## The Roofine 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...

# getting our money's worth?







## Getting our money's worth?

- Really a question of good performance on applications benchmarks
- Imagine profiling a mix of GPUaccelerated benchmarks ...
- GFLOP/s alone may not be particularly insightful





- 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.







- 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<sup>6</sup>x slowdowns.
  - Provide no insights into algorithm or implementation.
  - Provide no guidance to CS, AM, applications, or procurement.

	_					
	Fron	it End	Instru	uction	1110	struct
		_		Cacha		22 VID
			Ta	aq	-	
				-	1	
	Dur	a ala			-	LO Dyle
	Brad	ncn lister			Instructi	on Fetch
	(BE					(16 B w
		0,		M	<mark>DP MO</mark> P	M <mark>O</mark> P
					Instruc	tion Oue
					(50, 2x	25 entrie
				M		P MO
		MicroC	ode			5-Way D
		Sequer	ncer	Compl	ex Simn	le Simi
		(MS RO	1 DM)	Decod	er Decod	er Deco
				1-4 µ0	Ps µ <mark>O</mark> P	μΟ
			4	μOPs		_
	Decoded Strea	m Buffer (DSB)				
	(µOP )	Cache)	6 μOPs			
	(1.5k µ0r (64 B v	vindow)				MU
				op Stre	am Alloc	ation Quei
			10	eccor to		-
	Register A	lias Table (RAT	) 2+9/10	μορ	μορ	μορ μ
				<b>`</b>		
-	Load		· · · · ·	-		
_		Maria Elimina	Ren	ame /	Allocate	/ Retire
		Move Elimina	ation Ren	name / leOrdei	' Allocate r Buffer (2	/ Retire 24 entrie
	FP	Move Elimina	ation Rer F	name / leOrdei 1 <mark>0</mark> 9	Allocate r Buffer (2 µ <mark>0</mark> P	/ Retire 24 entrie µ0
		Move Elimina	ation Rer F	ame / eOrdei	( Allocate r Buffer (2 μ <mark>ο</mark> Ρ	/ Retire 24 entrie µ0
		Move Elimina <b>µOP</b> Iteger Physical	ation Rer F P Register File	name / teOrder	(Allocate r Buffer (2 <b>µ0</b> P	/ Retire 24 entrie po Schedu
	FP FP Int Int	Move Elimina <b>µOP</b> Iteger Physical (180 Regis	Register File sters)	name / leOrder	( Allocate r Buffer (2 <b>µ0P</b> Unified Re	/ Retirer 24 entrie po Schedu servatior (97 entri
	PP PP Pr Pr Pr Pr Pr Pr Pr Pr	Move Elimina µOP teger Physical (180 Regis	Register File	name / leOrder	(Allocate r Buffer (2 µ0P Unified Re	/ Retirer 24 entrie po Schedu servatior (97 entri Port 6
	PP Pr Pr Pr Pr Pr Pr Pr Pr Pr	Move Elimina µOP teger Physical (180 Regist Port 1 µOP	ation Rer pop Register File sters)	pame / leOrder	(Allocate r Buffer (2 pop Unified Re	/ Retirer 24 entrie po Schedu servation (97 entri Port 6 port 6
	PP Int start Store Port 0 µOP	Move Elimina HOP iteger Physical (180 Regis Port 1 HOP	Register File	ame / eOrder μOP	(Allocate r Buffer (2 µ0P Unified Re	/ Retire 24 entrie po Schedu servatior (97 entri Port 6 port 6
		Move Elimina HOP Heger Physical (180 Registress Port 1 HOP U INT ALU INT ALU	ation Rer pop	Port 5	Allocate r Buffer (2 µop Unified Re	/ Retire: 24 entrie 24 entrie 5 chedu servatior (97 entri 97 entri 107 entrie 107 entrie 107 entrie
	PP Port o Int Lint Port o INT DD INT DD INT Vert	Move Elimina HOP Heger Physical (180 Registress Port 1 HOP U INT ALLY ALLY INT Vect A	Register File sters)	Port 5 μOP	Allocate r Buffer (2 µ0P	/ Retire: 24 entrie 24 entrie 5 chedu servatior (97 entri (97 entri))))))))))))))))))))))))))))))))))))
	PP Prot 0 μ0P INT ALL INT OP INT Vect.	Move Elimina pop iteger Physical (180 Regin Port 1 pop U INT ALU INT Vect A MUL INT Vect A	Register File sters)	Port 5 port 5 port 5 port 1 INT ALL LEA	(Allocate r Buffer (2 µ0P Unified Re	/ Retire: 24 entrie μ0 Schedu servatior (97 entr (97 entr (97 entr (97 entr μ0Ρ
	PP Int Ir Barry Port 0 μ0P INT ALL INT PORT INT ALL INT Vect I PP FM	Move Elimina µOP deger Physical (180 Regit µOP U INT ALU INT Vect A INT Vect M IP FMA	Register File sters)	Port 5 Port 5 µOP INT ALL LEA ALU IN MUL IN A	(Allocate r Buffer (2 µ0P Unified Re	/ Retire: 24 entrie 40 Schedu servatior (97 entr Port 6 40 Branch
	PP P P P P P P P P P P P P	Move Elimina Move Elimina deger Physical (180 Regis Port 1 µ0P U INT ALU INT ALU INT Vect A ALU INT Vect A FP FMA 122b fused Bit Scan	Accession Register File Register File INT Vect	Port 5 µOP INT ALL LEA ALU LEA ALU LEA Coptional	Allocate r Buffer (2 μορ Unified Re Unified Re Unified Re Unified Re	/ Retire: 24 entrie 40 Schedu servatior (97 entr Port 6 40 Branch
	Port 0 Port 0	Move Elimina Move Elimina deger Physical (180 Regis Port 1 µOP U INT ALU INT Vect A ALUI INT Vect M PP FMA D12b fused Bit Scan	Register File sters)	Port 5 port 5	Allocate r Buffer (2 μορ Unified Re Unified Re Unified Re Unified Re	/ Retire: 24 entrie 40 Schedu servation (97 entr Port 6 40 Branch
	Port 0 Int all Port 0 Port 0 INT ALD INT Vect 1 PF PM Vect Str FP DD	Move Elimina pop teger Physical (180 Regis Port 1 pop U INT ALU INT Vect A HUL INT Vect A EP FMA Bit Scan	Register File ters)	Port 5 PORT 5	/ Allocate r Buffer (2 unified Re J J Vect ALU FP FMA only)	/ Retire: 24 entrie 40 Schedu servation (97 entr Port 6 #0P
	Port 0 Port 0	Move Elimina pop teger Physical (180 Regis Port 1 pop U INT ALU INT Vect A UL INT Vect A UL INT Vect A INT Vect A Bit Scan	Register File sters)	Port 5 μOP INT ALL LEA	/ Allocate r Buffer (2 unified Re J J Vect ALU FP FMA only)	/ Retire: 24 entrie 40 Schedu servatior (97 entr 97 entr 97 entr 09 INT ALL Brance
	INT Vect. INT Vect. Vect Str FP DD. Brance	Move Elimina HOP Heger Physical (180 Regis Port 1 HOP INT ALU INT Vect A AUL INT Vect A HUL INT Vect A Bit Scan Ing Scution I	Register File tters) INT Vect INT Vect Stars Sta	and P and And P and And P and And P and And And P and And And And And And And And And And A	(Allocate Puffer (2 µ0P Unified Re	/ Retireu 24 entrie 40 Schedu servatior (97 entr 97 entr 97 entr 107 ALL Branct
	Port 0 Port 0	Move Elimina pop iteger Physical (180 Regis Port 1 pop U INT ALU INT MUL ALU INT Vect M FP FMA Bit Scan ing Scution I	Register File sters)	and P and And P and And P and And P and And And P and And And And And And And And And And A	/ Allocate r Buffer (2 µOP Unified Re	/ Retireu 24 entrie 24 entrie 5chedu servatior (97 entr 07 entr 07 entr 07 entr
	Port of port of por	Move Elimina HOP Ateger Physical (180 Regit Port 1 HOP U INT ALU INT ALU INT Vect A A PP FMA DISD Forsed Bit Scan (190 Regit Comparison of the second Bit Scan Comparison of the second Scattering (190 Regit) Comparison of the second Dist Scan Comparison of the second Dist Scan Dist Sca	Register File	ACP	(Allocate Puffer (2 prop Unified Re	/ Retireu 24 entrie 24 entrie 5chedu servatior (97 entr Port 6 00 INT ALL Serand
	Port of port of por	Move Elimina HOP HOP HEGER Physical (180 Regis Port 1 HOP U INT ALU INT ALU INT Vect A ALU INT Vect A FP FMA Station I	Register File ters)	ACP	/ Allocate Puffer (2 unified Re	/ Retireu 24 entrie 24 entrie 24 entrie 35 schedu Schedu Servation (97 entri 07 entri 07 entri 07 entri
	Port 0 µ0P INT ALL INT OP INT Vect I PFP FMU FPP FM	Move Elimina Move Elimina (180 Regit Port 1 Port 1	Register File sters)	Port 5 µOP INT ALL LEA ALLU IN LOP DO INT ALL LEA ALLU IN LOP DO INT ALL LEA ALLU IN LOP DO INT ALL LEA ALLU IN DO INT ALL LEA ALLO INT ALL ALLO INT ALLO INTA ALLO INTE ALLO	/ Allocate Puffer (2 unified Re	/ Retireu 24 entrie 24 entrie 24 entrie Schedu Schedu Servation (97 entri Port 6 µOP
	Port 0 Port 0	Move Elimina Move Elimina (180 Regit Port 1 Port 1	Register File ters)	Port 5 µOP INT ALL LEA ALLUI IN b (zmm (optional	(Allocate Fuffer (2 Unified Re	/ Retire: 24 entrie 24 entrie 24 entrie 25 entrie 26 entrie 27 entrie 27 entrie 28 entrie 29 entrie 29 entrie 29 entrie 20 ent
	PP Proto Port 0 Port 0 Por	Move Elimina Move Elimina (180 Regis Port 1 POP U INT ALU INT MUL INT Vect A A PP FMA S12b fused Bit Scan	Register File ters)	Port 5 µOP ITTALU IN ITTALU IN LEA ALLI IN IN IL IN IL IN IL IL	(Allocate Fuffer (2 µOP Unified Re	/ Retire: 24 entrie 24 entrie 24 entrie 25 entrie 26 entrie 27 entrie 27 entrie 28 entrie 29 entrie 29 entrie 29 entrie 29 entrie 20 ent





- We could take a CS approach and look at performance counters...
  - Record microarchitectural events on CPUs/GPUs
  - Use arcane architecture-specific terminology Ο
  - May be broken Ο
  - We may be able to show correlation Ο between events, but...
  - Improviding 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



## 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 Ο
  - Readily understandable by the extremely broad DOE community Ο
  - intuition, insights, and guidance to CS, AM, apps, procurement, and vendors Ο

### Roofline is just such a model



- Which takes longer?
  - o Data Movement
  - Compute?



## Time = max { #FP ops / Peak GFLOP/s #Bytes / Peak GB/s



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



Time<br/>#FP ops= max1 / Peak GFLOP/s<br/>#Bytes / #FP ops / Peak GB/s



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



#FP ops<br/>Time= min {Peak GFLOP/s<br/>(#FP ops / #Bytes) \* Peak GB/s



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



## GFLOP/s = min { AI \* Peak GB/s

Arithmetic Intensity (AI) = measure of data locality



## (DRAM) Roofline Model

### GFLOP/s = min { Peak GFLOP/s AI \* Peak GB/s

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

## Peak GFLOP/s AI \* Peak GB/s GFLOP/s = min

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

Roofline tessellates the localityperformance plane into five regions...



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



 Think back to our mix of benchmarks...







• We can sort benchmarks by arithmetic intensity...





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





Benchmarks near the roofline are making good use of computational resources



50% of Peak



## **General Performance Optimization Strategy**

Get to the Roofline





## **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

- $\circ$   $\,$  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...

of

D (c32) (77)

•ADD (c1) (9.2)

5.00

BEST PAPER,

50.00

GPU

### **DRAM's not the** bottleneck...

 Cache bandwidth and cache locality • PCle bandwidth

### ... The Hierarchical **Roofline Model**



### Lack of Parallelism...

... Roofline Scaling

Analysis

0.50

Arithmetic Intensity (Flops/Byte)

- Idle Cores/SMs
- Insufficient ILP/TLP
- Divergence and Predication

**Trajectories** 

GFlop/s 10.0

2

5

0.01

0.05

### **Machine Learning Applications...**

• Mixed Precision • Not enough Tensor Core

OPs

### Integer-heavy Codes... ○ Non-FP inst. impede **FLOPs** • No FP instructions

## ... Additional Ceilings

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





22

### ... The Instruction **Roofline Model**

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





## **Prototypes: Using Roofline for Applications**

Arithmetic Intensity (FLOP/Byte)

23

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

### Roofline at NERSC/LBL<sup>1</sup>

- KNL (Cori)
- Compared against Haswell (Xeon)
- KNL optimization/readiness as a performance trendline



Arithmetic Intensity (FLOP/Byte

### **Roofline used at ALCF<sup>2</sup>**

- OpenCL and SYCL(shown)
- RAJA LCALS benchmarks
- o Intel Gen9 GT4e (left) and NVIDIA **V100 (right)**







### **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)**



Roofline					
5.6					
93,493,394,664.2					
0.0 sity [FLOP/byte]	1,00	0.0	10,0	00.0	



## **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 Ο can do extra FLOPs if they move less data
  - those in the "Bytes are free" region can move Ο extra data if it saves them on FLOPs
- Motivates and provides quantitative analysis for the potential for...
  - Alternate data structures/representations Ο
  - re/precomputation of data Ο
  - communication-avoiding algorithms Ο
  - alternate and high order numerical methods Ο





## Summary



212



## 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 Roofine 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





# Acknowled genents

- 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-AC05-000R22725.







LAWRENCE BERKELEY NATIONAL LABORATORY BERKELEY LAB



### **Arithmetic Intensity**

- Measure of data locality (data reuse)
- Ratio of <u>Total Flops</u> performed to <u>Total Bytes</u> moved
- For the DRAM Roofline...
  - Total Bytes to/from DRAM
  - $\circ$   $\,$  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 Ο (**Benchmarking**)
- Unique to each architecture Ο
- Common to all apps on that architecture Ο





### **Roofline is made of two components**

### Machine Model

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

### **Application Characteristics**

- Dots defined by application GFLOP's and Ο GB's (Application Instrumentation)
- Unique to each application Ο
- Unique to each architecture Ο





### 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



50% of Peak



### 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 <u>high performance</u> (GFLOP/s), but still make poor use of a machine (%peak)





### 37

### **Roofline Example**

Consider a 7-point constant coefficient stencil...







### 38

## **Roofline Example**

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







## **Roofline Example**

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







## **Roofline Example**

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

<pre>#pragma omp parallel for for(k=1;k<dim+1;k++){ for(i_1+i_dim+1+i_+)}<="" pre=""></dim+1;k++){></pre>
TOP(J=1; J<01m+1; J++){
for(1=1;1 <d1m+1;1++){< td=""></d1m+1;1++){<>
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 ];
}}}

	Com	p
P	erfect	C
ŧ	ţ	
	HE	31





### 41

## **Roofline Example**

- Consider a 7-point constant coefficient stencil...
  - 7 FLOPs Ο
  - 8 memory references (7 reads, 1 store) per point Ο
  - Ideally, cache will filter all but 1 read and 1 write per point 0
  - 7 / (8+8) = 0.44 FLOPs per byte (DRAM)  $\succ$

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

<pre>#pragma omp parallel for</pre>
<pre>for(k=1;k<dim+1;k++){< pre=""></dim+1;k++){<></pre>
for(j=1;j <dim+1;j++){< td=""></dim+1;j++){<>
<pre>for(i=1;i<dim+1;i++){< pre=""></dim+1;i++){<></pre>
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 ];
}}}



### Peak GFLOP/s GFLOP/s ≤ AI \* HBM GB/s

7-point Stencil



## Why We Use Roofline...

### 1. Determine when we're done optimizing code

- Assess performance relative to machine capabilities Ο
- Track progress towards optimality Ο
- Motivate need for algorithmic changes Ο
- 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? Ο
  - Why do some compilers outperform others? Ο

### 4. Predict performance on future machines / architectures

- Set realistic performance expectations 0
- Drive for Architecture-Computer Science-Applied Math Co-Design Ο

