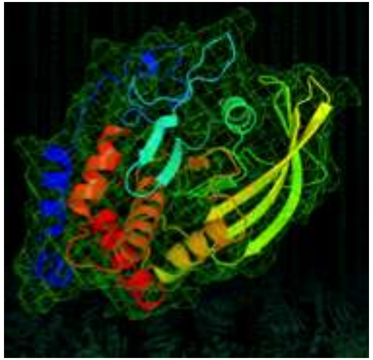
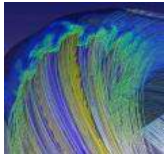
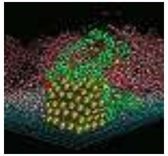
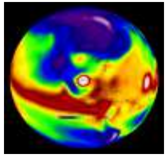
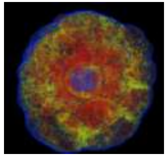
Introduction to the Roofline Model

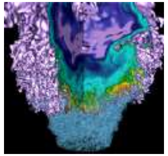










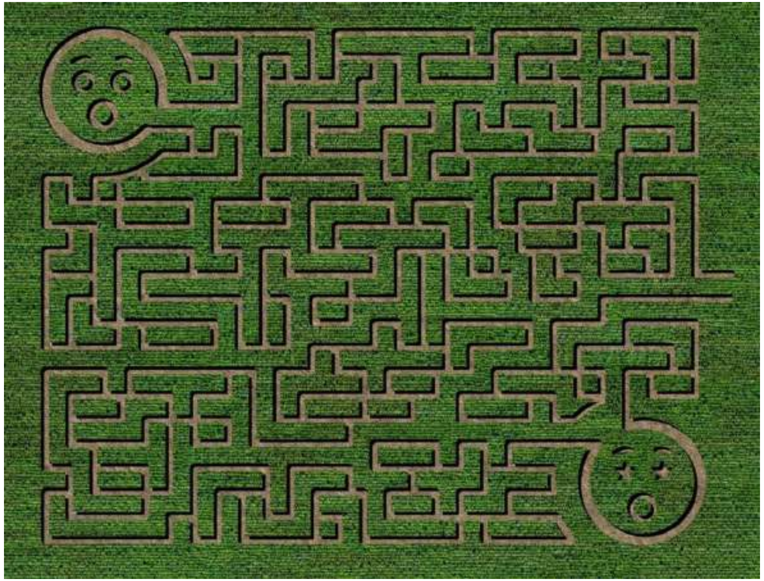


Charlene Yang

Lawrence Berkeley National Laboratory

Jun 16 2019, Frankfurt

Performance Models

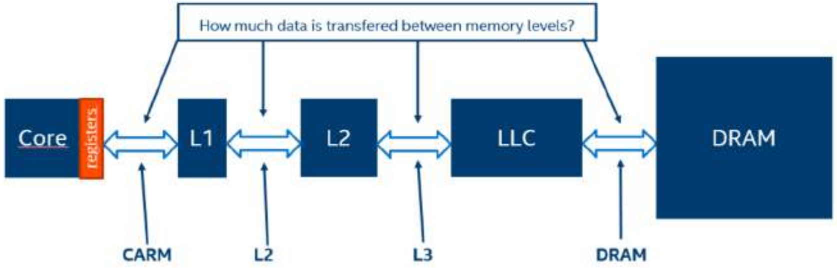


The Maze of Performance Optimization

The Map !!!

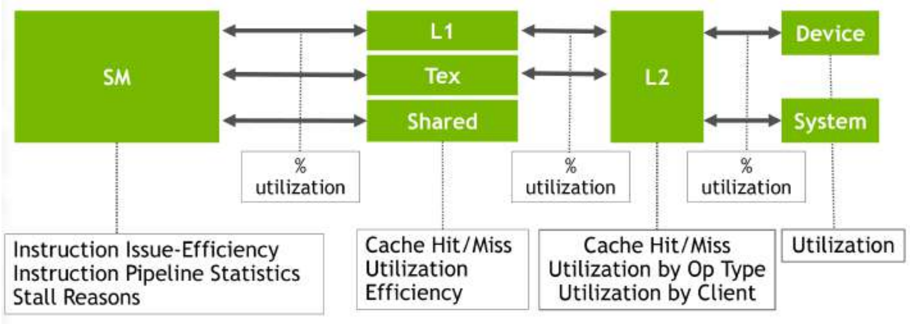
Performance Models

Modern architectures are complicated!



Intel Haswell CPU[1]

NVIDIA Volta GPU[2]



1. https://software.intel.com/en-us/articles/integrated-roofline-model-with-intel-advisor
2. http://on-demand.gputechconf.com/gtc/2016/presentation/s6659-avinash-baliga-perfworks.pdf

Performance Models

- **Many components contribute to the kernel run time**
- **An interplay of application characteristics and machine characteristics**

Roofline Model

#FP operations	FLOP/s
Cache data movement	Cache GB/s
DRAM data movement	DRAM GB/s
PCIe data movement	PCIe bandwidth
MPI Message Size	Network Bandwidth
MPI Send:Wait ratio	Network Gap
#MPI Wait's	Network Latency
IO	File systems

Focus on one or two dominant components!

Roofline Performance Model

- **Sustainable performance is bound by**

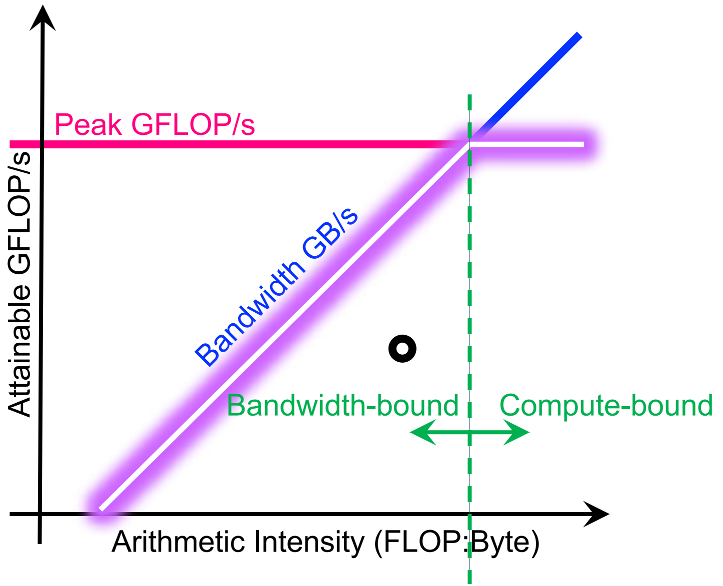
$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI} * \text{Peak GB/s} \end{cases}$$

- **Arithmetic Intensity (AI) =**

FLOPs / Bytes

- **How did this come about?**

  → **A CPU DRAM example**

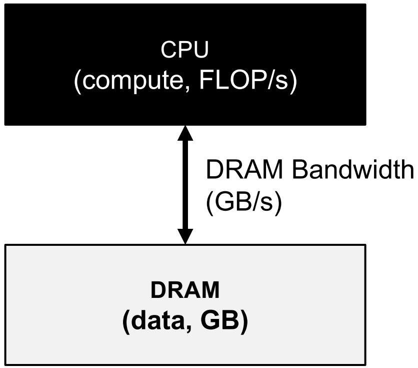


Transition @ AI ==
Peak GFLOP/s / Peak GB/s ==
'Machine Balance'

(CPU DRAM) Roofline

- One could hope to always attain peak performance (FLOP/s)
- However, finite locality (reuse) and bandwidth limit performance.
- Assume:
  - Idealized processor/caches
  - Cold start (data in DRAM)

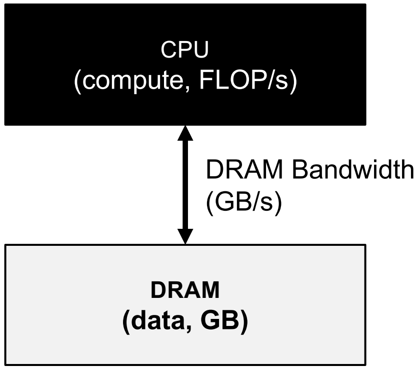
$$\text{Time} = \max \begin{cases} \text{\#FP ops / Peak GFLOP/s} \\ \text{\#Bytes / Peak GB/s} \end{cases}$$



(CPU DRAM) Roofline

- One could hope to always attain peak performance (FLOP/s)
- However, finite locality (reuse) and bandwidth limit performance.
- Assume:
  - Idealized processor/caches
  - Cold start (data in DRAM)

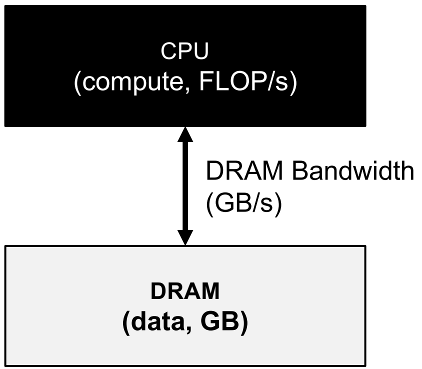
$$\frac{\text{Time}}{\text{\#FP ops}} = \max \begin{cases} 1 \,/\, \text{Peak GFLOP/s} \\ \text{\#Bytes} \,/\, \text{\#FP ops} \,/\, \text{Peak GB/s} \end{cases}$$



(CPU DRAM) Roofline

- One could hope to always attain peak performance (FLOP/s)
- However, finite locality (reuse) and bandwidth limit performance.
- Assume:
  - Idealized processor/caches
  - Cold start (data in DRAM)

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

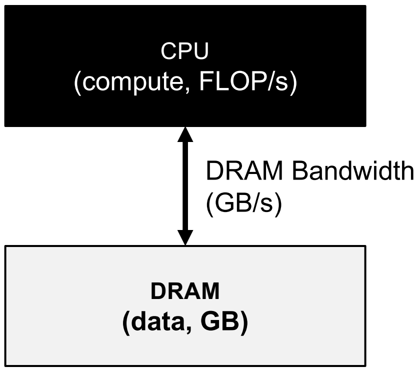


(CPU DRAM) Roofline

- One could hope to always attain peak performance (FLOP/s)
- However, finite locality (reuse) and bandwidth limit performance.
- Assume:
  - Idealized processor/caches
  - Cold start (data in DRAM)

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

Arithmetic Intensity (AI) = FLOPs / Bytes (as presented to DRAM)



Roofline Performance Model

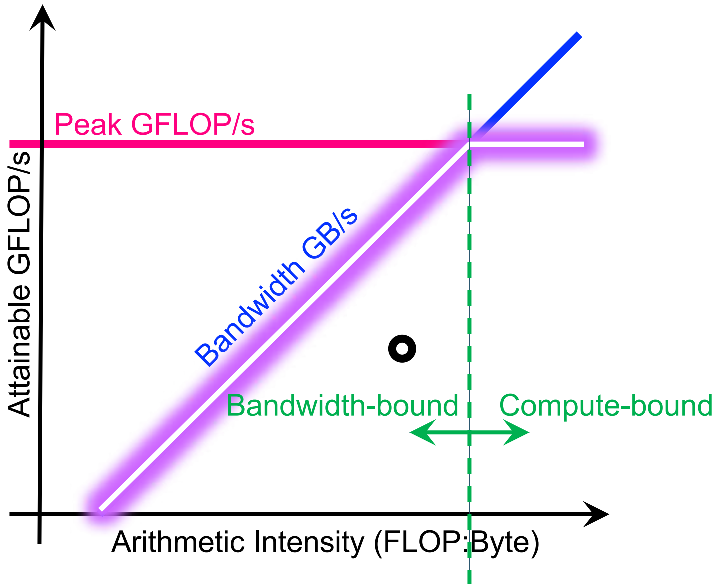
- Thus we obtain the model as

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

where Arithmetic Intensity (AI) is

FLOPs / Bytes

- Machine Balance (FLOPs/Byte) = **8.9** (V100, DP, HBM) or **5.1** (KNL, DP, HBM)



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

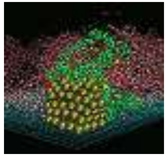
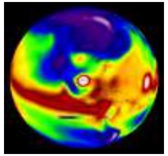
Roofline Performance Model

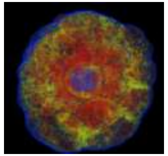
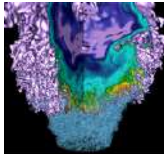
- A throughput-oriented model
  - tracks rates not times, i.e. GFLOP/s, GB/s, not seconds

- An abstraction over
  - architectures, ISA (CPU, GPU, Haswell, KNL, Pascal, Volta)
  - programming models, programming languages
  - numerical algorithms, problem sizes

- In log-log scale to easily extrapolate performance along Moore's Law

More Advanced on Roofline

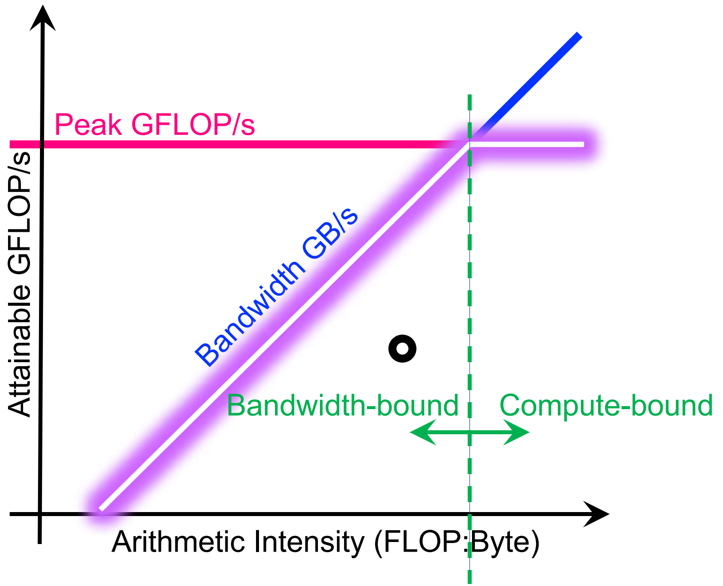



Roofline Performance Model

- **This is a single Roofline**

- **What about the memory hierarchy, different execution configurations, and instruction mixes?**

→ Hierarchical Roofline
→ Multiple compute ceilings



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

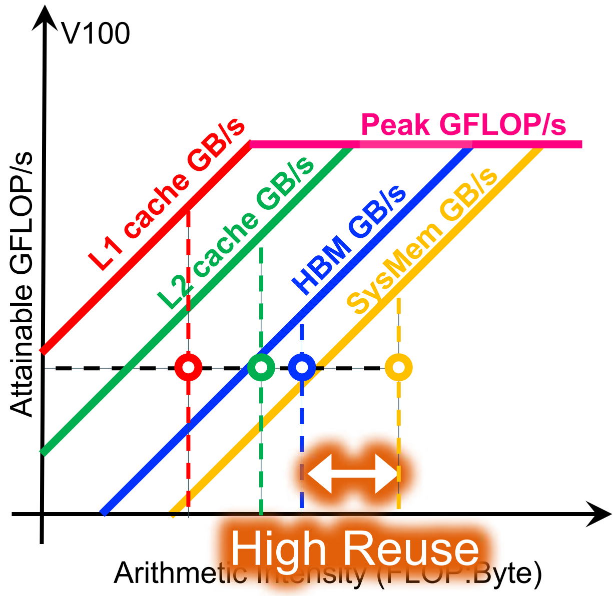
Hierarchical Roofline

- Superposition of multiple Rooflines
  - Incorporate full memory hierarchy
  - Arithmetic Intensity =

    FLOPs / Bytes$_{L1/L2/HBM/SysMem}$

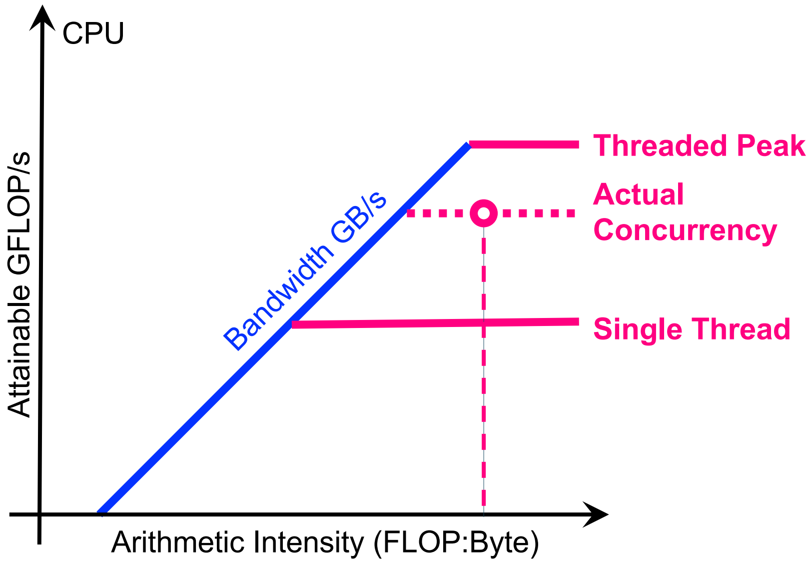
- Each kernel will have multiple AI's but one observed GFLOP/s performance

- Hierarchical Roofline tells you about **cache locality**



Multiple Compute Ceilings

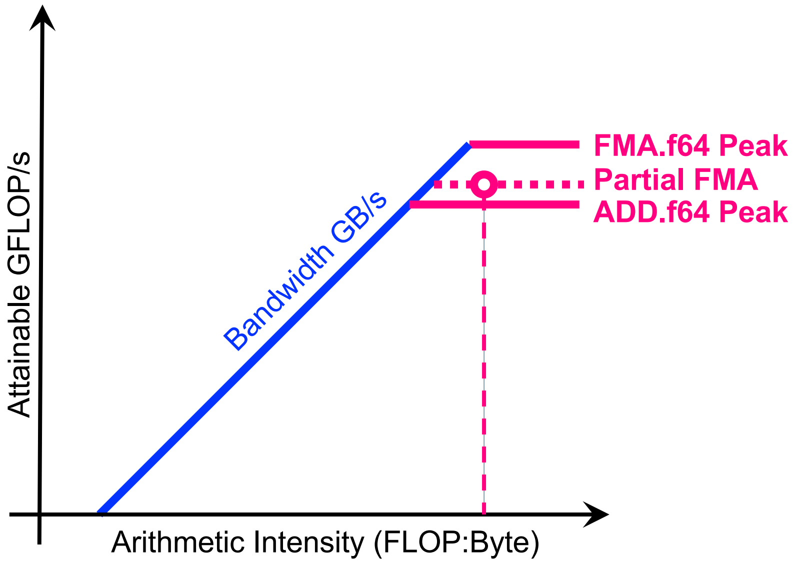
- Impact of **execution configuration**
- Concurrency affects your peak
  - OpenMP thread concurrency
  - SM occupancy
  - load balance
  - threadblock/thread configuration



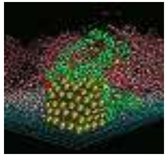
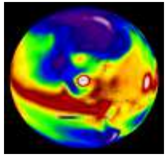
- Performance is bound by the **actual concurrency** ceiling

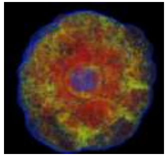
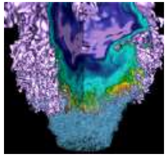
Multiple Compute Ceilings

- Impact of **instruction mix**
- Applications are usually a mix of FMA.f64, ADD.f64, MUL.f64…
- Performance is a **weighted** average … bound by a **partial FMA** ceiling



Roofline Drives Optimization

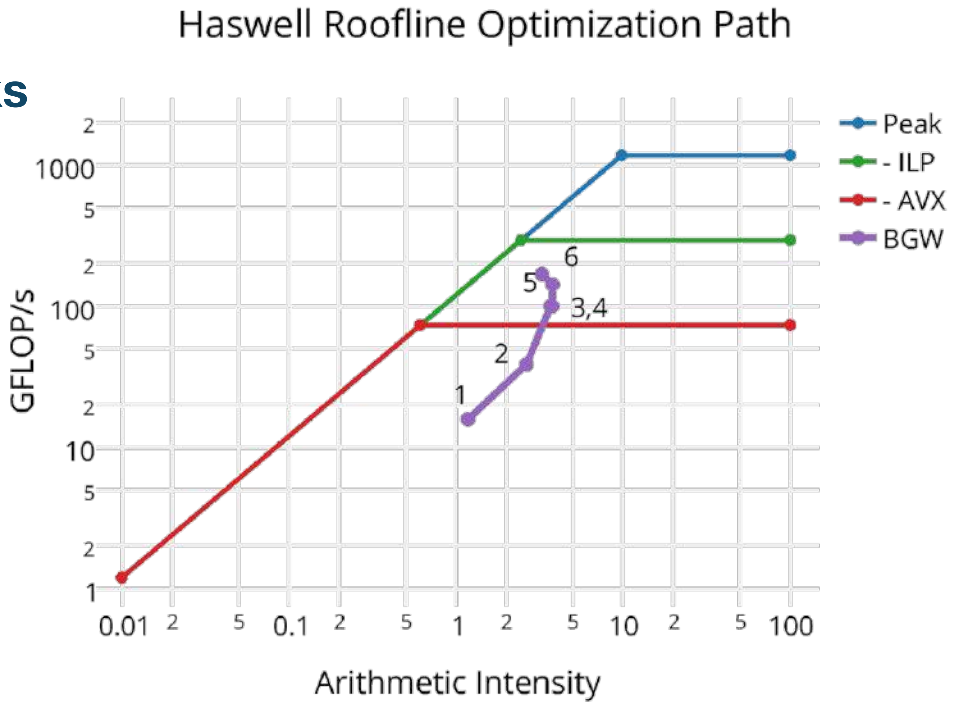
Roofline Performance Model

The Roofline Model

- **helps you identify the bottlenecks**
- **guides you through optimization**
- **tells you when to stop**

An example:

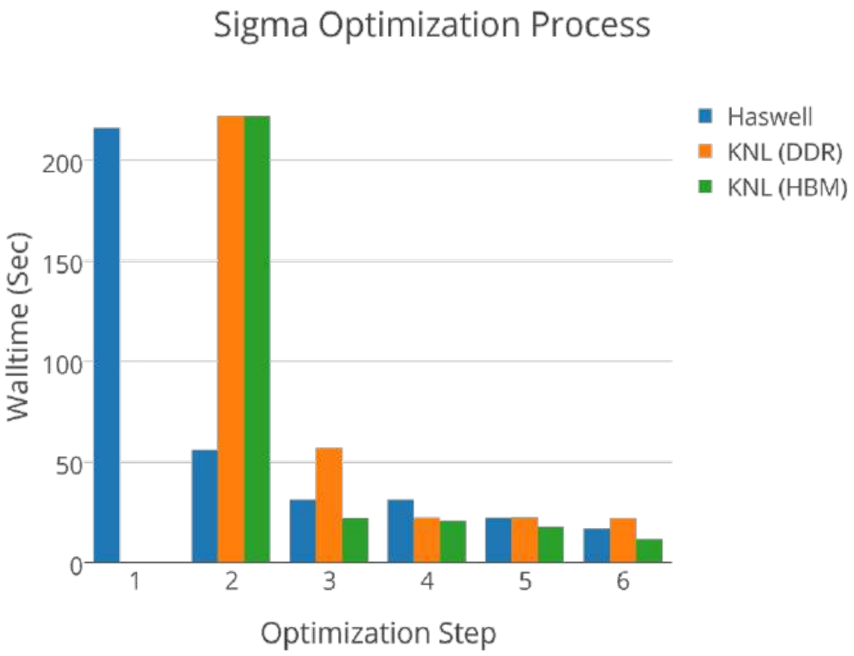
- **NESAP for Cori - BerkeleyGW**



Roofline Example: BerkeleyGW

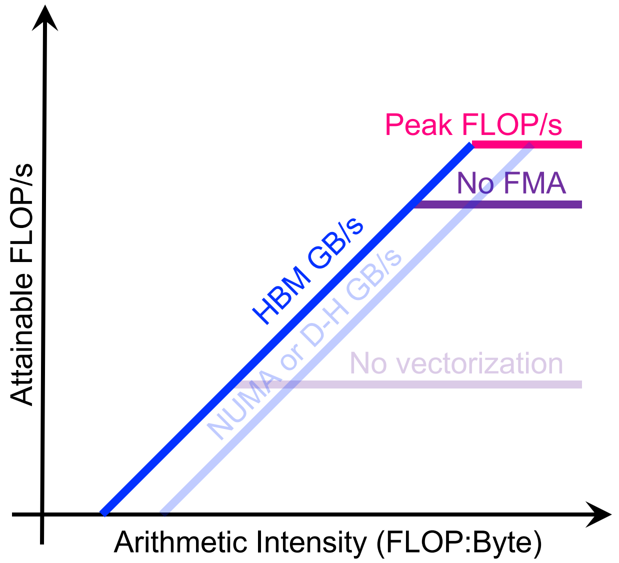
Optimization Path for Kernel-C (Sigma):

1. Add OpenMP
2. Initial Vectorization
   - loop reordering
   - conditional removal
3. Cache-Blocking
4. Improved Vectorization
   - divides
5. Hyper-threading



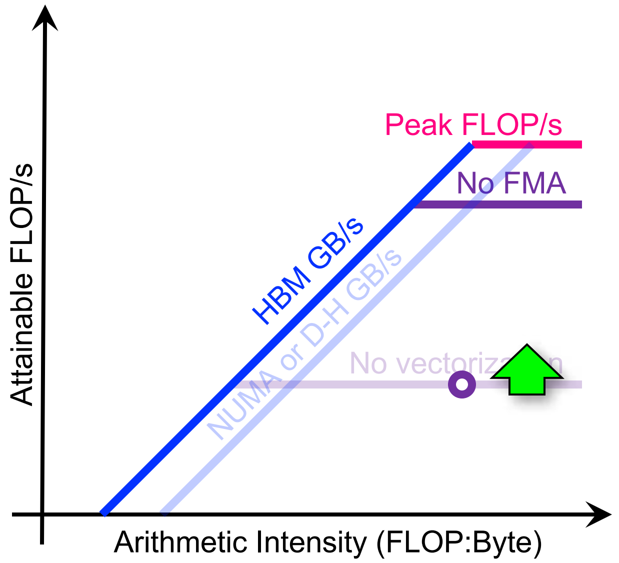
General Optimization Strategy

- **Broadly speaking, three approaches to improving performance:**



General Optimization Strategy

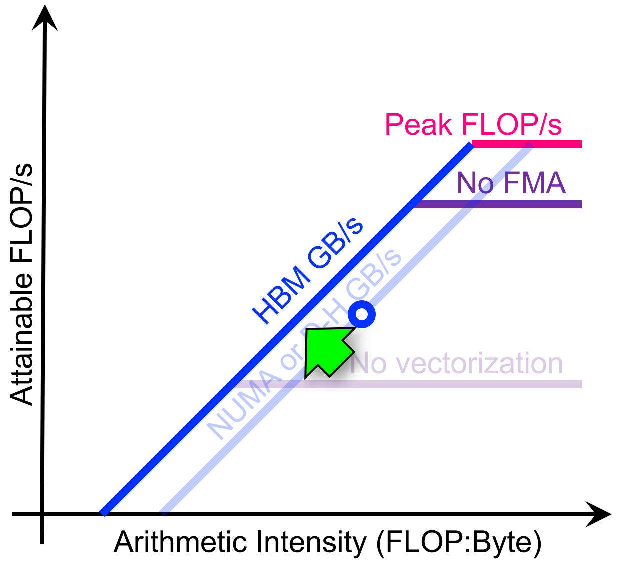
- Broadly speaking, three approaches to improving performance:
- Maximize compute performance
  - multithreading
  - vectorization
  - increase SM occupancy
  - utilize FMA instructions
  - minimize thread divergence



General Optimization Strategy

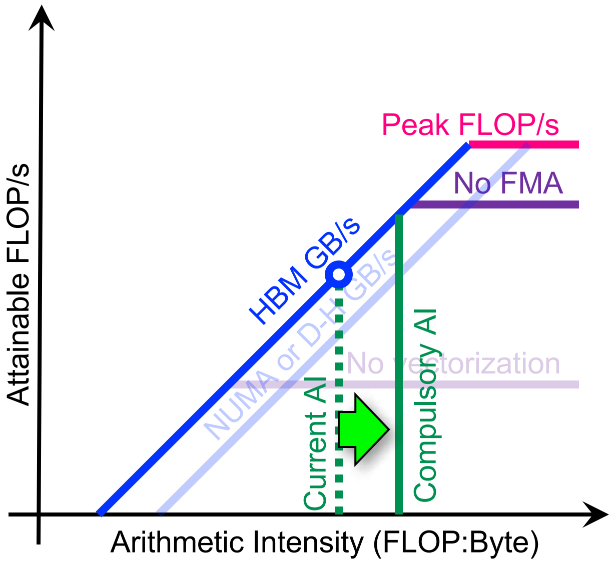
- Broadly speaking, three approaches to improving performance:
- Maximize compute performance

- Maximize memory bandwidth
  - utilize higher-level caches
  - NUMA-aware allocation
  - avoid H-D transfers
  - avoid uncoalesced memory access

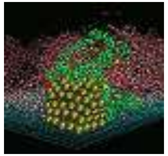
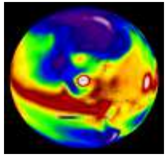


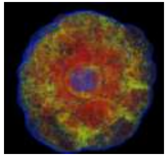
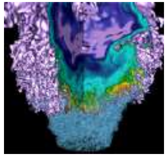
General Optimization Strategy

- **Broadly speaking, three approaches to improving performance:**
- **Maximize compute performance**
- **Maximize memory bandwidth**
- **Improve AI**
  - **minimize data movement**
  - **exploit cache reuse**



Roofline Data Collection

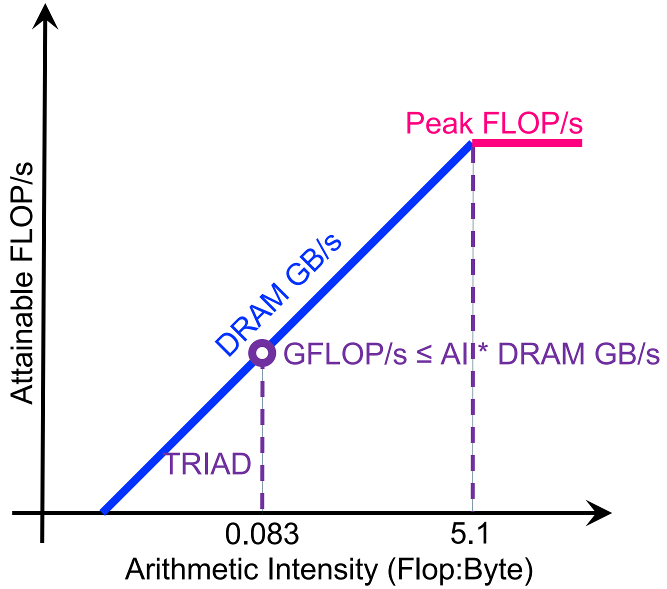
Pen and Paper

- **Example #1: STREAM Triad**

```
for(i=0;i<N;i++){
  Z[i] = X[i] + alpha*Y[i];
}
```

- **2 FLOPs per iteration**
- **Transfer 24 bytes per iteration**
  - **read X[i], Y[i], and write Z[i]**

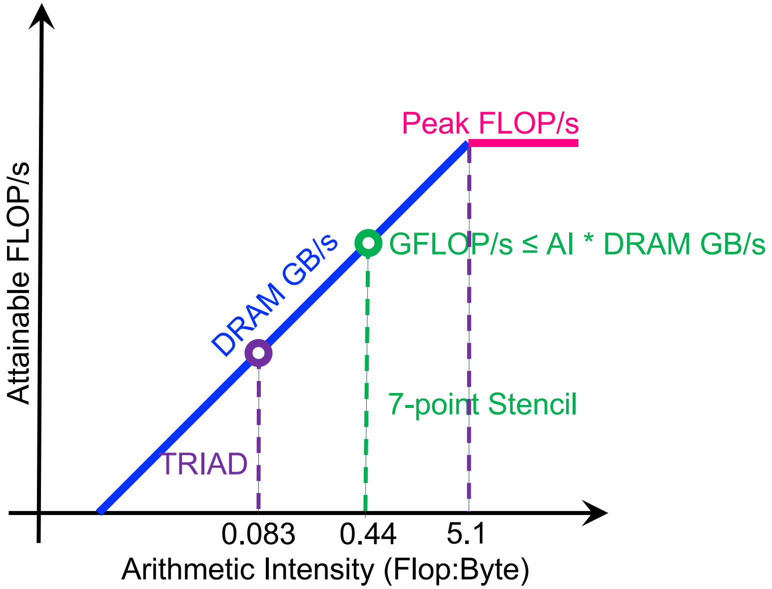
- **AI = 0.083 FLOPs per byte**
- **Memory bound**



Pen and Paper

- **Example #2: 7-pt stencil**
  - **7 FLOPs; 8 memory references (7 reads, 1 store) per pt**
  - **Cache can filter all but 1 read and 1 write per pt**
  - **AI = 0.44 FLOPs per byte**
  - **Memory bound, but 5x the GFLOP/s rate**

```
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  ];
}}}
```

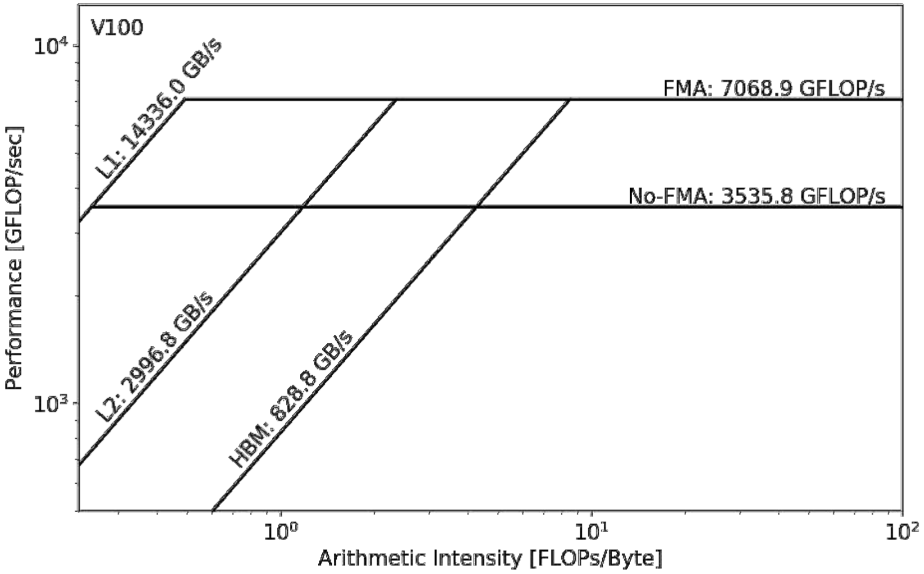


Pen and Paper

- Not scalable for real-life applications
- Millions of lines of code; mix of different languages
- Complicated modern architecture
  - memory hierarchy, caching effects
  - ISA
- Different problem sizes

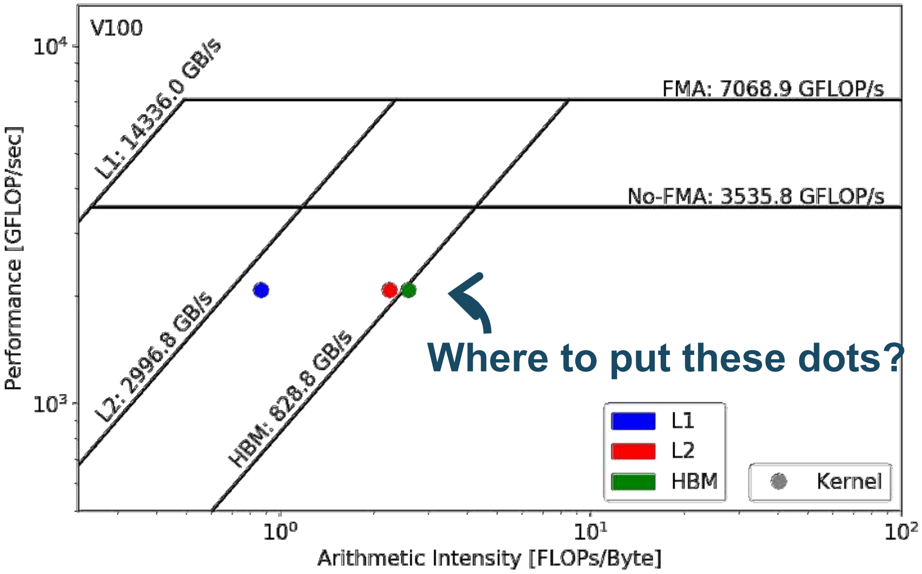
We need tools!

We Need Tools!



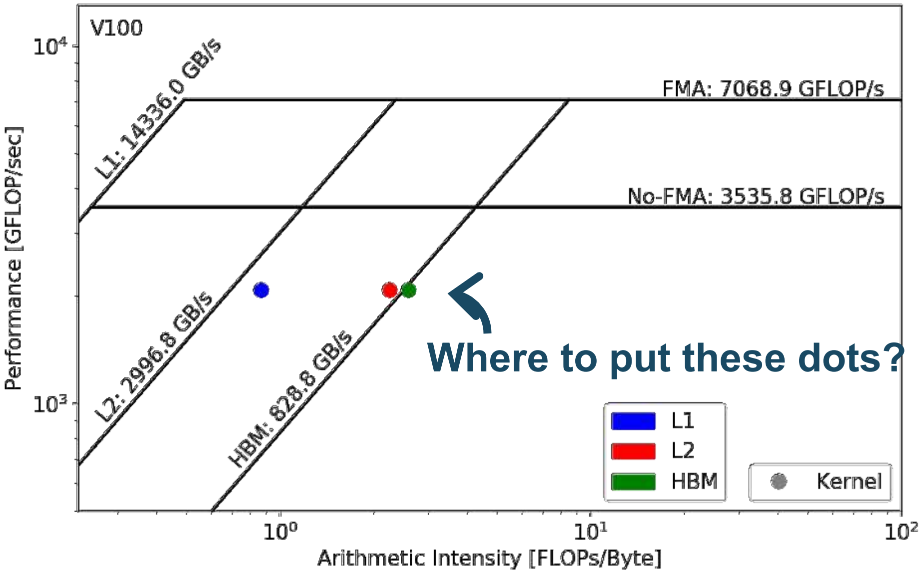
- **Roofline ceilings**
  - **vendor specifications**
  - **empirical measurements**
    - **ERT**
    - https://bitbucket.org/berkeleylab/cs-roofline-toolkit

We Need Tools!



Where to put these dots?

We Need Tools!



Where to put these dots?

Require three raw measurements:
- **Runtime**
- **FLOPs**
- **Bytes (on each cache level)**

In order to calculate AI and GFLOP/s:

$$\text{Arithmetic Intensity} = \frac{\text{FLOPs}}{\text{Data Movement}}$$

(FLOPs/Byte)

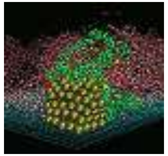
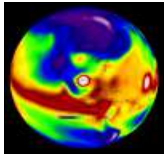
$$\text{Performance} = \frac{\text{FLOPs}}{\text{Runtime}}$$

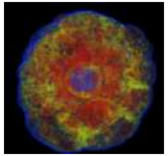
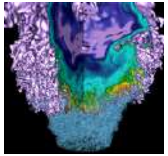
(GFLOP/s)

Methodology to Construct Roofline

1. **Collect Roofline ceilings**
   - **compute** (FMA/no FMA) and **bandwidth** (DRAM, L2, …)
   - ERT: https://bitbucket.org/berkeleylab/cs-roofline-toolkit
2. **Collect application performance**
   - **FLOPs**, **bytes** (DRAM, L2, …), **runtime**
   - SDE, VTune, LIKWID, Advisor, nvprof, …
3. **Plot Roofline with Python Matplotlib (or other tools of your preference)**
   - **arithmetic intensity**, **GFLOP/s** performance, **ceilings**
   - example scripts: https://github.com/cyanguwa/nersc-roofline

Automated Data Collection

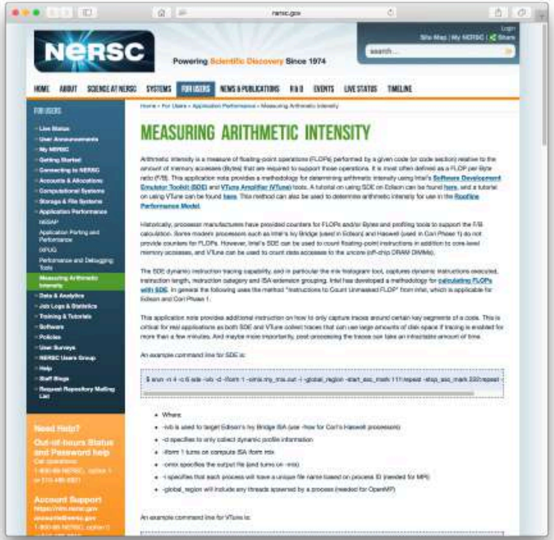



Data Collection on Intel CPUs

The not-so-automated way 1:

- Intel SDE for FLOPs (emulation)
- Intel VTune for DRAM bytes (HW counters)
- Runtime
- DRAM Roofline only
- Used by NESAP for Cori
  - NERSC Exascale Science Application Program
  - http://www.nersc.gov/users/application-performance/measuring-arithmetic-intensity/



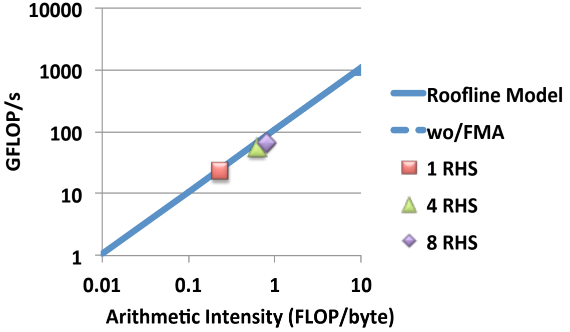
U.S. DEPARTMENT OF ENERGY | Office of Science

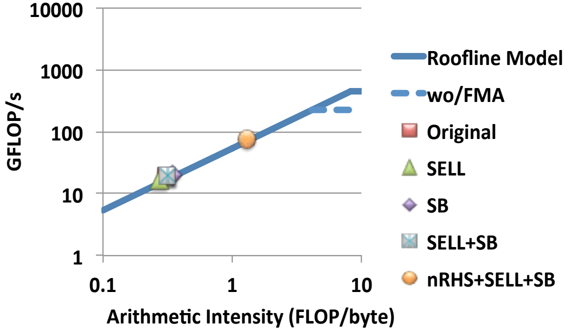
BERKELEY LAB

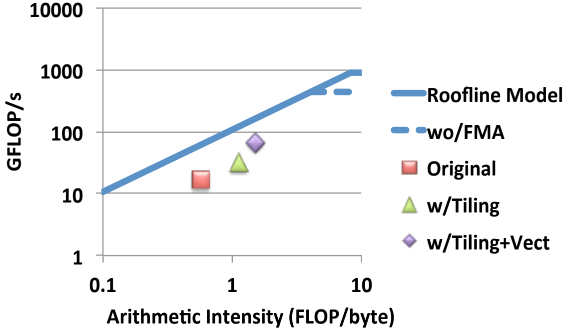
Data Collection on Intel CPUs

MFDn | EMGeo | PICSAR

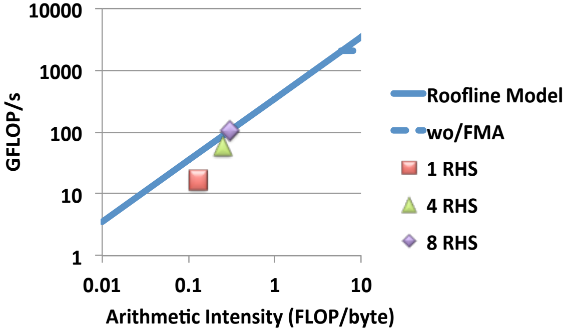
2P HSW

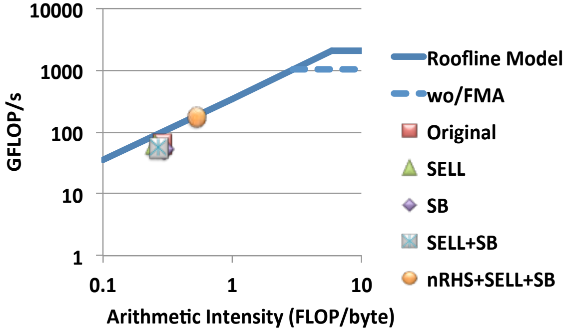


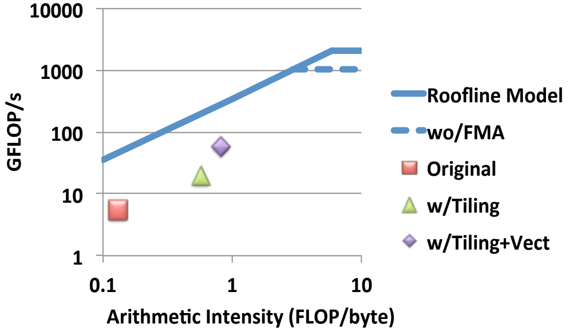




KNL







U.S. DEPARTMENT OF ENERGY | Office of Science

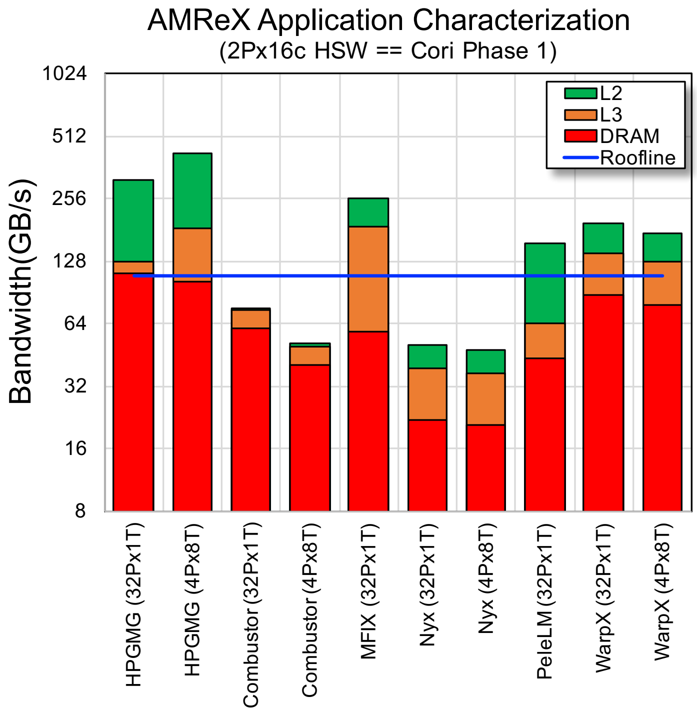
DRAM Rooflines of NESAP Codes

BERKELEY LAB

Data Collection on Intel CPUs

The not-so-automated way 2:

- **LIWID for FLOPs and bytes**
  - **Both are based on HW counters**
- **Runtime**
- **Hierarchical Roofline**
- **Limited by quality of HW counters**
- **High-level characterization, no callstack**

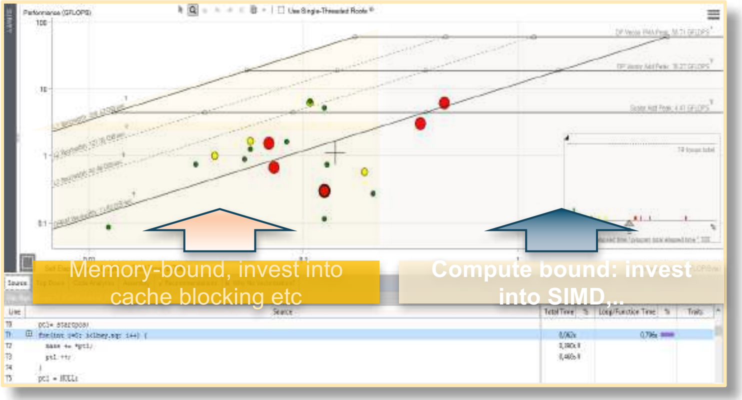


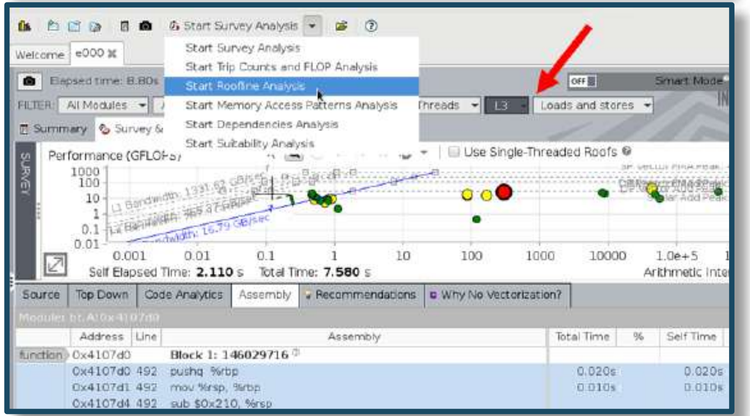
https://github.com/RRZE-HPC/likwid

Data Collection on Intel CPUs

The fully automated way:

- Intel Advisor, Roofline feature
- Instrument applications automatically
  - one dot per loop nest/function
- FLOPs, bytes and runtime
- Hierarchical Roofline
- Integrates with other Advisor capabilities
- Benchmarks target system





Data Collection on NVIDIA GPUs

- **Still very manual at this stage, but…**

- **Runtime:**
  - Internal timers or `nvprof --print-gpu-trace`
- **FLOPs:**
  - DP/SP/HP counters and metrics, `nvprof --metrics 'flop_count_dp/sp/hp'` or `` `tensor_precision_fu_utilization' ``
- **Bytes for different cache levels:**
  - Bytes = (read transactions + write transactions) x transaction size
  - `nvprof --metrics 'metric_name'` e.g. `gld/gst_transactions`

- **Hierarchical Roofline**

Summary

- The Roofline Model formulizes the interaction between machine characteristics and application characteristics, and guides optimization
  - Peak computational throughput and bandwidth
  - Arithmetic intensity, cache locality, instruction mix…

- Automate Roofline data collection
  - Intel CPUs
    - Intel SDE + Intel VTune, Intel Advisor
  - NVIDIA GPUs
    - nvprof, Nsight Compute

More in the next few talks!

Reference

- S. Williams, A. Waterman and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009
- LBNL CRD Roofline Research:

  https://crd.lbl.gov/departments/computer-science/PAR/research/roofline
- Empirical Roofline Toolkit (ERT):

  https://bitbucket.org/berkeleylab/cs-roofline-toolkit
- Python scripts for plotting manually-collected Roofline:

  https://github.com/cyanguwa/nersc-roofline/tree/master/Plotting

Acknowledgement

- 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.

Thank You