The Roofline Model:

A Bridge between Computer Science, Applied Math, and Computational Science

Samuel Williams
Computational Research Division
Lawrence Berkeley National Lab
SWWilliams@lbl.gov

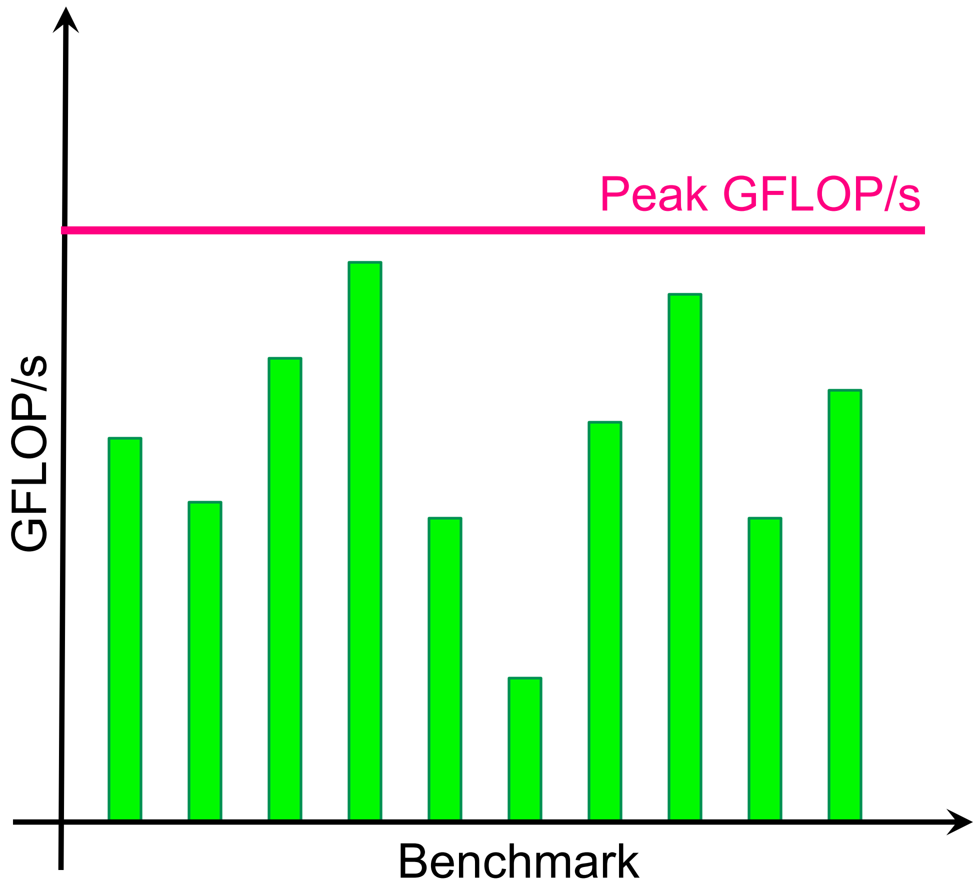
DOE is spending millions of dollars porting applications to GPUs…

How do we know if we are getting our money's worth?

Getting our money's worth?

- Really a question of good performance on applications benchmarks

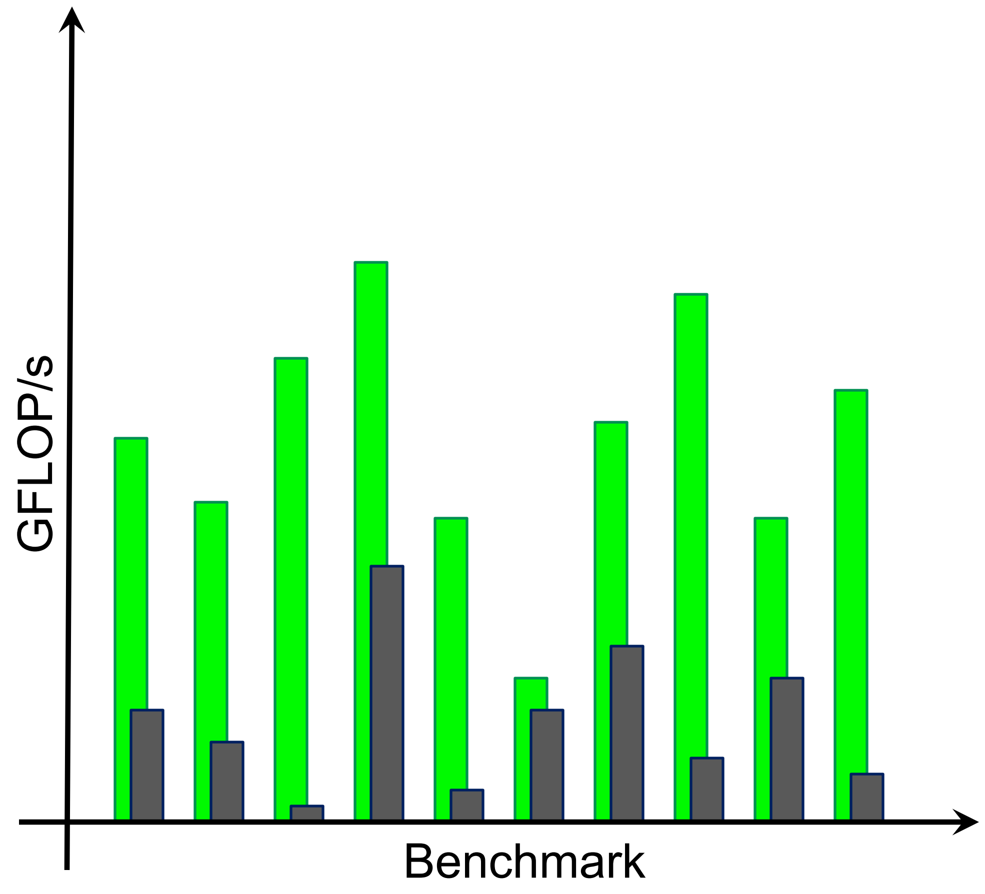
- Imagine profiling a mix of GPU-accelerated benchmarks …
- GFLOP/s alone may not be particularly insightful



Are we getting good performance?

- We could compare performance to a CPU…
  - Speedup may seem random
  - Aren't GPUs always 10x faster than a CPU?
  - If not, what does that tell us about architecture, algorithm or implementation?

  - **'Speedup' provides no insights into architecture, algorithm, or implementation.**
  - **'Speedup' provides no guidance to CS, AM, applications, procurement, or vendors.**

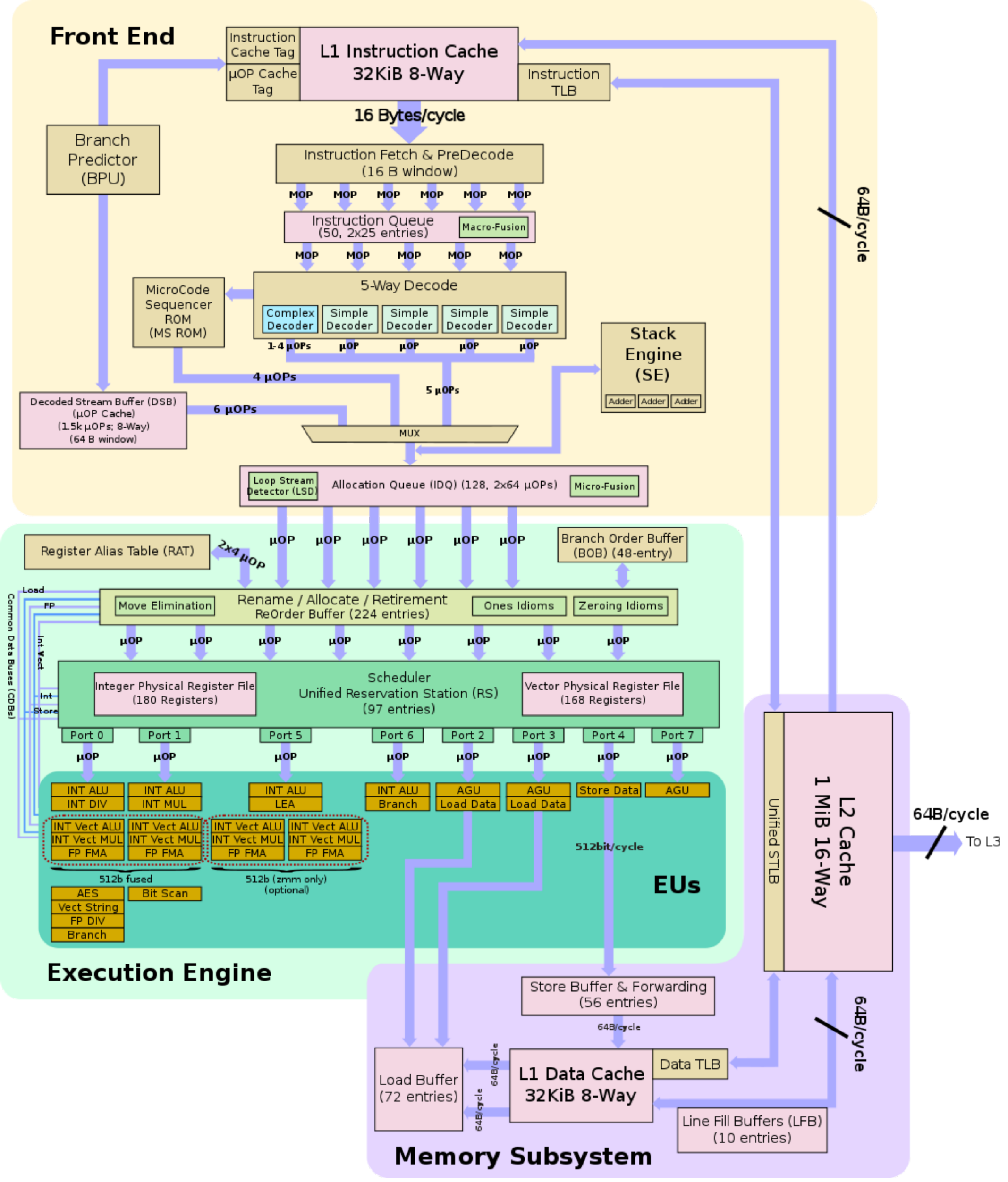


Are we getting good performance?

- We could take an architectural approach and build a simulator to understand every nuance of performance…
    - Modern architectures are incredibly complex
    - Simulators may perfectly reproduce performance, but can incur 10^6x slowdowns.

➢ **Provide no insights into algorithm or implementation.**

➢ **Provide no guidance to CS, AM, applications, or procurement.**



Are we getting good performance?

- We could take a CS approach and look at performance counters…
  - Record microarchitectural events on CPUs/GPUs
  - Use arcane architecture-specific terminology
  - May be broken

  - We may be able to show correlation between events, but…
  - **…providing actionable guidance to CS, AM, applications, or procurement can prove elusive.**

```
.
.
.
FRONTEND_RETIRED.LATENCY_GE_8_PS
FRONTEND_RETIRED.LATENCY_GE_16_PS
FRONTEND_RETIRED.LATENCY_GE_32_PS
RS_EVENTS.EMPTY_END
FRONTEND_RETIRED.L2_MISS_PS
FRONTEND_RETIRED.L1I_MISS_PS
FRONTEND_RETIRED.STLB_MISS_PS
FRONTEND_RETIRED.ITLB_MISS_PS
ITLB_MISSES.WALK_COMPLETED
BR_MISP_RETIRED.ALL_BRANCHES_PS
IDQ.MS_SWITCHES
FRONTEND_RETIRED.LATENCY_GE_2_BUBBLES_GE_1_PS
BR_MISP_RETIRED.ALL_BRANCHES_PS
MACHINE_CLEARS.COUNT
MEM_LOAD_RETIRED.L1_HIT_PS
MEM_LOAD_RETIRED.FB_HIT_PS
MEM_LOAD_UOPS_RETIRED.L1_HIT_PS
MEM_LOAD_UOPS_RETIRED.HIT_LFB_PS
MEM_INST_RETIRED.STLB_MISS_LOADS_PS
MEM_UOPS_RETIRED.STLB_MISS_LOADS_PS
MEM_LOAD_RETIRED.L2_HIT_PSMEM_LOAD_UOPS_RETIRED.L2_HIT_PS
MEM_LOAD_RETIRED.L3_HIT_PS
MEM_LOAD_UOPS_RETIRED.LLC_HIT_PS
MEM_LOAD_UOPS_RETIRED.L3_HIT_PS
MEM_LOAD_RETIRED.L3_MISS_PS
MEM_LOAD_UOPS_RETIRED.LLC_MISS_PS
MEM_LOAD_UOPS_MISC_RETIRED.LLC_MISS_PS
MEM_LOAD_UOPS_RETIRED.L3_MISS_PS
MEM_INST_RETIRED.ALL_STORES_PS
MEM_UOPS_RETIRED.ALL_STORES_PS
ARITH.DIVIDER_ACTIVE
ARITH.DIVIDER_UOPS
ARITH.FPU_DIV_ACTIVE
INST_RETIRED.PREC_DIST
IDQ.MS_UOPS
INST_RETIRED.PREC_DIST
.
.
.
```


BERKELEY LAB

What's missing…

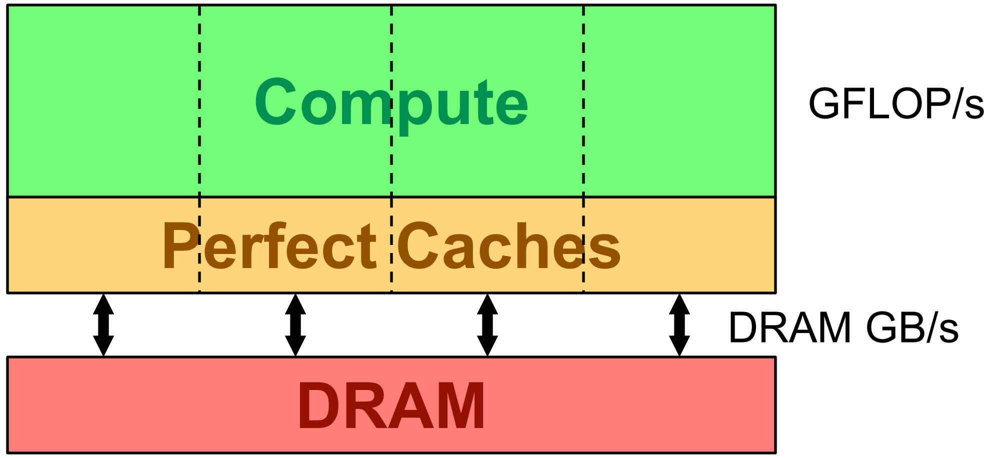
- Each community speaks their own language and develops specialized tools/methodologies

- Need common mental model of application execution on a target system
- Sacrifice accuracy to gain…
  - Architecture independence / extensibility
  - Readily understandable by the extremely broad DOE community
  - intuition, insights, and guidance to CS, AM, apps, procurement, and vendors

➤ **Roofline is just such a model**

Data Movement or Compute?

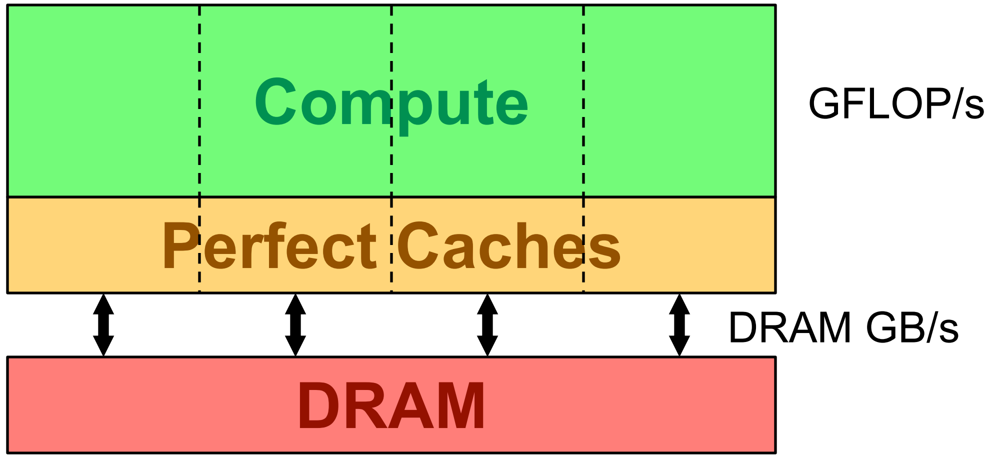
- Which takes longer?
  - Data Movement
  - Compute?



$$\text{Time} = \max \begin{cases} \text{\#FP ops / Peak GFLOP/s} \\ \text{\#Bytes / Peak GB/s} \end{cases}$$

Data Movement or Compute?

- Which takes longer?
  - Data Movement
  - Compute?
- Is performance limited by compute or data movement?

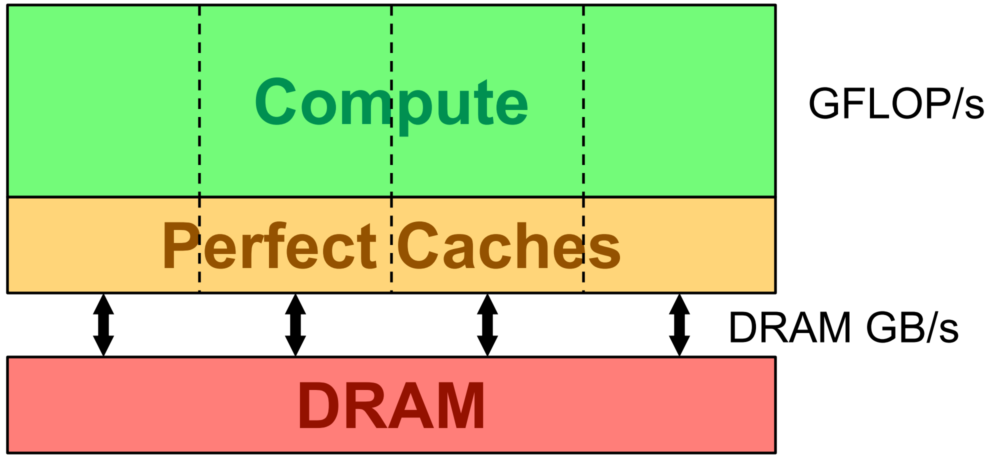


$$\frac{\text{Time}}{\text{\#FP ops}} = \max \begin{cases} 1 \,/\, \text{Peak GFLOP/s} \\ \text{\#Bytes} \,/\, \text{\#FP ops} \,/\, \text{Peak GB/s} \end{cases}$$

BERKELEY LAB

Data Movement or Compute?

- Which takes longer?
  - Data Movement
  - Compute?
- Is performance limited by compute or data movement?

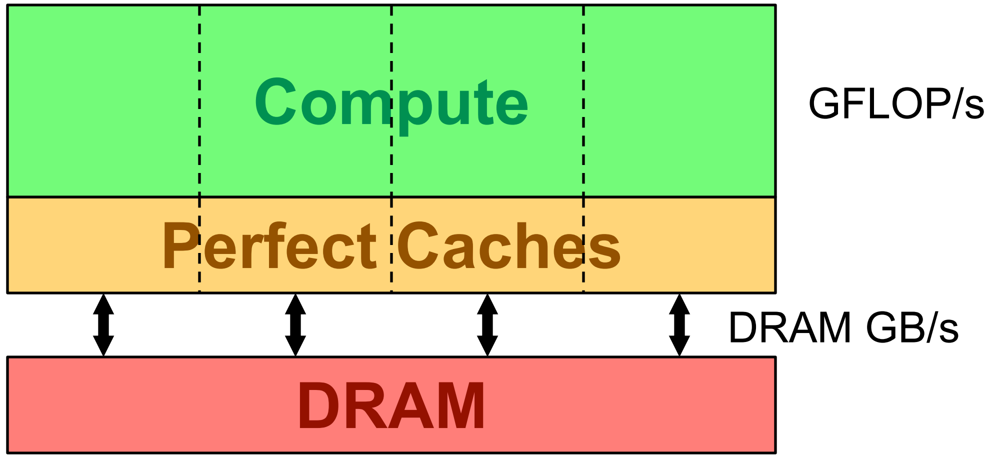


$$\frac{\#\text{FP ops}}{\text{Time}} = \min \begin{cases} \text{Peak GFLOP/s} \\ (\#\text{FP ops} / \#\text{Bytes}) * \text{Peak GB/s} \end{cases}$$

BERKELEY LAB

Data Movement or Compute?

- Which takes longer?
  - Data Movement
  - Compute?
- Is performance limited by compute or data movement?



$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI * Peak GB/s} \end{cases}$$

Arithmetic Intensity (AI) = measure of data locality

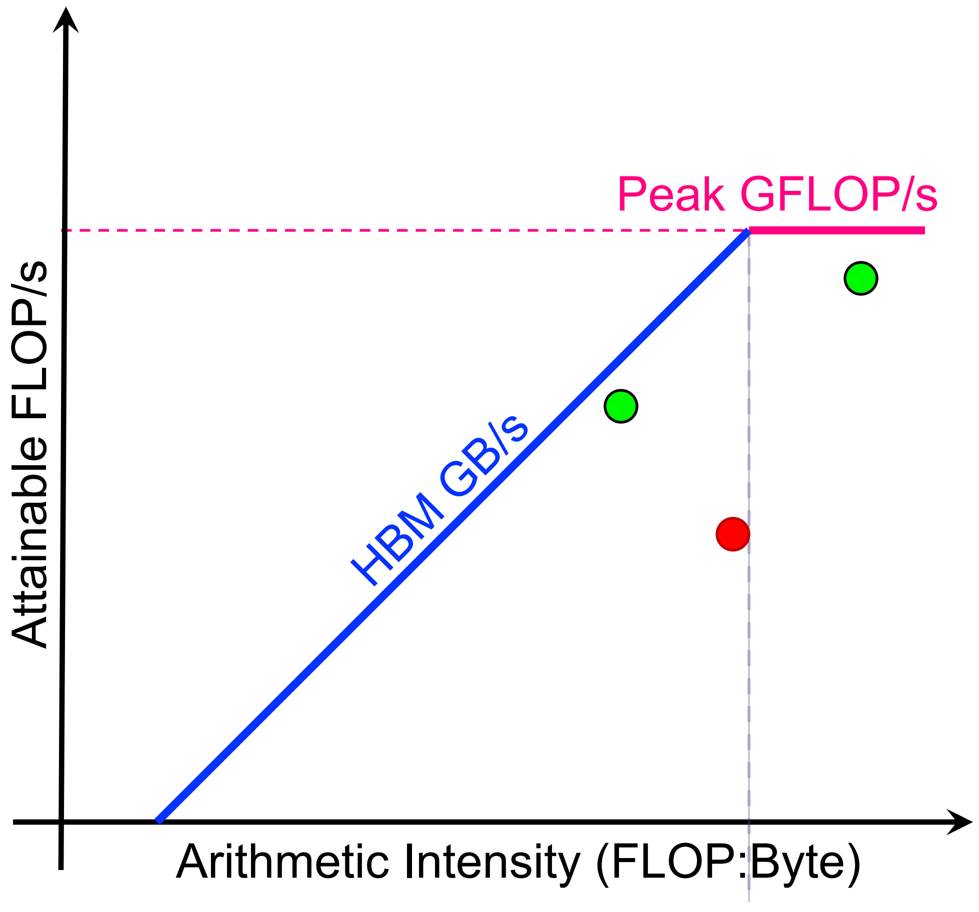
BERKELEY LAB

(DRAM) Roofline Model

$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI * Peak GB/s} \end{cases}$$

AI (Arithmetic Intensity) = FLOPs / Bytes (moved to/from DRAM)

- Plot bound on **Log-log scale** as a function of AI (data locality)
- Measure application (AI,GF/s) and scatter plot in the resultant 2D locality-performance plane.



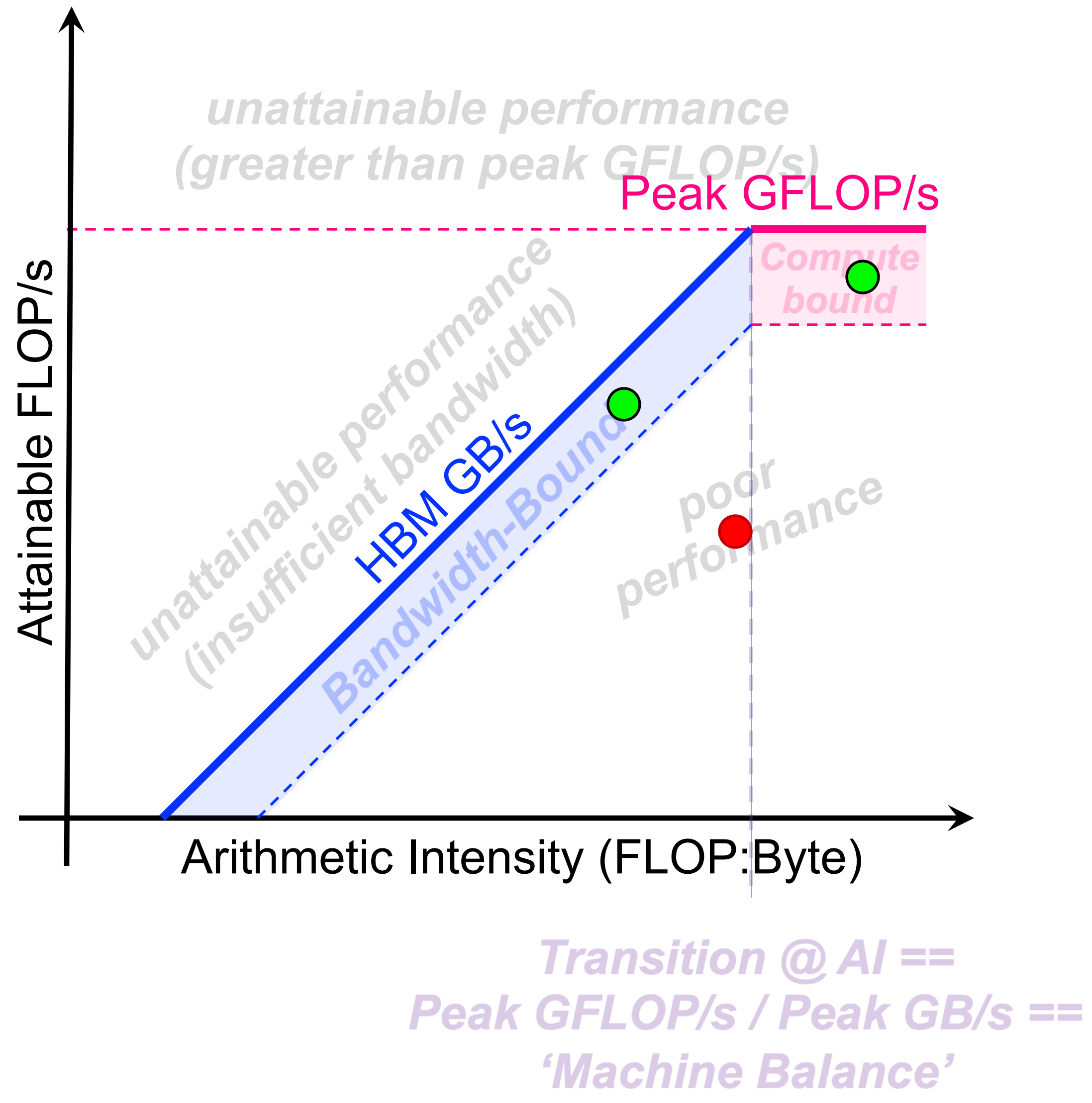
Transition @ AI ==
Peak GFLOP/s / Peak GB/s ==
'Machine Balance'

(DRAM) Roofline Model

$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI * Peak GB/s} \end{cases}$$

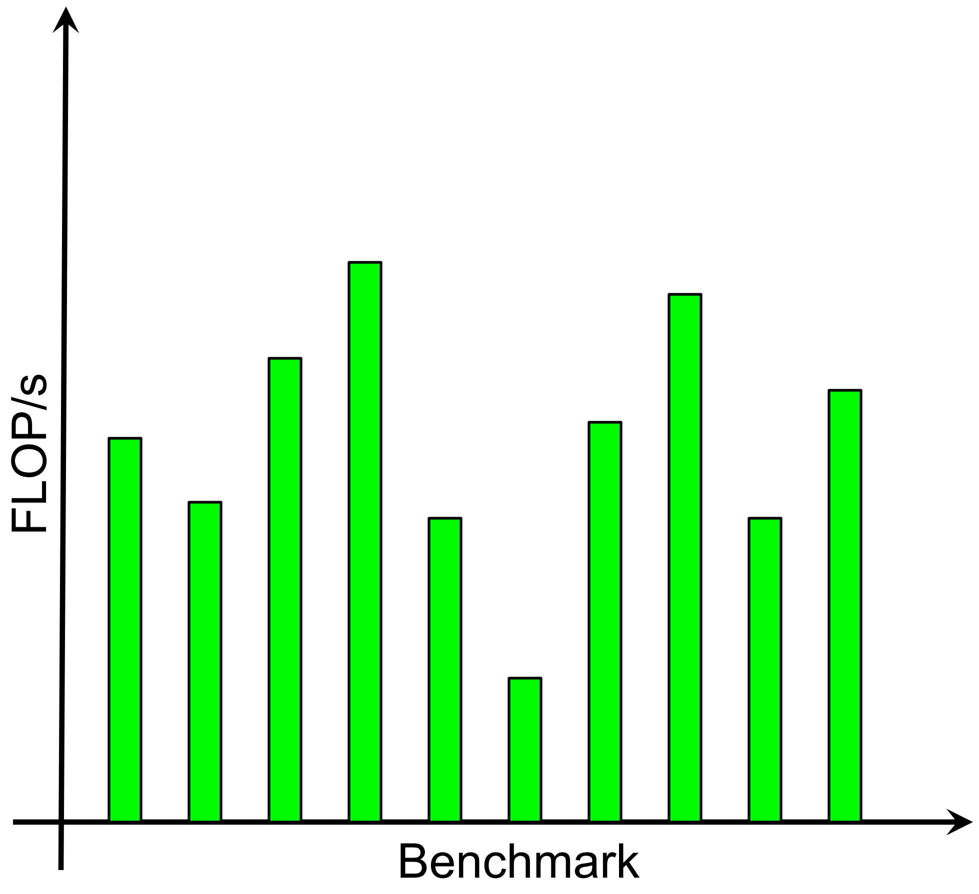
AI (Arithmetic Intensity) = FLOPs / Bytes (moved to/from DRAM)

- Roofline tessellates the locality-performance plane into five regions…



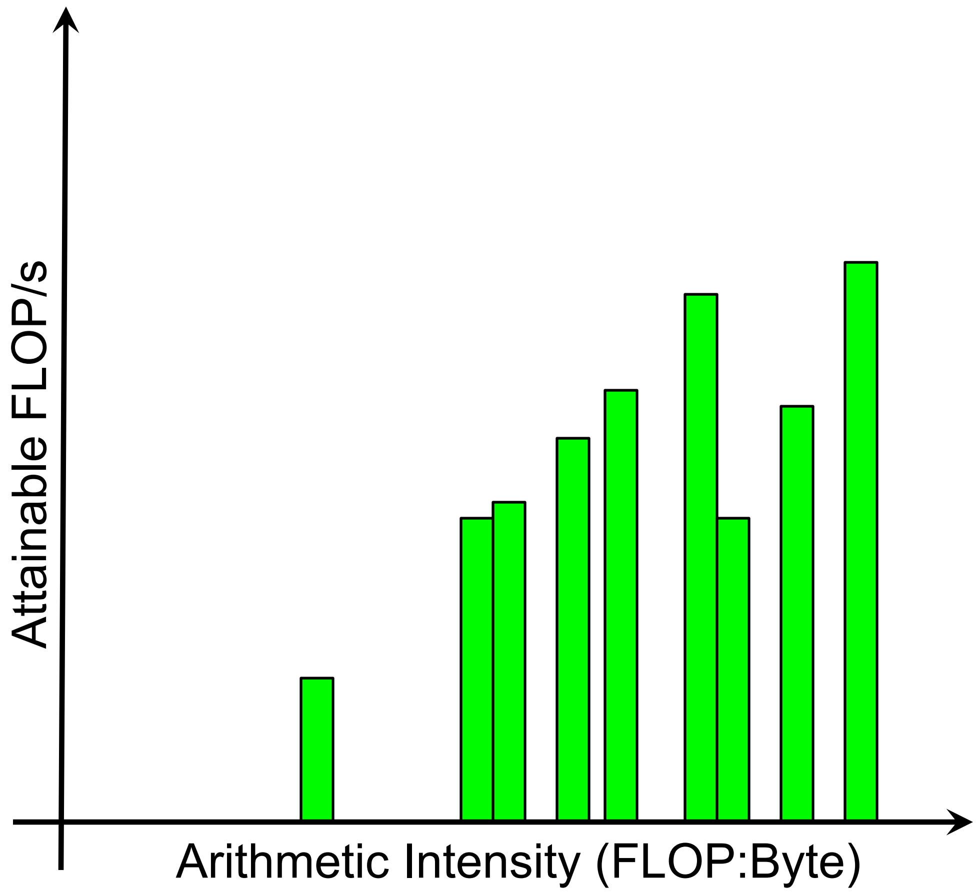
Are we getting good performance?

- Think back to our mix of benchmarks…



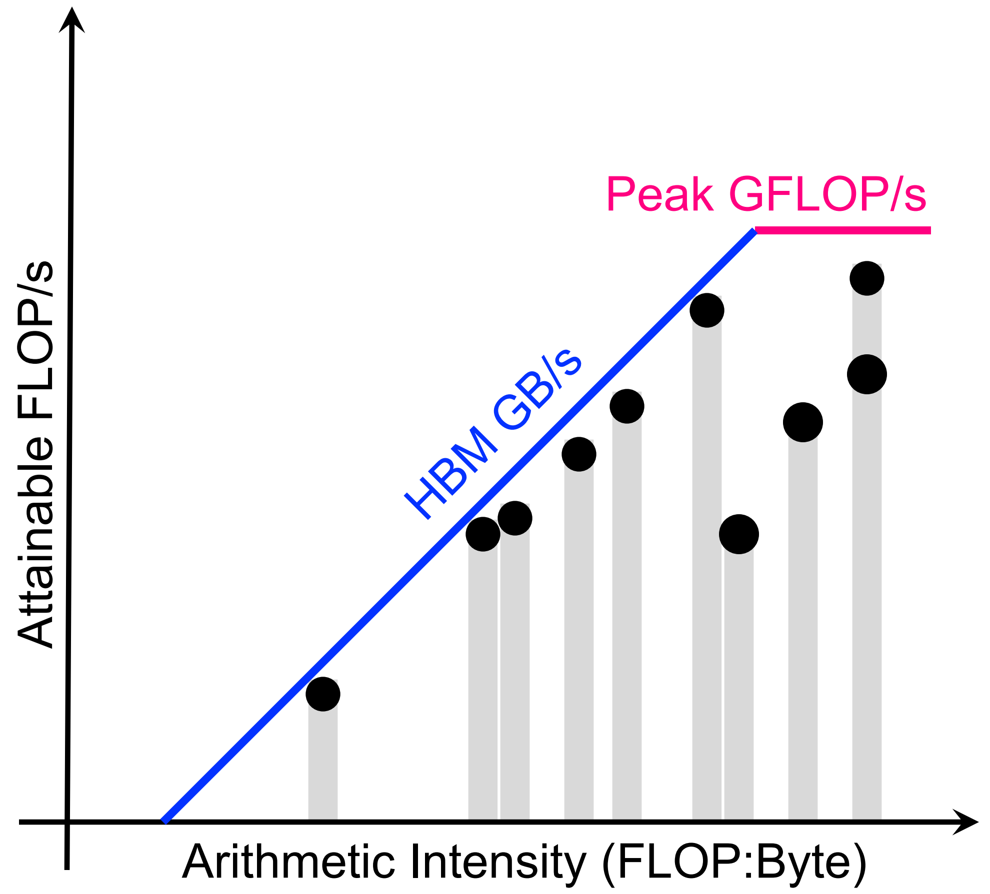
Are we getting good performance?

- We can sort benchmarks by arithmetic intensity…



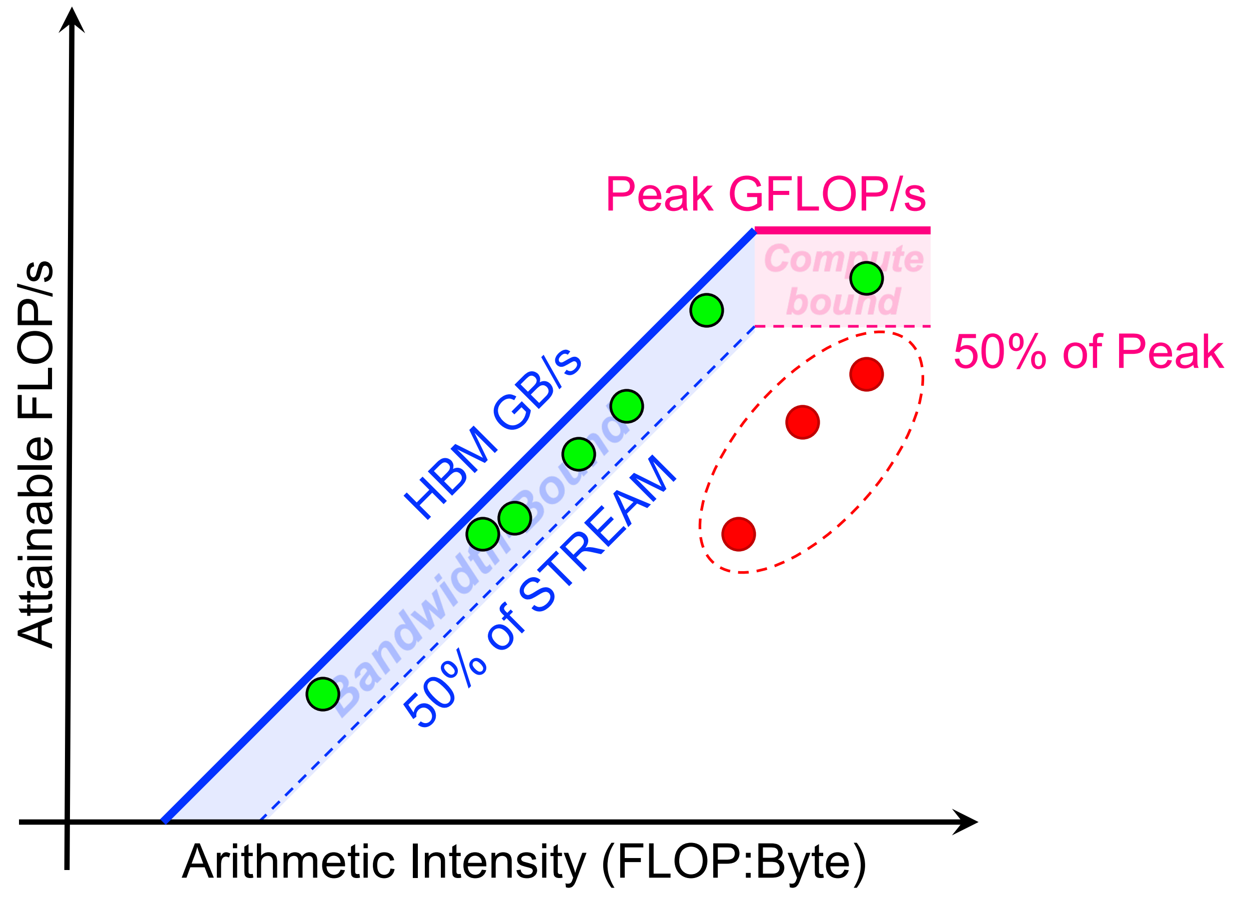
Are we getting good performance?

- We can sort benchmarks by arithmetic intensity…
- … and compare performance relative to machine capabilities



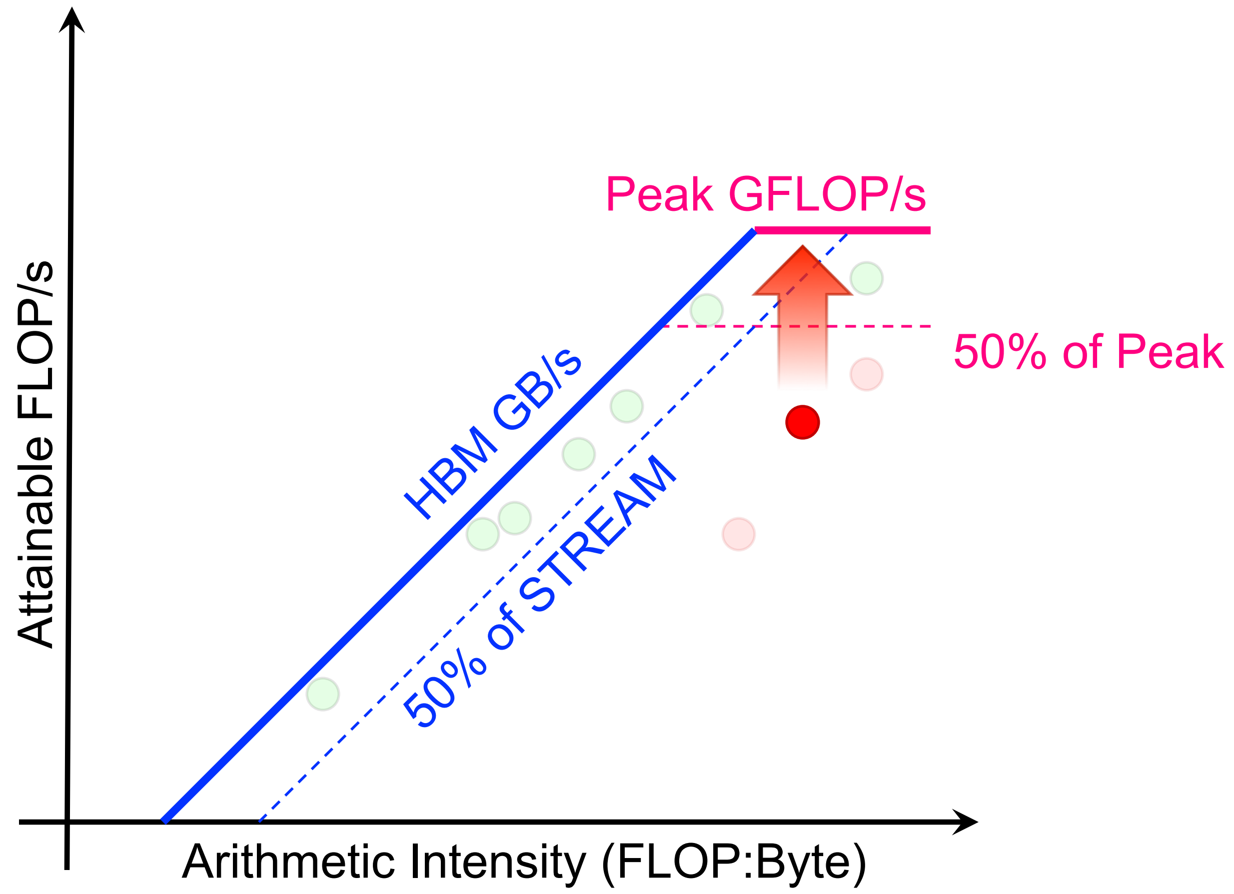
Are we getting good performance?

- Benchmarks near the roofline are making **good use** of computational resources



General Performance Optimization Strategy

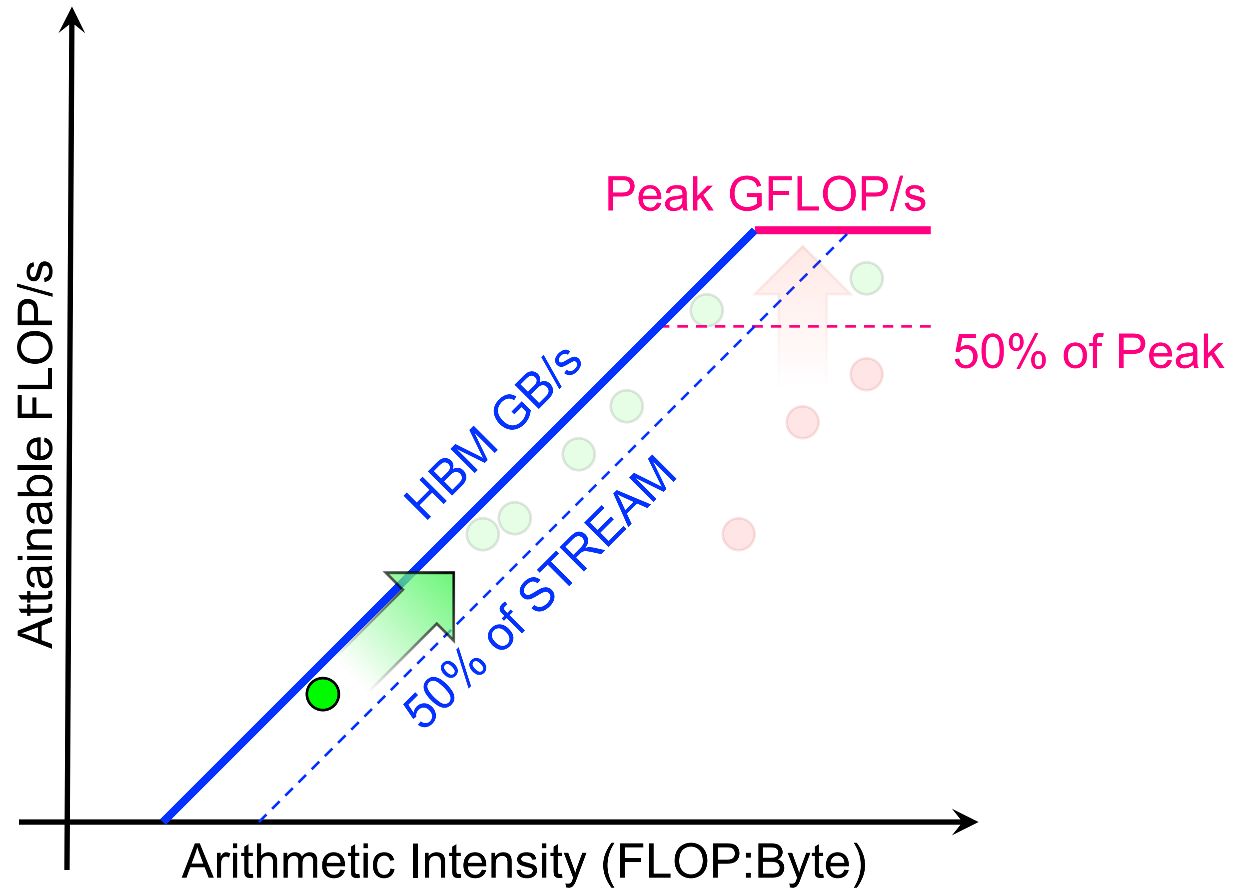
- Get to the Roofline



BERKELEY LAB

General Performance Optimization Strategy

- Get to the Roofline
- Increase Arithmetic Intensity when bandwidth-limited
  - Reducing data movement increases AI



Roofline and SciDAC

15 years of Roofline R&D

- 15 years of Roofline activities fall into 3 categories:
  - Research into extending the model
  - Prototype implementations
  - Integration in production tool

- Virtuous Cycle
  - Research gives rise to prototypes
  - Prototypes give rise to production/integration
  - Prototypes/Production give rise to new research

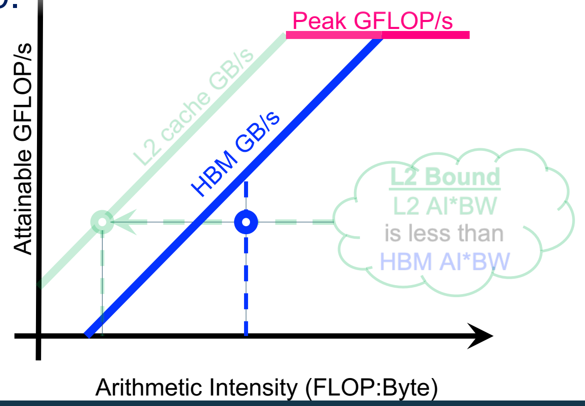
Research: Extending the Model

Simple DRAM model can be insufficient for a variety of reasons…

DRAM's not the bottleneck…
- Cache bandwidth and cache locality
- PCIe bandwidth

…The Hierarchical Roofline Model

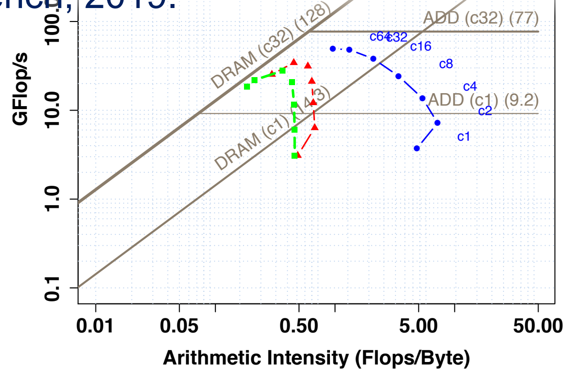
C. Yang, T. Kurth, S. Williams, "Hierarchical Roofline analysis for GPUs: Accelerating performance optimization for the NERSC-9 Perlmutter system", CCPE, 2019.



Lack of Parallelism…
- Idle Cores/SMs
- Insufficient ILP/TLP
- Divergence and Predication

… Roofline Scaling Trajectories

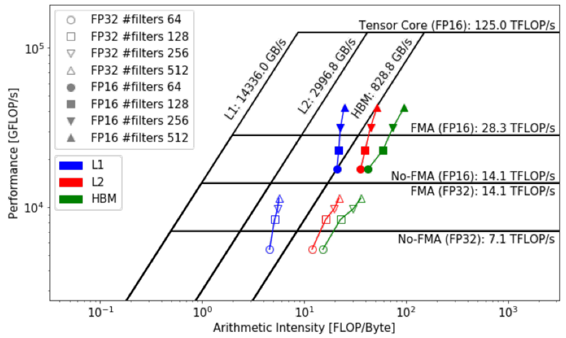
K. Ibrahim, S. Williams, L. Oliker, "Performance Analysis of GPU Programming Models using the Roofline Scaling Trajectories", BEST PAPER, Bench, 2019.



Machine Learning Applications…
- Mixed Precision
- Not enough Tensor Core OPs

… Additional Ceilings

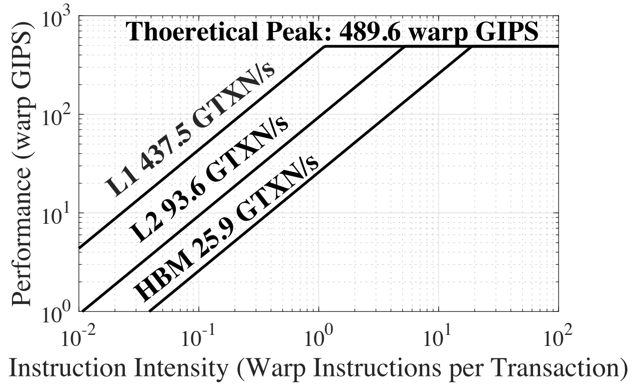
C. Yang, T. Kurth, S. Williams, "Hierarchical Roofline analysis for GPUs: Accelerating performance optimization for the NERSC-9 Perlmutter system", CCPE, 2019.



Integer-heavy Codes…
- Non-FP inst. impede FLOPs
- No FP instructions

… The Instruction Roofline Model

N. Ding, S. Williams, "An Instruction Roofline Model for GPUs", BEST PAPER, PMBS, 2019.



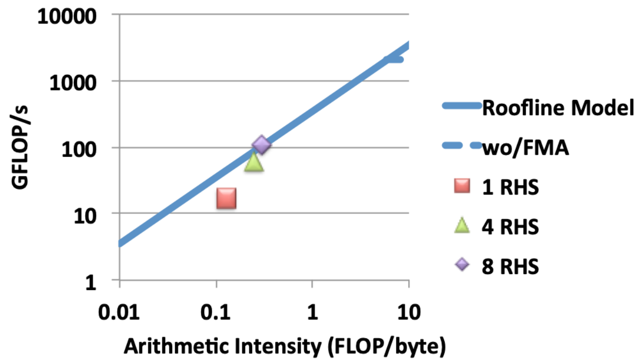
BERKELEY LAB

Prototypes: Using Roofline for Applications

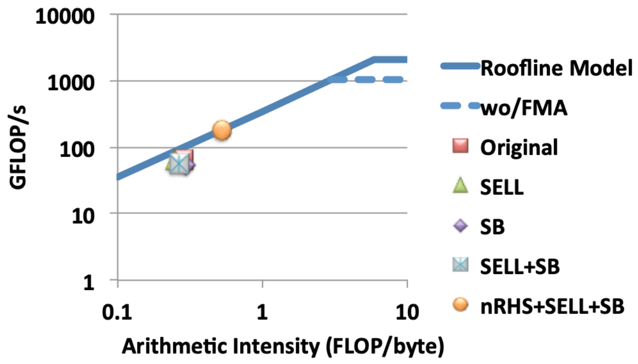
- Resultant prototypes allowed friendly stakeholders to evaluate apps using Roofline
- More recently, DOE centers used Roofline to analyze their applications…

Roofline at NERSC/LBL[1]
- KNL (Cori)
- Compared against Haswell (Xeon)
- **KNL optimization/readiness as a performance trendline**

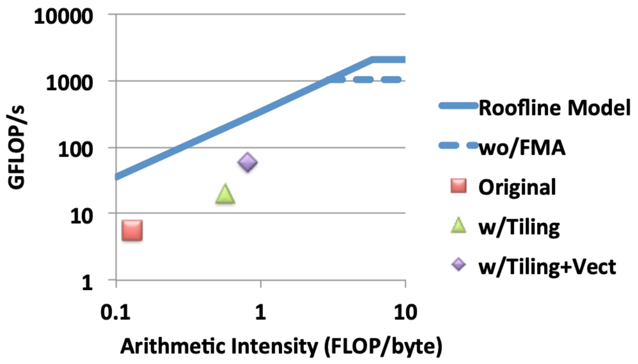
MFDn



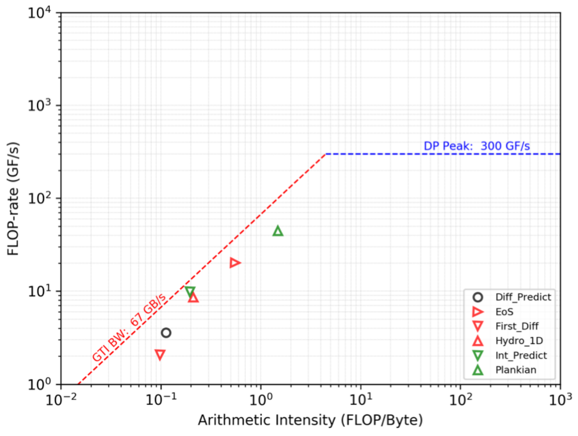
EMGeo

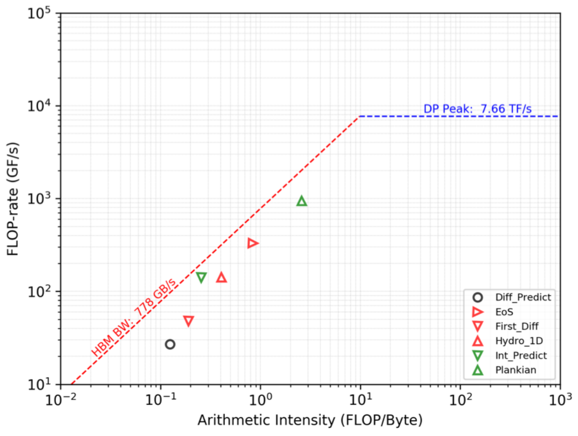


PICSAR



Roofline used at ALCF[2]
- OpenCL and SYCL(shown)
- RAJA LCALS benchmarks
- **Intel Gen9 GT4e (left) and NVIDIA V100 (right)**



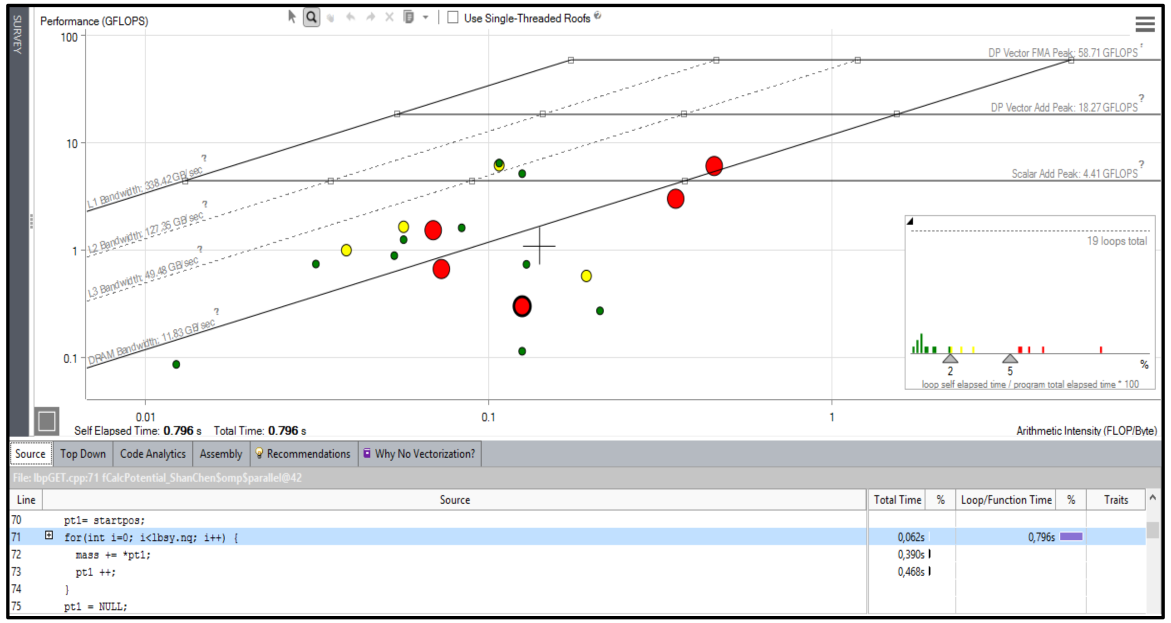


[1]Charlene Yang, Jack Deslippe, NERSC
[2]Brian Homerding, Colleen Bertoni, and Jaehyuk Kwack, ANL, ECP 2020 meeting

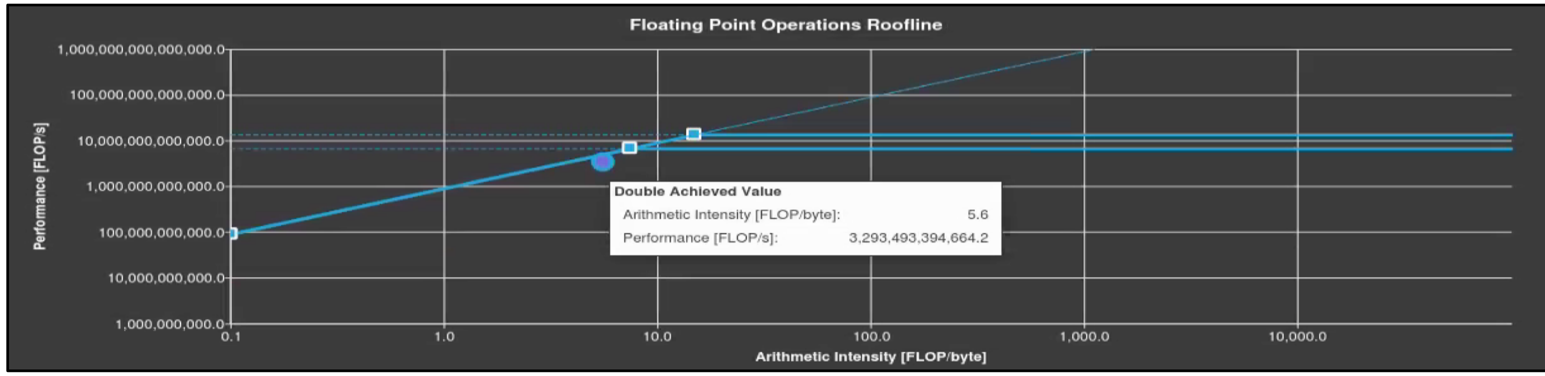
Productization: Vendor Integration

- Ultimately, users and DOE Centers don't want CS prototypes
- Need maintained, production-quality tools
- Roofline team collaborated with NERSC, ALCF, Intel, and NVIDIA…

Integration of Roofline into Intel Advisor (2017)



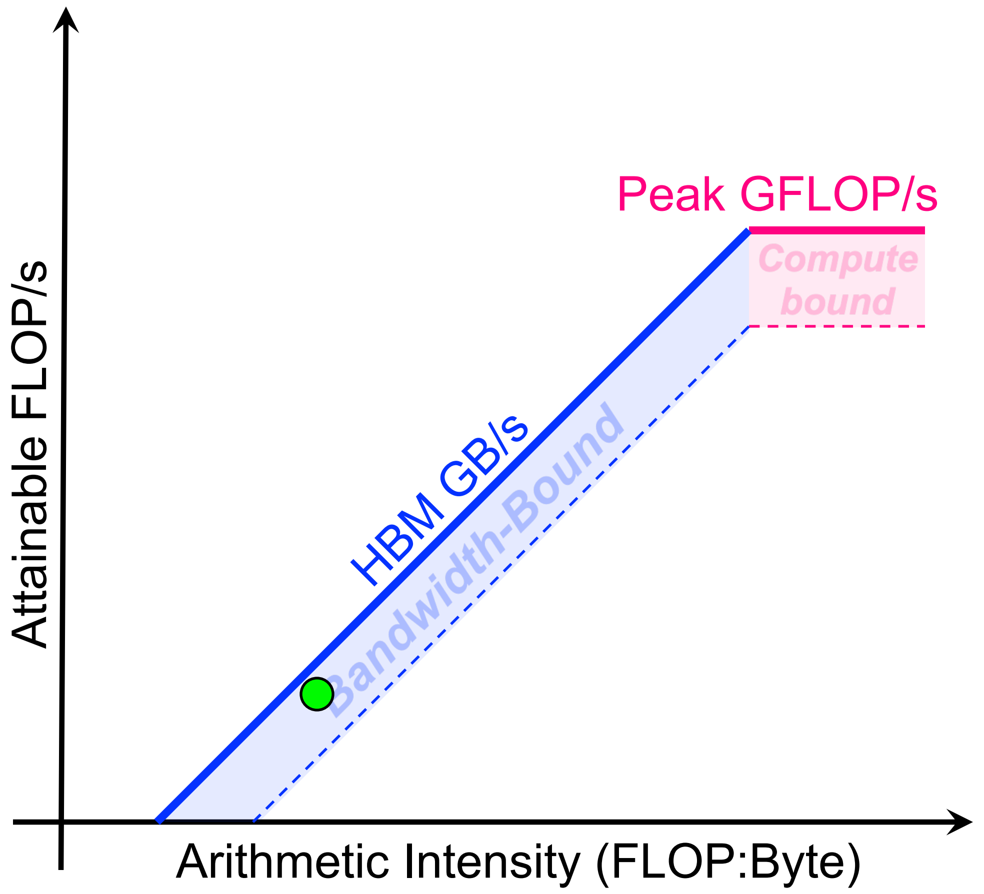
Integration of Roofline into NVIDIA Nsight Compute (2020)



BERKELEY LAB

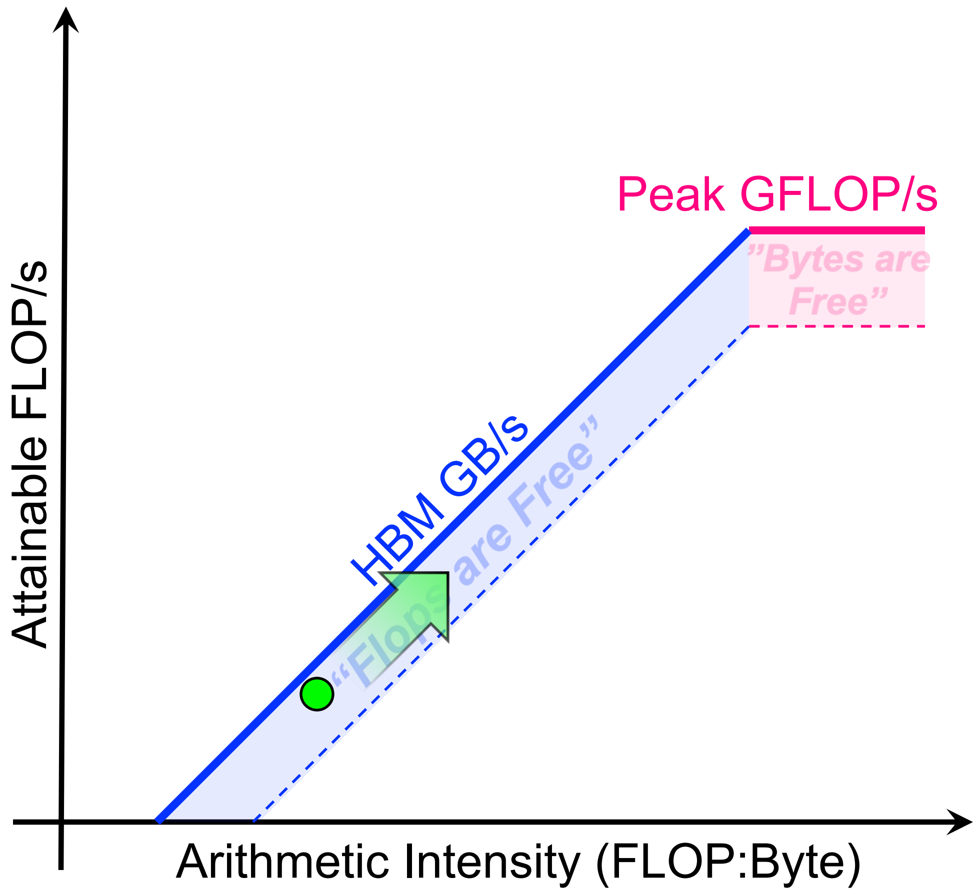
Optimization is only the beginning

- What do we do when we're on the Roofline?
- Need better algorithms
- Roofline enabled productive collaborations with Applied Math community



Computer Science-Applied Math CoDesign

- Think back to how Roofline tessellates the locality-performance plane
- We can rename the regions…
  - apps/algorithms in the "**FLOPs are free**" region can do extra FLOPs if they move less data
  - those in the "**Bytes are free**" region can move extra data if it saves them on FLOPs
- Motivates and provides quantitative analysis for the potential for…
  - Alternate data structures/representations
  - re/precomputation of data
  - communication-avoiding algorithms
  - alternate and high order numerical methods



Summary

Take away

- Roofline has helped hundreds of researchers understand and describe application performance relative to machine capabilities

- helps frame the conversation between…
  - Application Developers
  - Computer Scientists
  - Applied Mathematicians
  - Processor Vendors

…by providing a common language and mental model

- 15 years of SciDAC funding transformed Roofline from a whiteboard doodle into an integral component embedded within vendor performance analysis tools installed at multiple DOE centers

Thanks to the Entire Roofline Team…

Charlene Yang, Doug Doerfler, Thorsten Kurth,
Khaled Ibrahim, Nan Ding, Yunsong Wang, Jonathan Madsen,
Brian Van Straalen, Protonu Basu, Terry Ligocki,
Linda Lo, Matt Cordery, Andrew Waterman,
Jack Deslippe, Lenny Oliker, David Patterson

Acknowledgements

- This material is based upon work supported by the Advanced Scientific Computing Research Program in the U.S. Department of Energy, Office of Science, under Award Number DE-AC02-05CH11231.
- This material is based upon work supported by the DOE RAPIDS SciDAC Institute.
- This research used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.
- This research used resources of the Oak Ridge Leadership Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

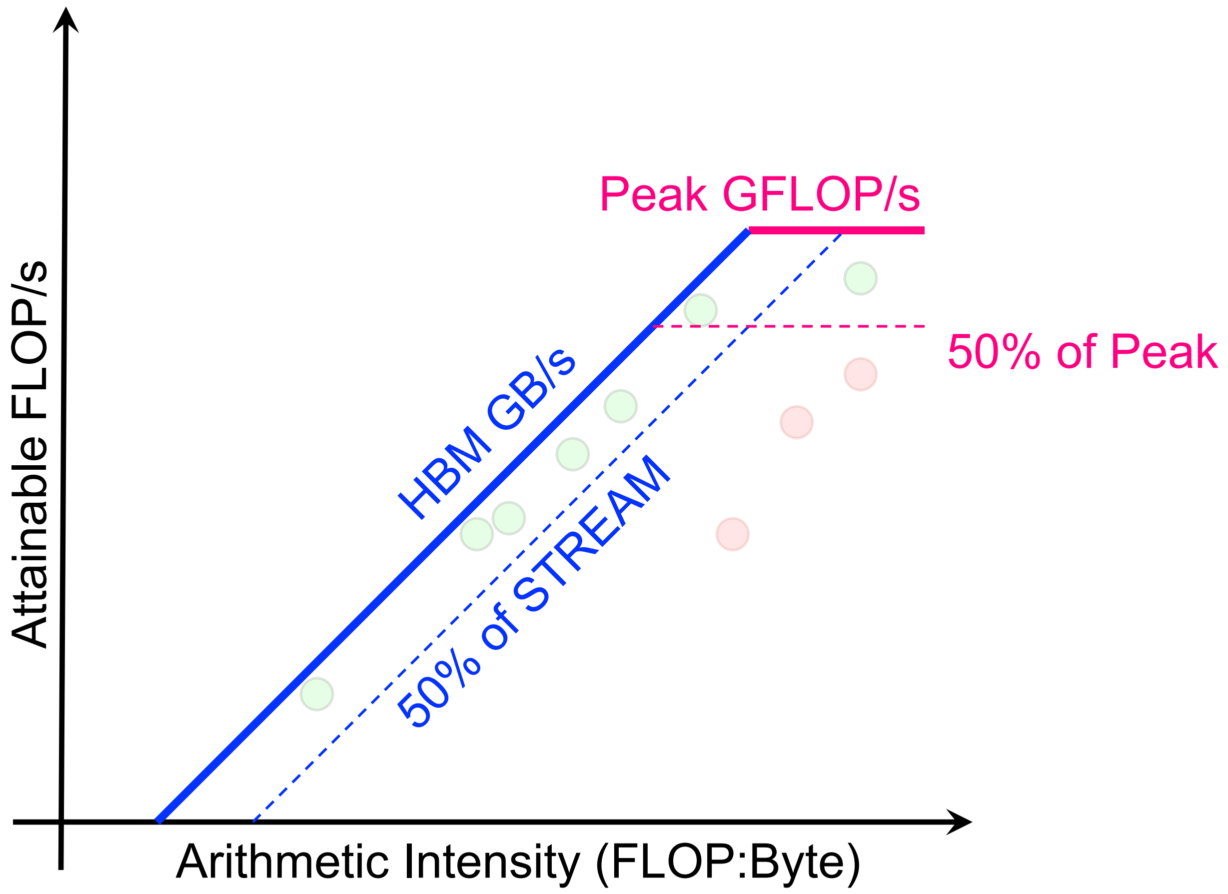
BACKUP

Arithmetic Intensity

- Measure of data locality (data reuse)
- Ratio of **Total Flops** performed to **Total Bytes** moved
- For the DRAM Roofline…
  - Total Bytes to/from DRAM
  - Includes all cache and prefetcher effects
  - Can be very different from total loads/stores (bytes requested)
  - Equal to ratio of sustained GFLOP/s to sustained GB/s (time cancels)

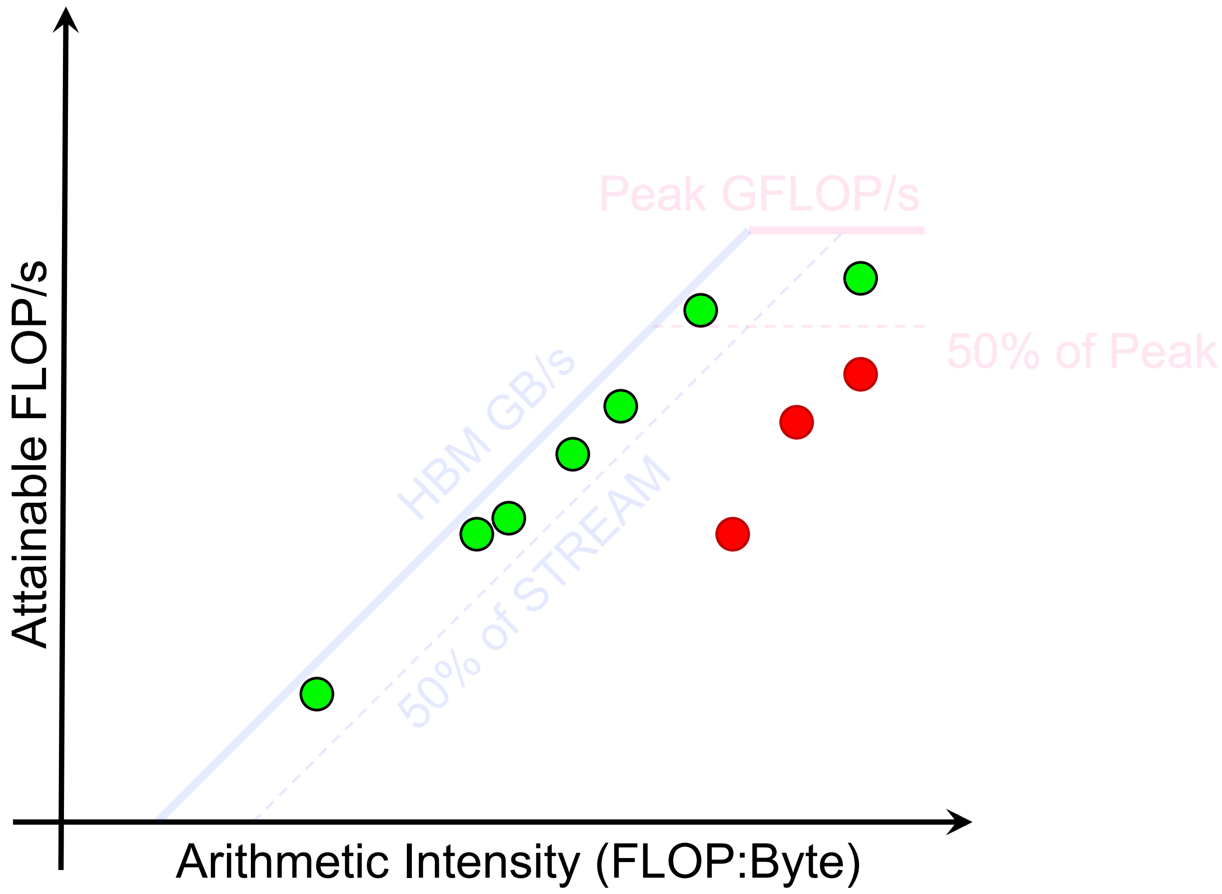
Roofline is made of two components

- Machine Model
  - Lines defined by peak GB/s and GF/s (**Benchmarking**)
  - Unique to each architecture
  - Common to all apps on that architecture



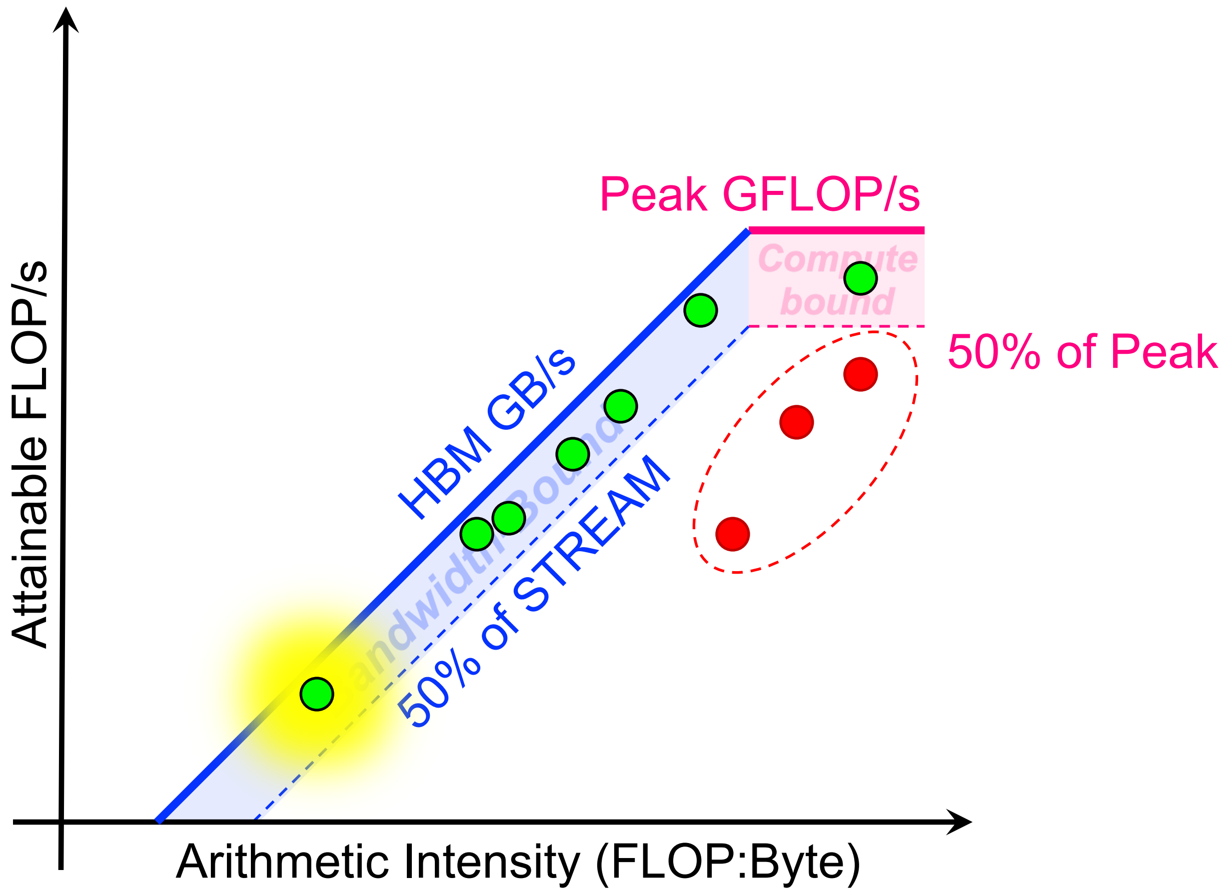
Roofline is made of two components

- Machine Model
  - Lines defined by peak GB/s and GF/s (**Benchmarking**)
  - Unique to each architecture
  - Common to all apps on that architecture
- Application Characteristics
  - Dots defined by application GFLOP's and GB's (**Application Instrumentation**)
  - Unique to each application
  - Unique to each architecture



What is “Good” Performance?

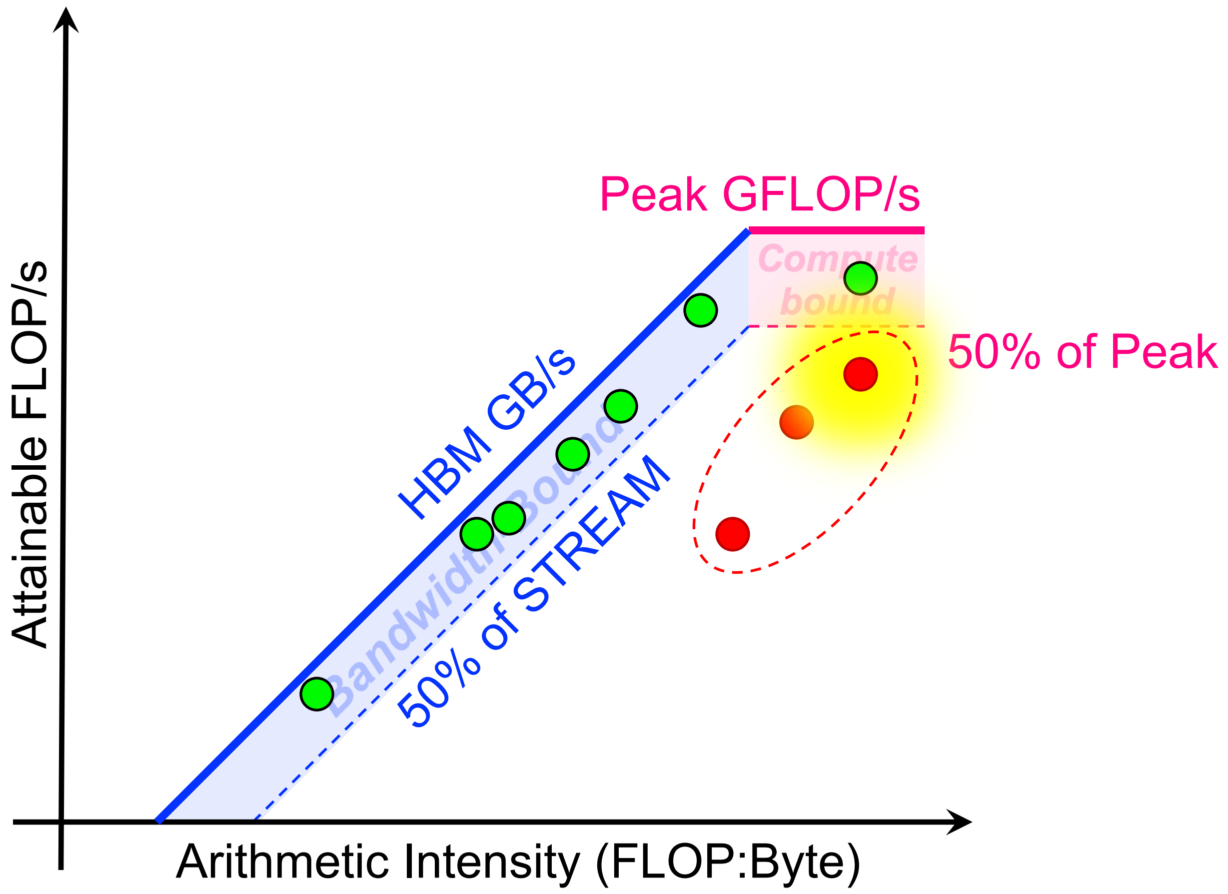
- Benchmarks near the roofline are making **good use** of computational resources
  - benchmarks can have **low performance** (GFLOP/s), but make **good use** (%STREAM) of a machine



BERKELEY LAB

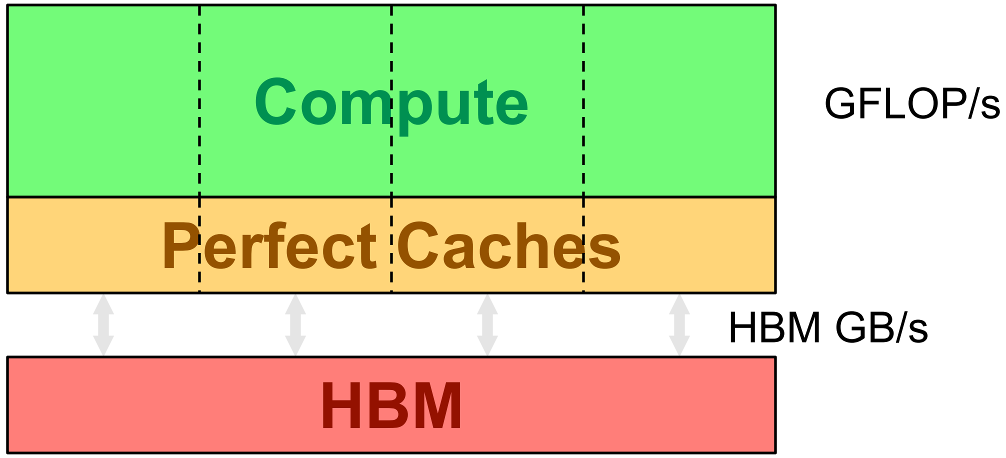
What is “Good” Performance?

- Benchmarks near the roofline are making **good use** of computational resources
  - benchmarks can have **low performance** (GFLOP/s), but make **good use** (%STREAM) of a machine
  - benchmarks can have **high performance** (GFLOP/s), but still make **poor use** of a machine (%peak)



Roofline Example

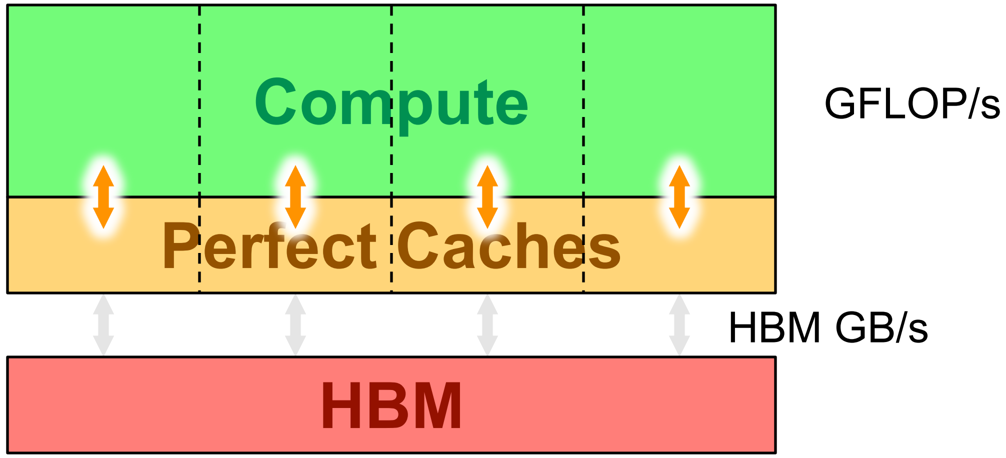
- Consider a 7-point constant coefficient stencil…




```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                    + old[k  ][j  ][i-1]
                    + old[k  ][j  ][i+1]
                    + old[k  ][j-1][i  ]
                    + old[k  ][j+1][i  ]
                    + old[k-1][j  ][i  ]
                    + old[k+1][j  ][i  ];
}}}
```

Roofline Example

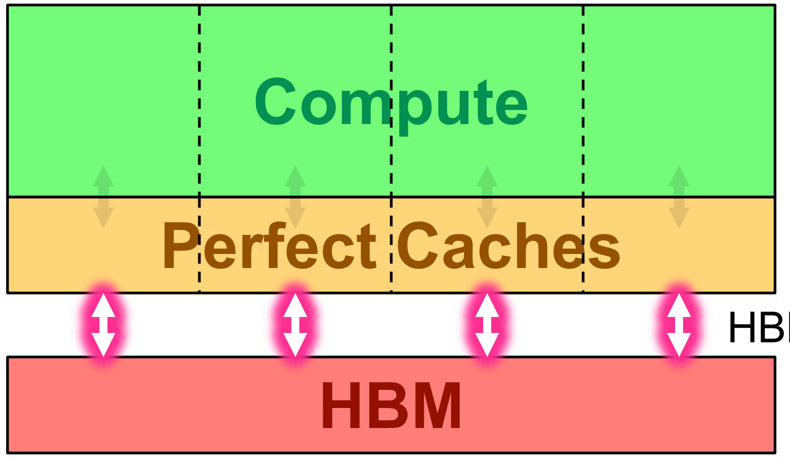
- Consider a 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - **AI = 7 / (8*8) = 0.11 FLOPs per byte**
    (measured at the L1)




```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                    + old[k  ][j  ][i-1]
                    + old[k  ][j  ][i+1]
                    + old[k  ][j-1][i  ]
                    + old[k  ][j+1][i  ]
                    + old[k-1][j  ][i  ]
                    + old[k+1][j  ][i  ]
}}}
```

Roofline Example

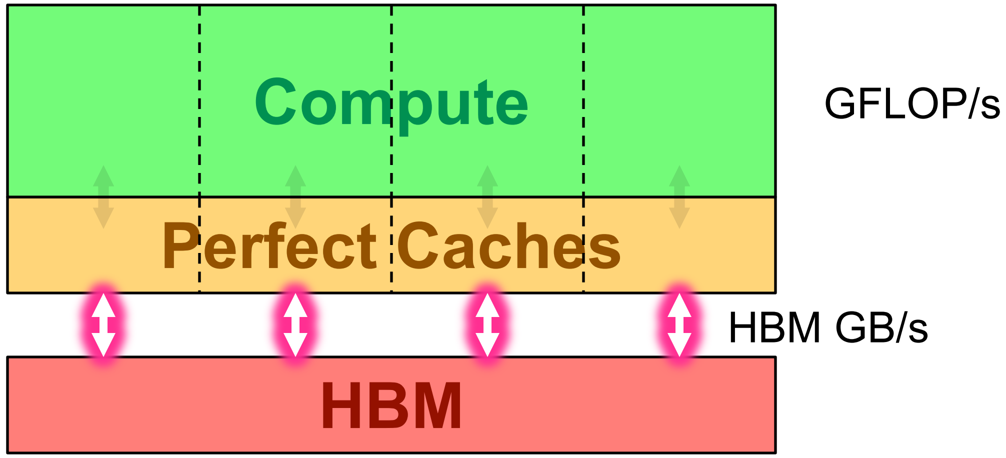
- Consider a 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - **Ideally, cache will filter all but 1 read and 1 write per point**




```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                    + old[k  ][j  ][i-1]
                    + old[k  ][j  ][i+1]
                    + old[k  ][j-1][i  ]
                    + old[k  ][j+1][i  ]
                    + old[k-1][j  ][i  ]
                    + old[k+1][j  ][i  ]
}}}
```

Roofline Example

- Consider a 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - Ideally, cache will filter all but 1 read and 1 write per point
  - ➤ **7 / (8+8) = 0.44 FLOPs per byte (DRAM)**




```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                    + old[k  ][j  ][i-1]
                    + old[k  ][j  ][i+1]
                    + old[k  ][j-1][i  ]
                    + old[k  ][j+1][i  ]
                    + old[k-1][j  ][i  ]
                    + old[k+1][j  ][i  ];
}}}
```

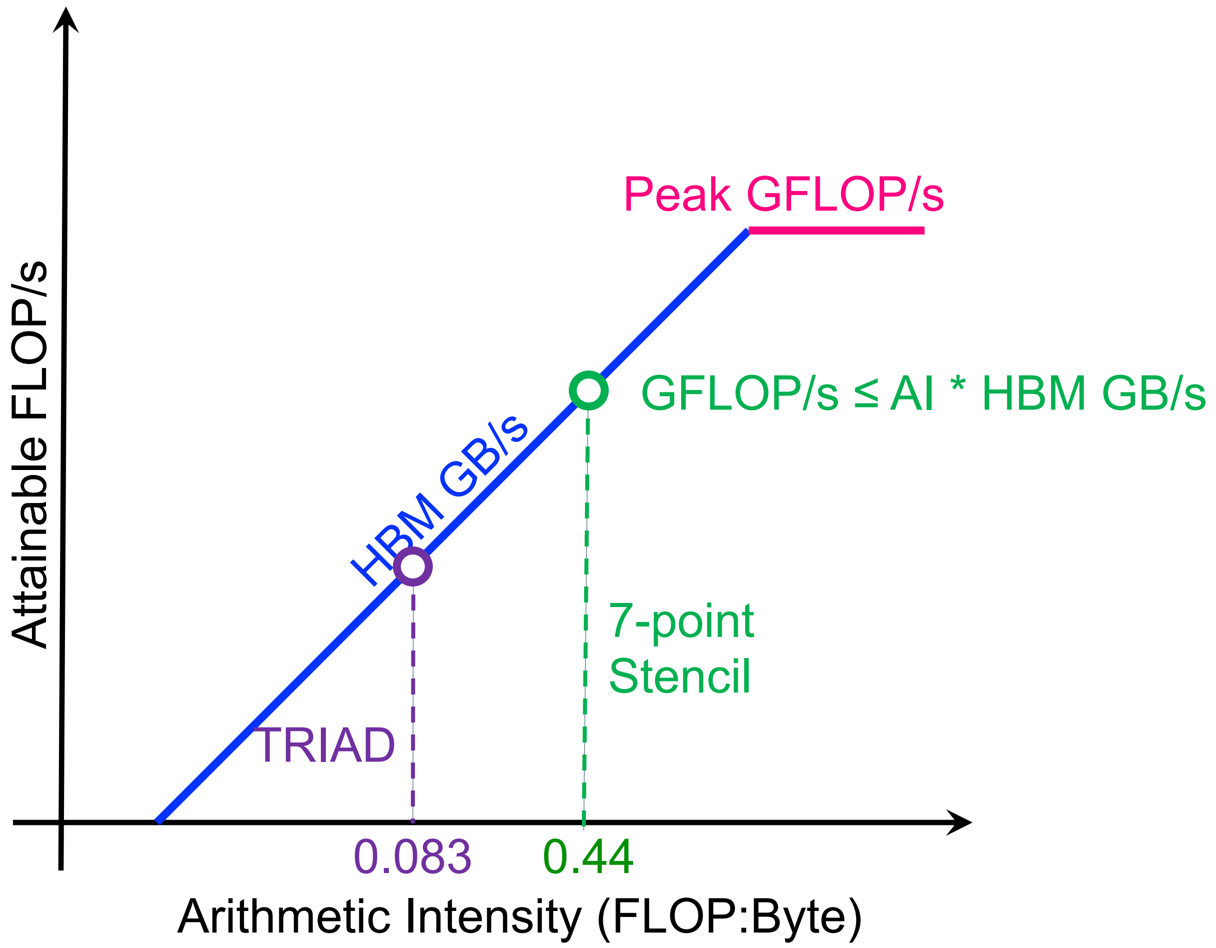

BERKELEY LAB

Roofline Example

- Consider a 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - Ideally, cache will filter all but 1 read and 1 write per point
  - **7 / (8+8) = 0.44 FLOPs per byte (DRAM)**

    == memory bound, but 5x the FLOP rate as TRIAD

```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                    + old[k  ][j  ][i-1]
                    + old[k  ][j  ][i+1]
                    + old[k  ][j-1][i  ]
                    + old[k  ][j+1][i  ]
                    + old[k-1][j  ][i  ]
                    + old[k+1][j  ][i  ];
}}}
```



Why We Use Roofline…

1. Determine when we're done optimizing code
   - Assess performance relative to machine capabilities
   - Track progress towards optimality
   - Motivate need for algorithmic changes

2. Identify performance bottlenecks & motivate software optimizations

3. Understand performance differences between Architectures, Programming Models, implementations, etc…
   - Why do some Architectures/Implementations move more data than others?
   - Why do some compilers outperform others?

4. Predict performance on future machines / architectures
   - Set realistic performance expectations
   - Drive for Architecture-Computer Science-Applied Math Co-Design