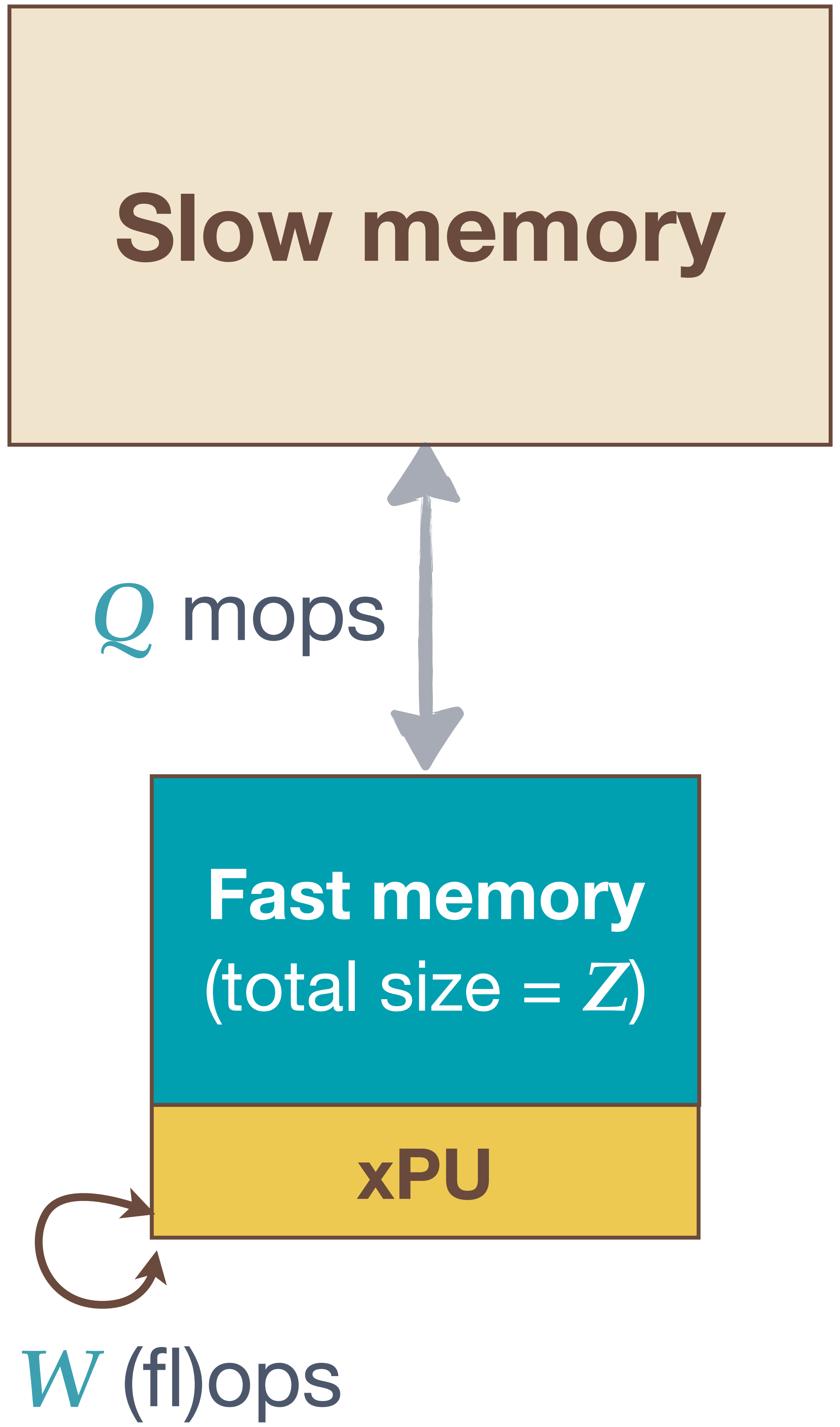
Source: Sam Williams (LBNL)

Goals Of Roofline Model

- A graphical aid that provides : realistic expectations of performance and productivity
- Show inherent hardware limitations for a given kernel
- Show potential benefit and priority of optimizations
- Focused on: rates and efficiencies (GF/s, % of peak)

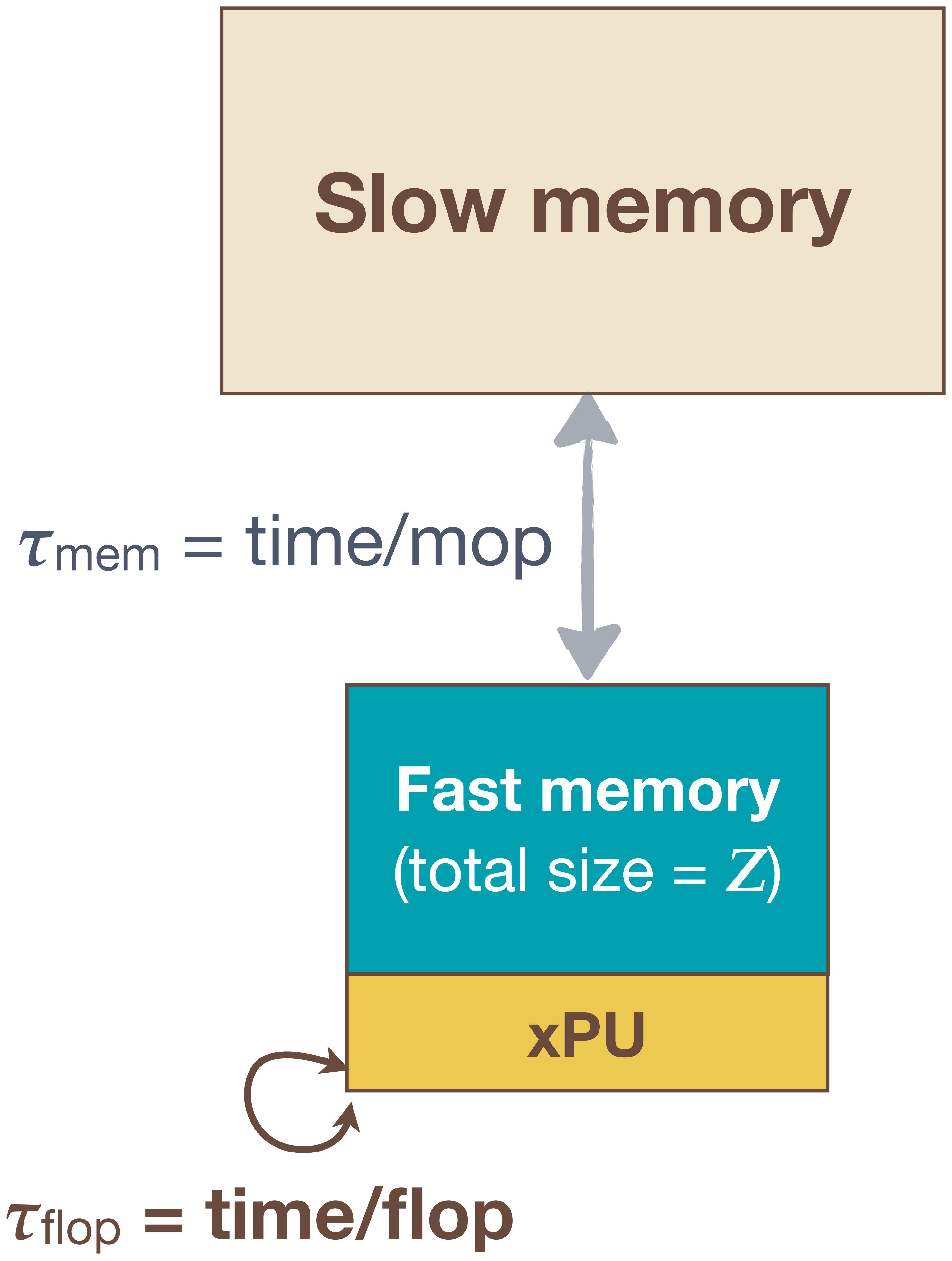
Three Principle Components

- Computation: measured in GF/sec, GTEPS
- Communication: GB/sec
- Locality: Cache Size, Data Reuse



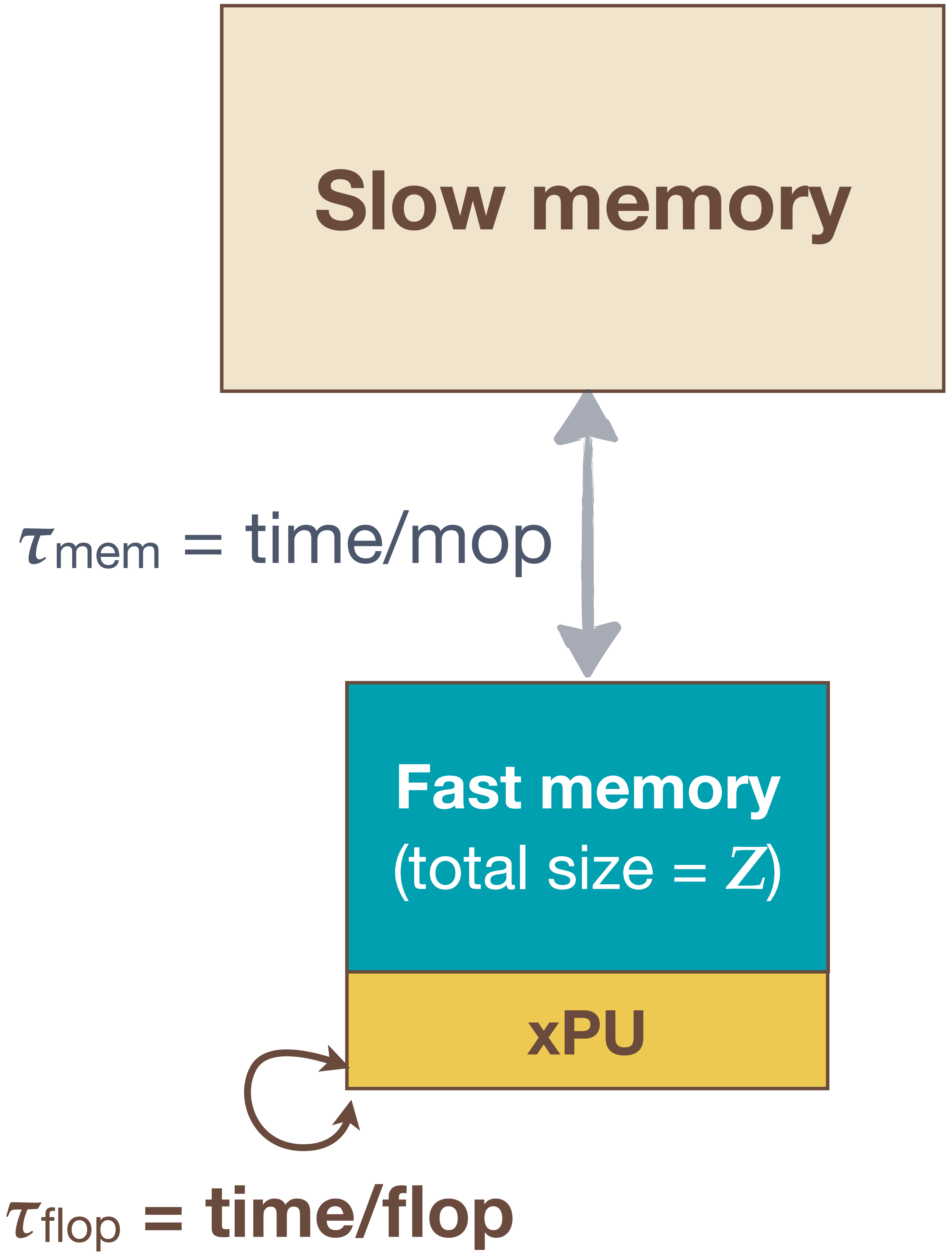
von Neumann bottleneck

$$
\begin{aligned}
W &\equiv \#\text{ (fl)ops} \\
Q &\equiv \#\text{ mem. ops (mops)} \\
&= Q(Z)
\end{aligned}
$$



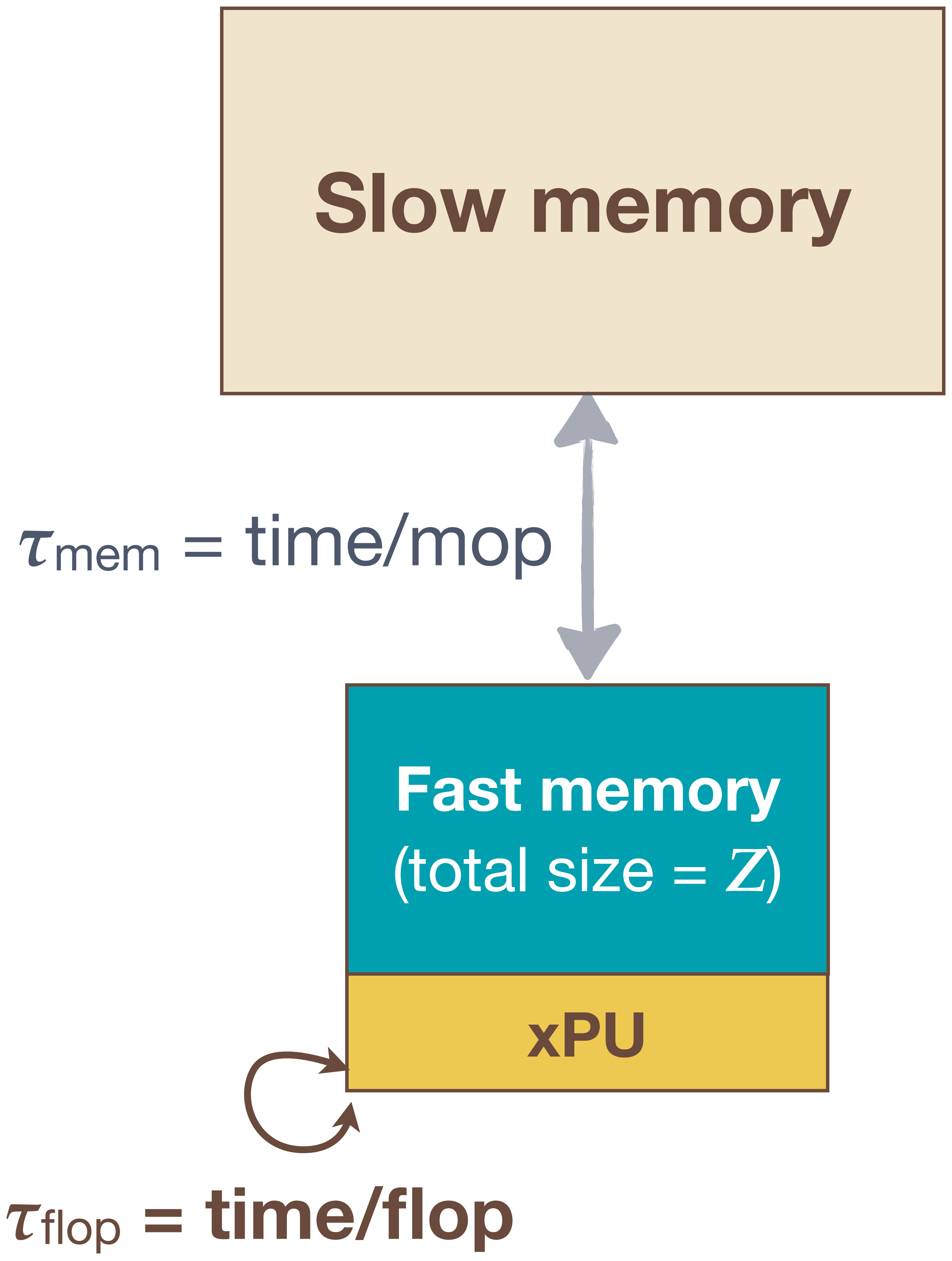
$$T = \max\left(W\tau_{\text{flop}}, Q\tau_{\text{mem}}\right)$$

von Neumann bottleneck



$$
\begin{aligned}
T &= \max\left(W\tau_{\text{flop}}, Q\tau_{\text{mem}}\right) \\
&= W\tau_{\text{flop}} \max\left(1, \frac{Q}{W}\frac{\tau_{\text{mem}}}{\tau_{\text{flop}}}\right)
\end{aligned}
$$

von Neumann bottleneck

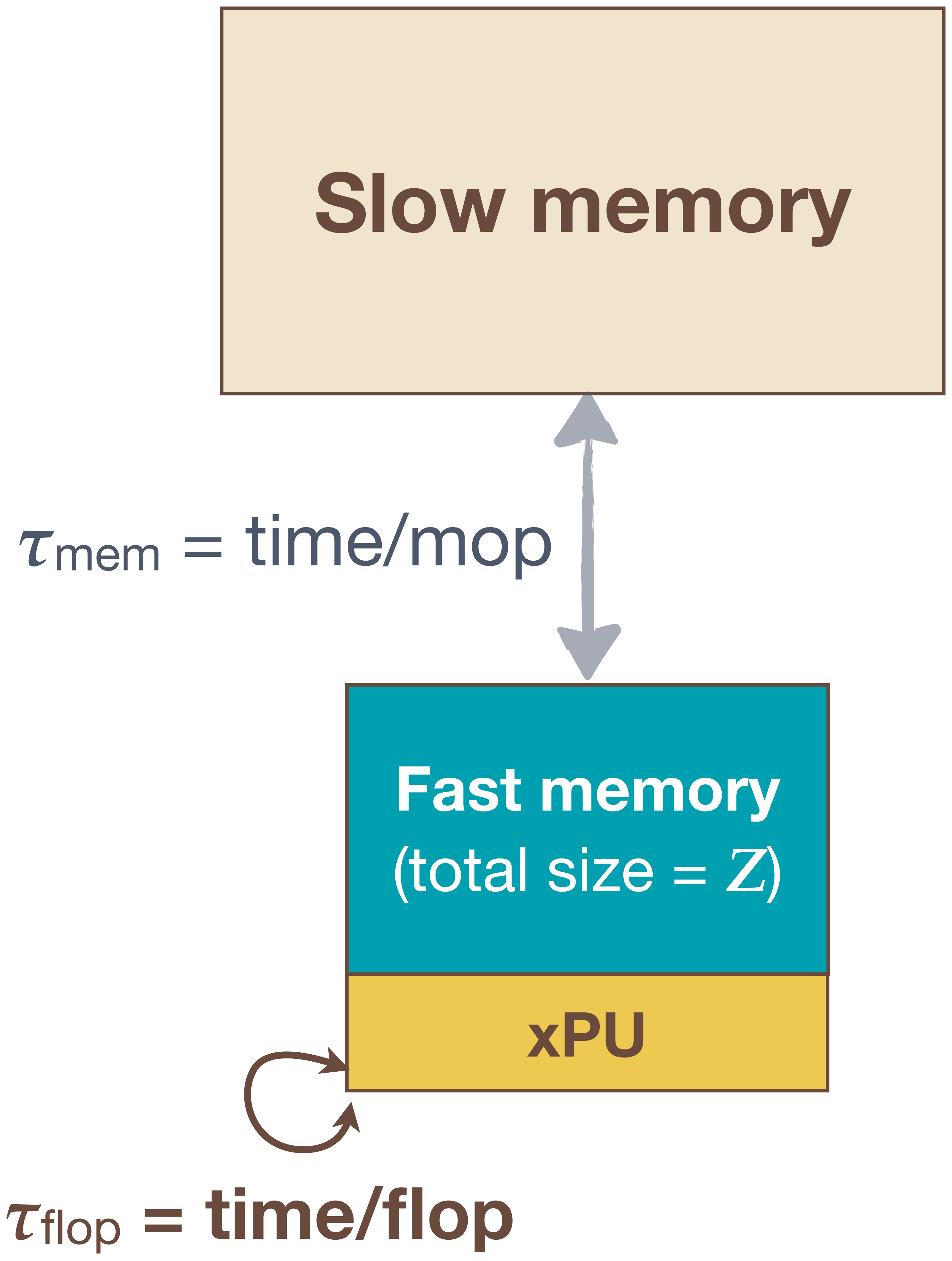


von Neumann bottleneck

$$
\begin{aligned}
T &= \max\left(W\tau_{\text{flop}}, Q\tau_{\text{mem}}\right) \\
&= W\tau_{\text{flop}} \max\left(1, \frac{Q}{W}\,\frac{\tau_{\text{mem}}}{\tau_{\text{flop}}}\right) \\
&= W\tau_{\text{flop}} \max\left(1, \frac{B_\tau}{I}\right)
\end{aligned}
$$

Intensity
(flop : mop)

Balance
(flop : mop)



von Neumann bottleneck

$$
\begin{aligned}
P &= \frac{W}{T} \\
&= \frac{1}{\tau_{fl}} \min\left(\frac{I}{B_\tau}, 1\right) \\
&= P_\infty \min\left(\frac{I}{B_\tau}, 1\right)
\end{aligned}
$$

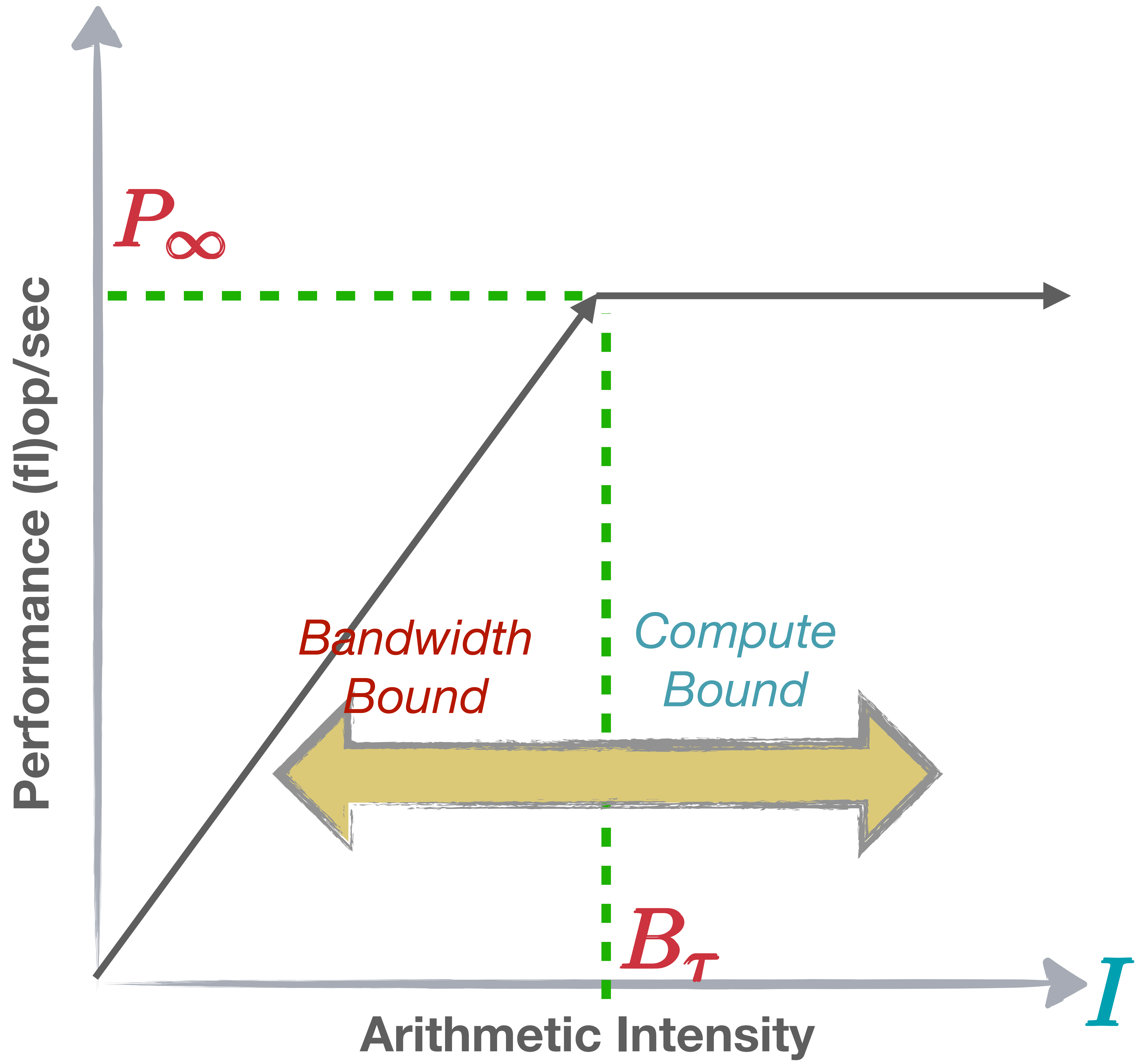
$P \equiv$ Performance (FLOP/sec)

$P_\infty \equiv$ Theoretical Peak

$B_\tau \equiv$ Machine Balance

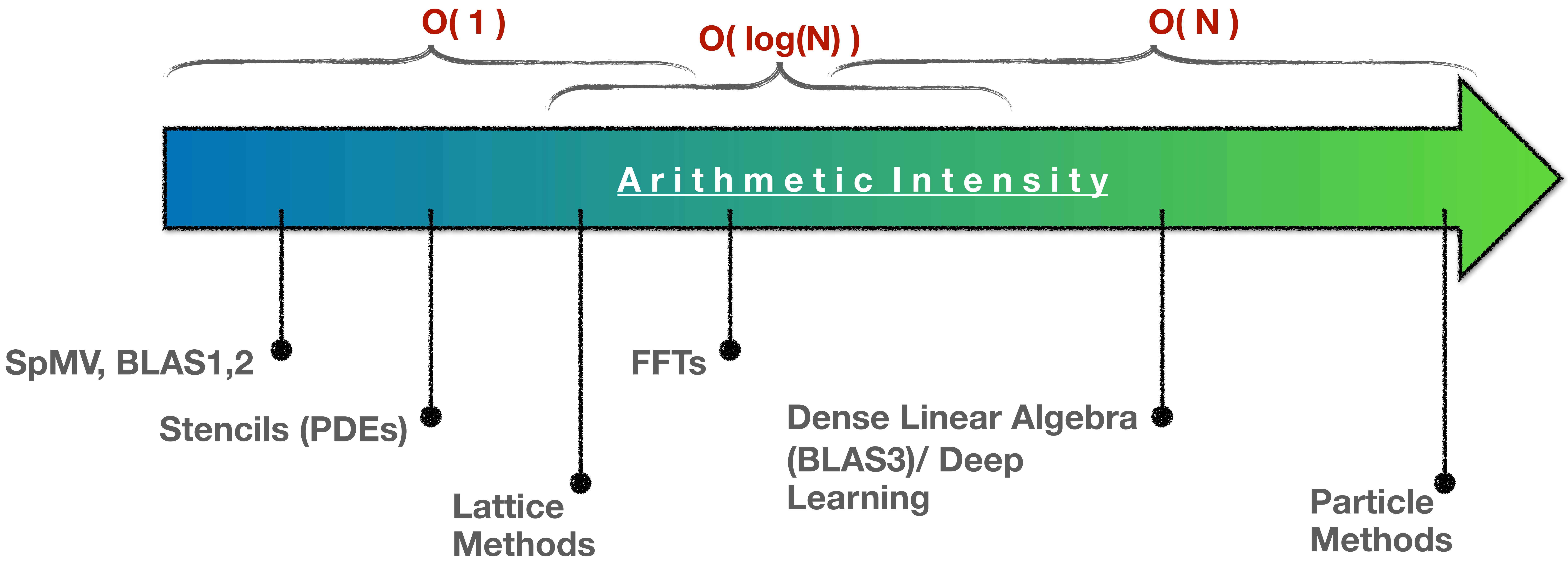
$I \equiv$ Arithetic Intensity

Roofline Curve



$$
\begin{aligned}
P &= \frac{W}{T} \\
&= \frac{1}{\tau_{fl}} \min\left(\frac{I}{B_\tau},\ 1\right) \\
&= P_\infty \min\left(\frac{I}{B_\tau},\ 1\right) \\
&= \min\left(\beta I,\ P_\infty\right)
\end{aligned}
$$

$\beta \equiv \dfrac{1}{\tau_{mem}}$ Data Transfer Bandwidth



❖ **True Arithmetic Intensity (AI) ~ Total Flops / Total DRAM Bytes**

- constant with respect to problem size for many problems of interest
- ultimately limited by compulsory traffic
- diminished by conflict or capacity misses.

```
for (int i=0; i<n; i++) {
    A[i] = B[i] + C[i]
```

$$W = n$$
$$Q = 24n$$
$$I = \frac{1}{24}$$

```
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
    C[i] += A[i][j]*B[j]
```

$$W = 2n^2$$
$$Q = 8n^2 + 16n$$
$$I = \frac{1}{4}$$

```
for (int k=0; k<n; k++) {
    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
        C[i][j] += A[i][k]*B[k][j]
```

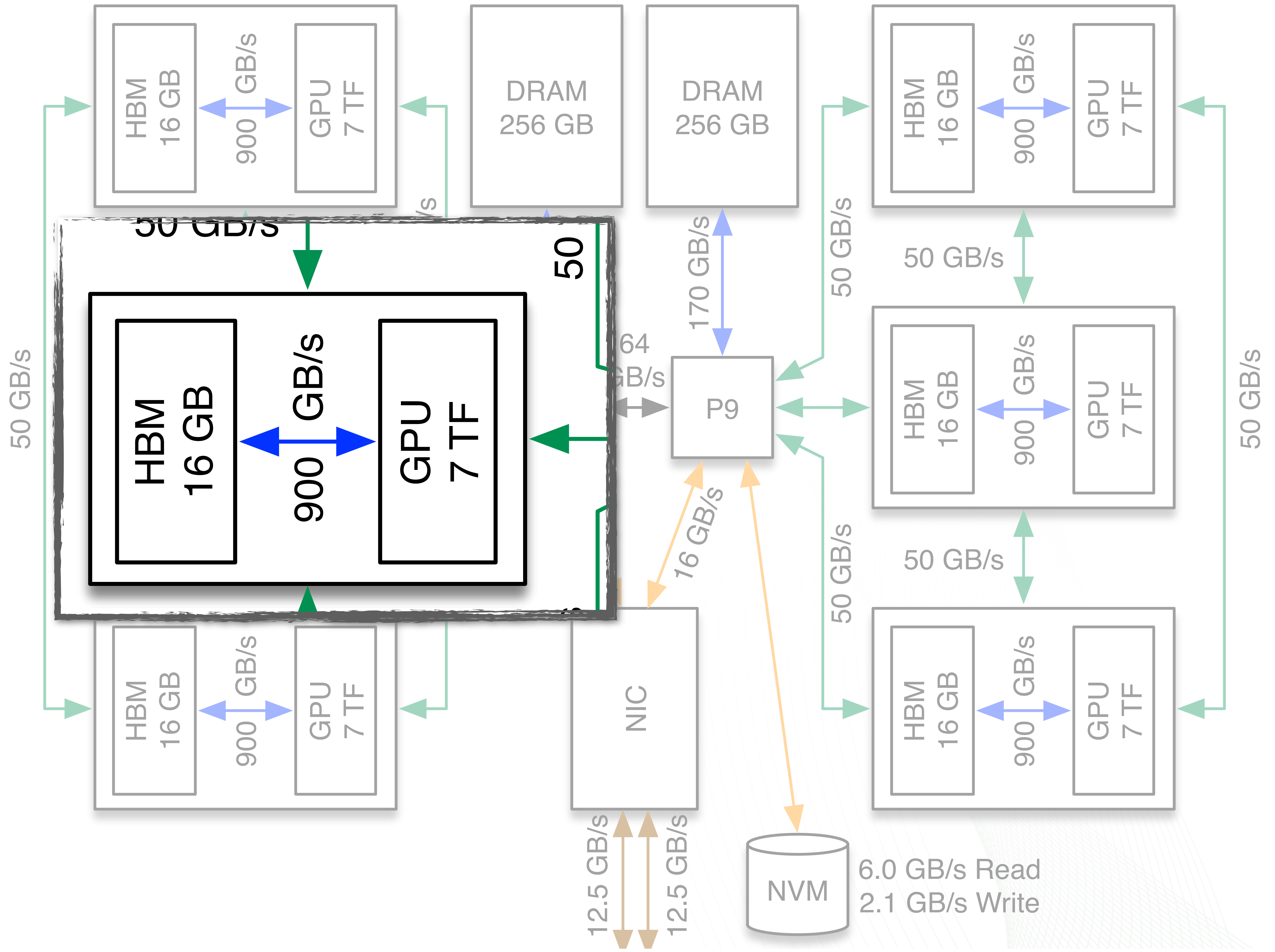
$$W = 2n^3$$
$$Q = 24n^2$$
$$I = \frac{n}{2}$$

```
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
    C[i] += A[i][j]*B[j]
```

$$W = 2n^2$$

$$Q = 8n^2 + 16n$$

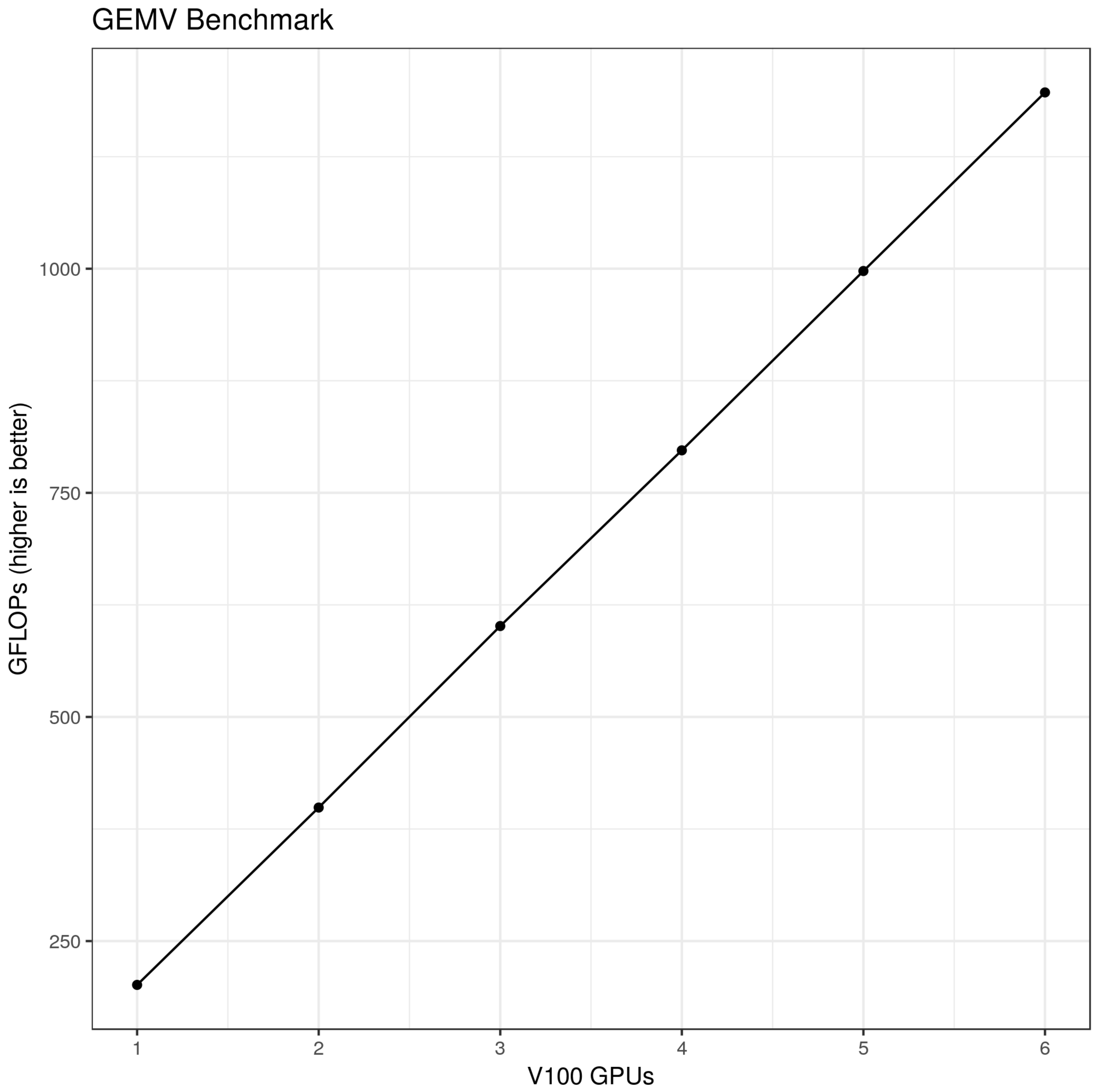
$$I = \frac{1}{4}$$



β = 900 GB/sec

P_∞ = 7000 GF/sec

P = min(900/4, 7000)
= 225GF/sec



GeMV on a Summit Node

- Double Precision Peak Per V100 is **7 TF/sec**
- But my friend is seeing only *200 GF/sec* GPU?

$$W = 2n^2$$

$$Q = 8n^2 + 16n$$

$$I = \frac{1}{4}$$

β **= 900 GB/sec**

P_∞ **= 7000 GF/sec**

P = min(900/4, 7000)
= 225GF/sec

```
for (int i=0; i<n; i++) {
    A[i] = B[i] + C[i]
```

$$I = \frac{1}{24} \qquad P = \min\left(\frac{900}{24}, 3500\right) = 37.4\text{GF/sec}$$

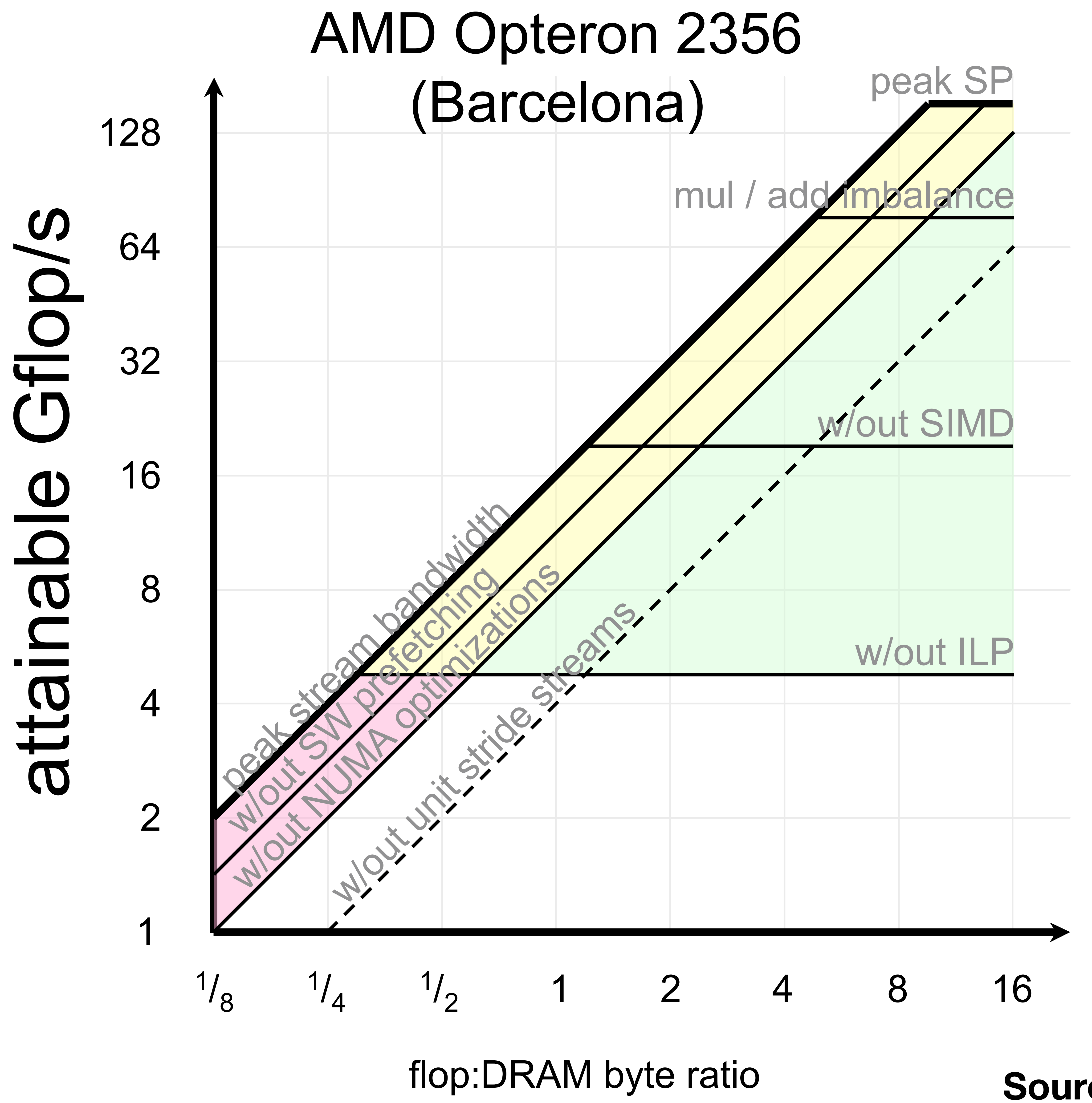
```
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
    C[i] += A[i][j]*B[j]
```

$$I = \frac{1}{4} \qquad P = \min\left(\frac{900}{4}, 7000\right) = 225\text{GF/sec}$$

```
for (int k=0; k<n; k++) {
    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
        C[i][j] += A[i][k]*B[k][j]
```

$$I = \frac{n}{2} \qquad P = \min\left(450n, 7000\right) = 7000\text{GF/sec}$$

Computation:

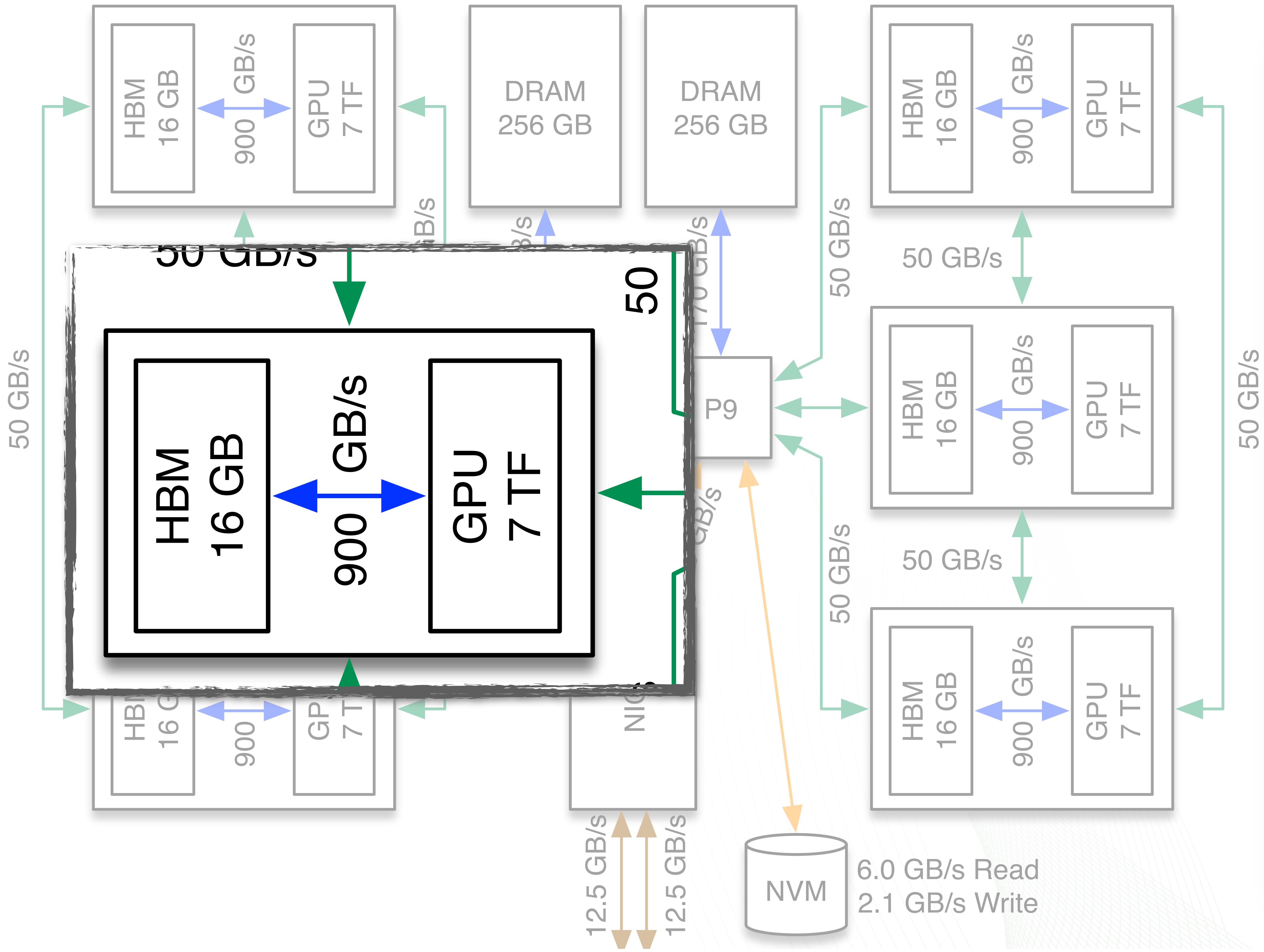


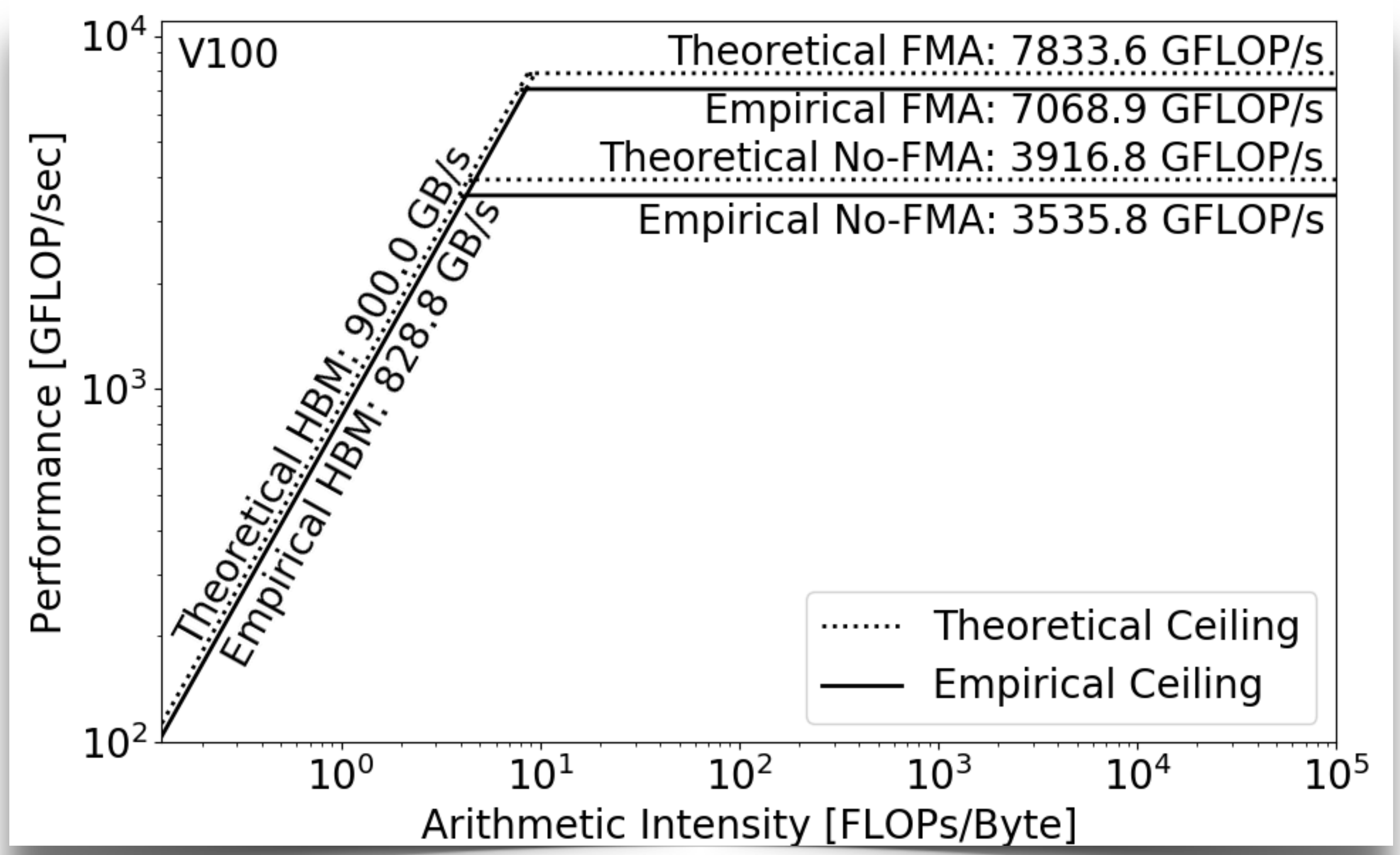
Estimating Peak Attainable Flop Rate

- Can you use FMA (Fused Multiply and Add): Common in Linear Algebra, Deep Learning
- Is the Code Vectorizable?
- Are you exposing enough instruction level parallelism?

Source: Sam Williams (LBNL)

Refining Roofline using: Microbenchmarks





Semi-ring Matrix Multiplication

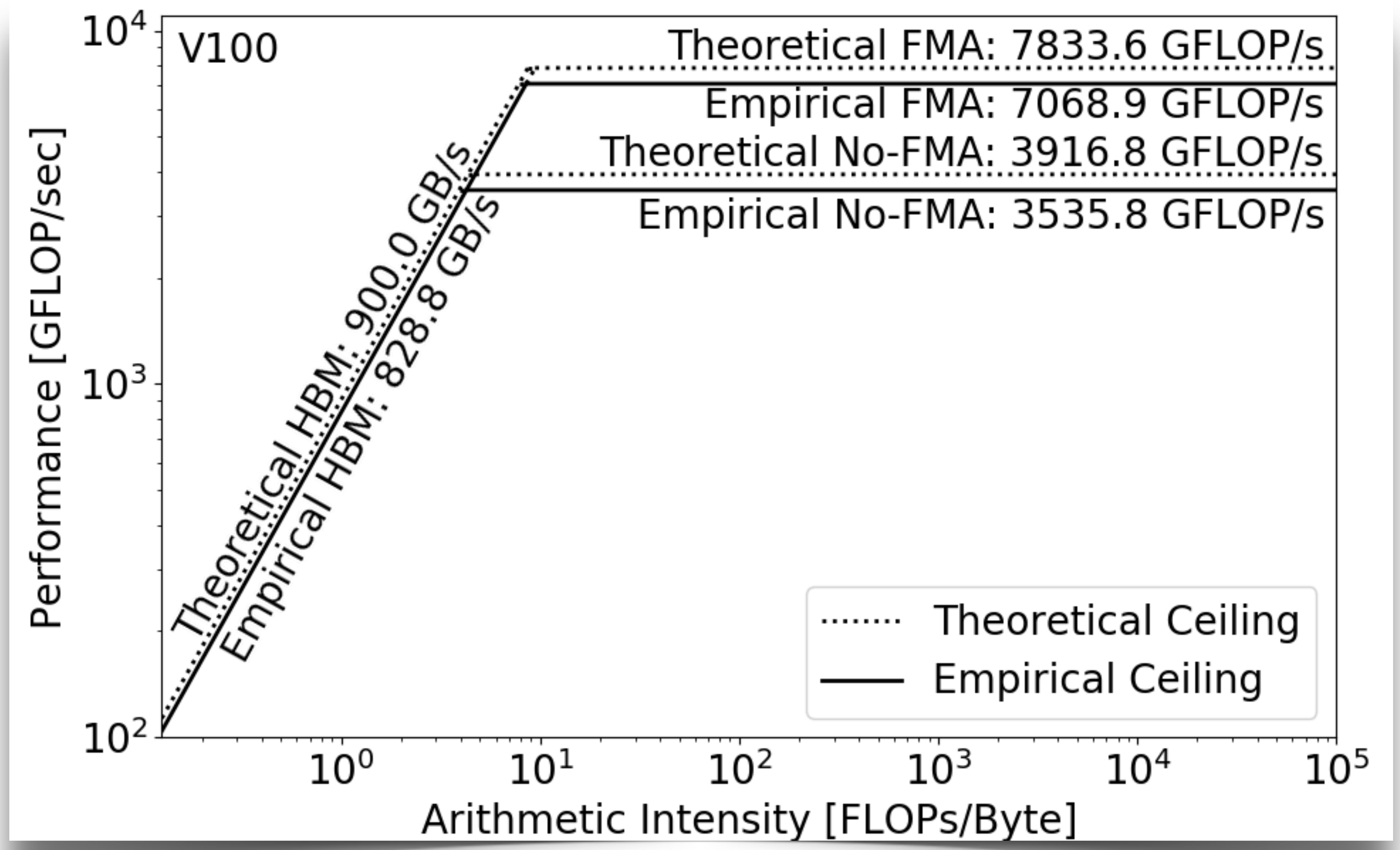
$$C \leftarrow C \oplus A \times B$$

```
for (int k=0; k<n; k++)
    for (int i=0; i<n; i++)
        for (int j=0; j<n; j++)
            C[i][j] = min(C[i][j], A[i][k]+B[k][j])
```

$$W = 2n^3$$

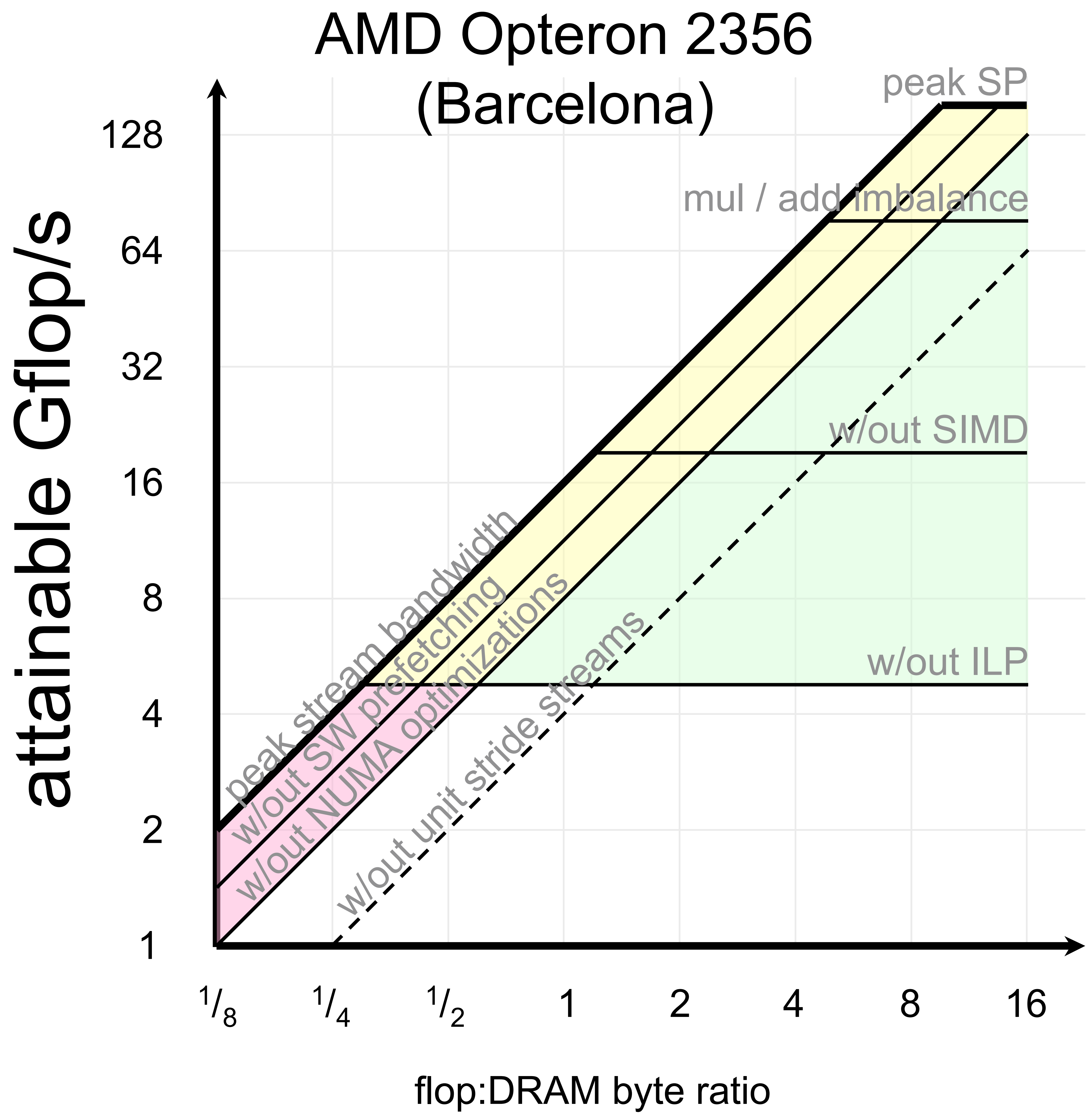
$$Q = 24n^2$$

$$I = \frac{n}{2}$$



Q. Given the roofline curves for V100, how much performance should I expect from semi-ring matrix multiplication kernel!?

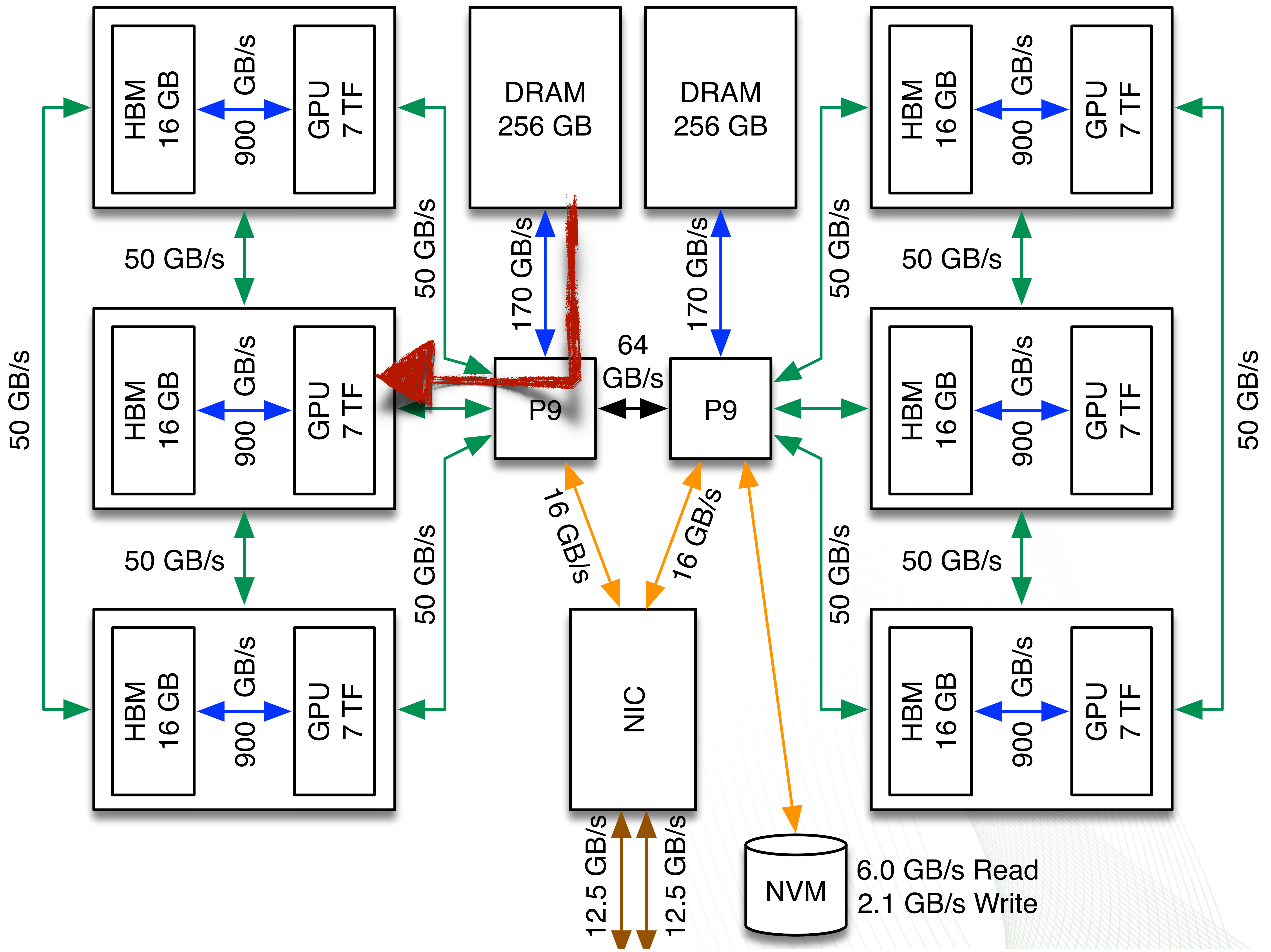
Communication:



Optimizing Communication Bound Kernels

- Low level optimization: prefetching, NUMA optimizations, coalescing memory accesses
- Compression/encoding to reduce metadata
- Exploiting total available bandwidth

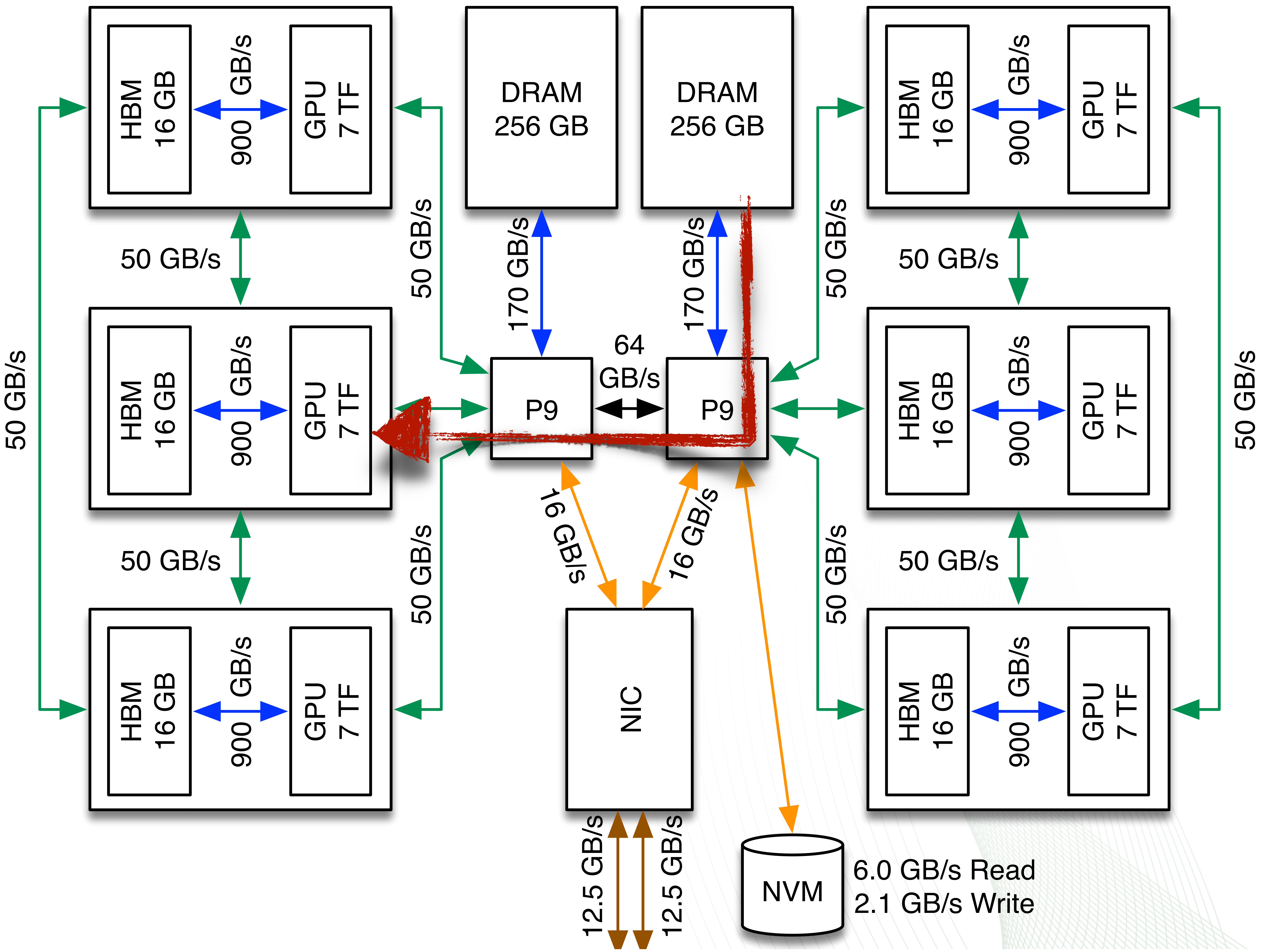
Exploiting Total Available Bandwidth



Q. What is effective **β** when we are streaming data from DRAM to the GPU

A. 170 GB/sec
B. 50 GB/sec
C. 900 GB/sec
D. 150 GB/sec

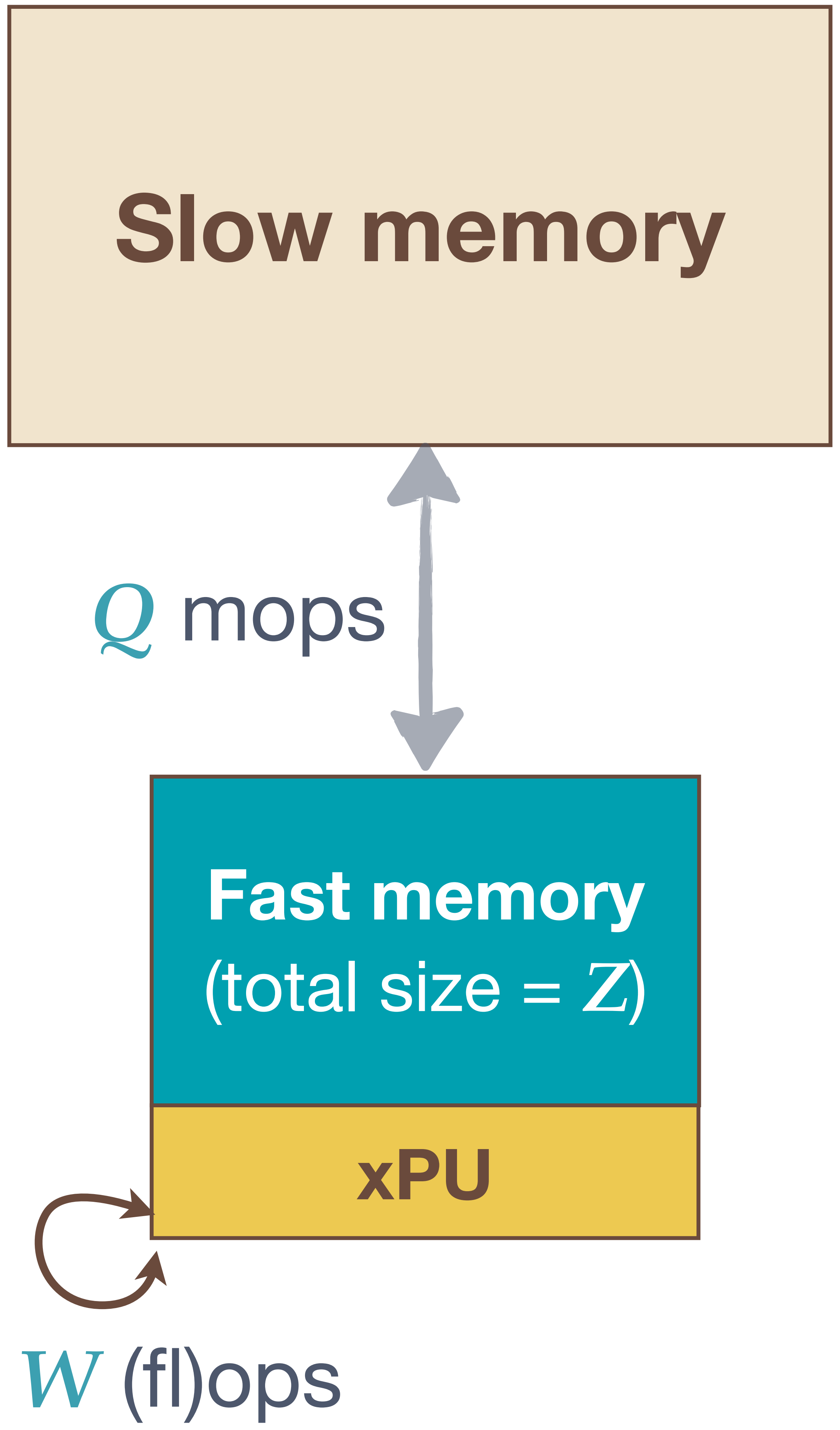
Exploiting Total Available Bandwidth



Q. What is effective **β** when we are streaming data from DRAM to the GPU

A. 170 GB/sec
B. 50 GB/sec
C. 64 GB/sec
D. 150 GB/sec

Locality and Cache Size




```
for (int k=0; k<n; k++) {
    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
        C[i][j] += A[i][k]*B[k][j]
```

$$W = 2n^3$$

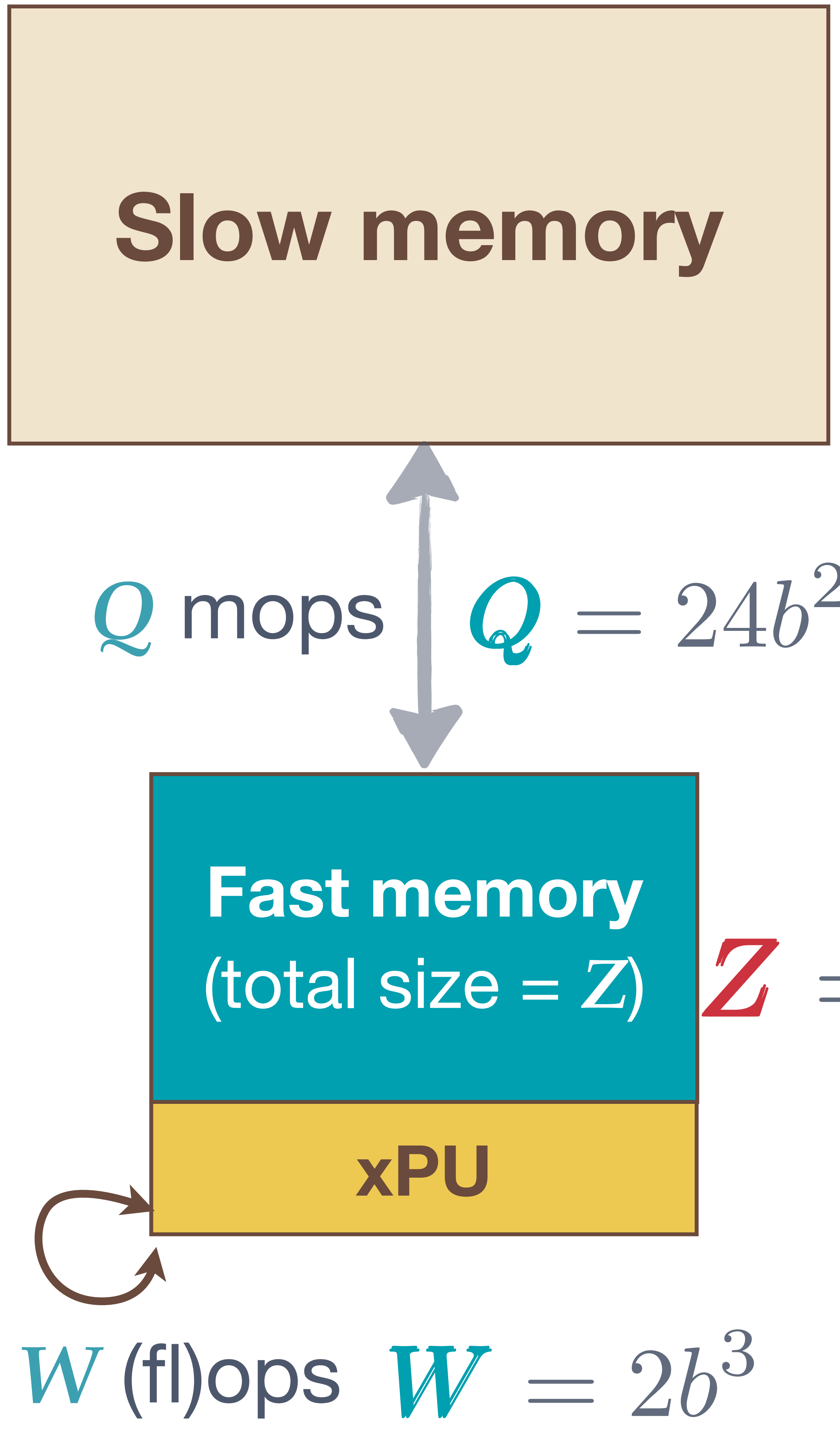
$$Q = 24n^2$$

$$I = \frac{n}{12}$$

What happens if the Cache size Z=O(1)?

$$I = \frac{1}{8}$$

Locality and Cache Size



$\boldsymbol{Q} = 24b^2$

$\boldsymbol{Z} = 24b^2$

$\boldsymbol{W} = 2b^3$

```
for (int kk=0; kk<n; kk+=b)
  for (int ii=0; ii<n; i+=b)
    for (int jj=0; jj<n; jj+=b)
      for (int k=kk; k< min(n,kk+b); k++)
        for (int i=ii; i<min(n,ii+b); i++)
          for (int j=jj; j<min(n,jj+b); j++)
            C[i][j] = min(C[i][j], A[i][k]+B[k][j])
```

$\boldsymbol{I} = \frac{b}{12}$ $\quad$ $\boldsymbol{I} = \frac{\sqrt{Z}}{24\sqrt{6}}$

$\boldsymbol{I} \geq \boldsymbol{B_\tau}$

$\implies \boldsymbol{Z} \geq 6192\boldsymbol{B_\tau^2}$

To sum up:

Summary

- Performance and scalability can be extremely non-intuitive even to computer scientists
- Performance is a question of how well an kernel's characteristics map to an architecture's characteristics
- The Roofline model is a **visually intuitive figure** for kernel analysis and optimization
- It is easily extended to other architectural paradigms.

To learn more:

- Williams, Samuel, et al. "The roofline model: A pedagogical tool for program analysis and optimization." *2008 IEEE Hot Chips 20 Symposium (HCS)*. IEEE, 2008.
- LBL's **Empirical Roofline Tool**: https://bitbucket.org/berkeleylab/cs-roofline-toolkit/src/master/
- **Computing FLOPs:** https://software.intel.com/content/www/us/en/develop/articles/calculating-flop-using-intel-software-development-emulator-intel-sde.html