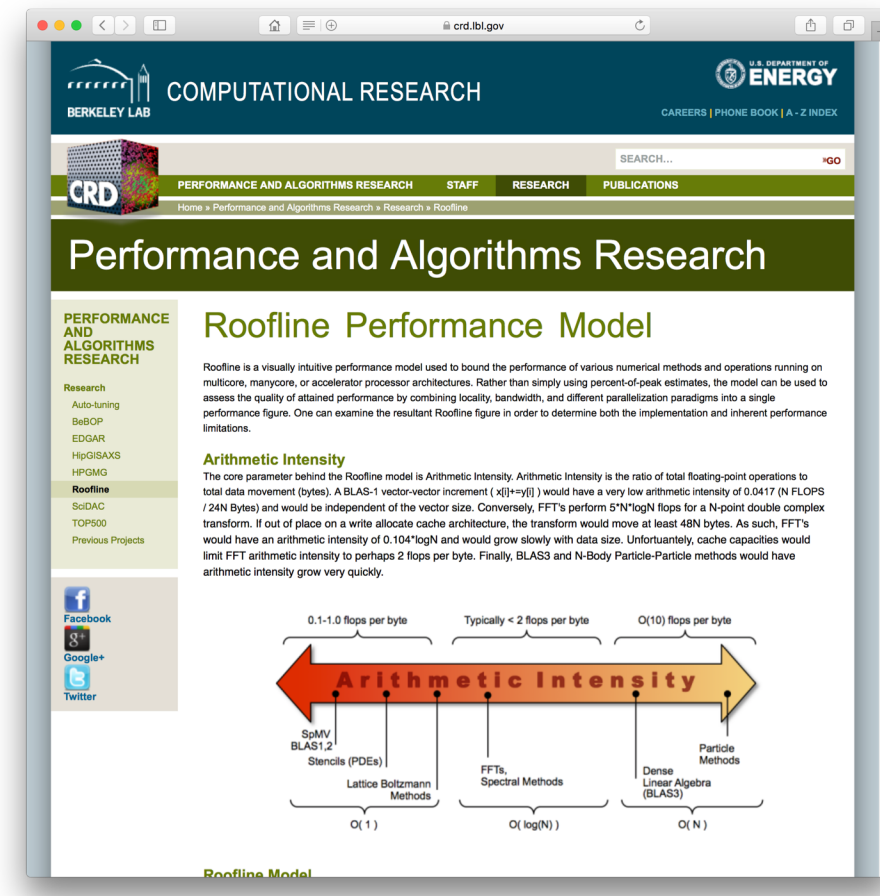


Performance Analysis using the Roofline Model

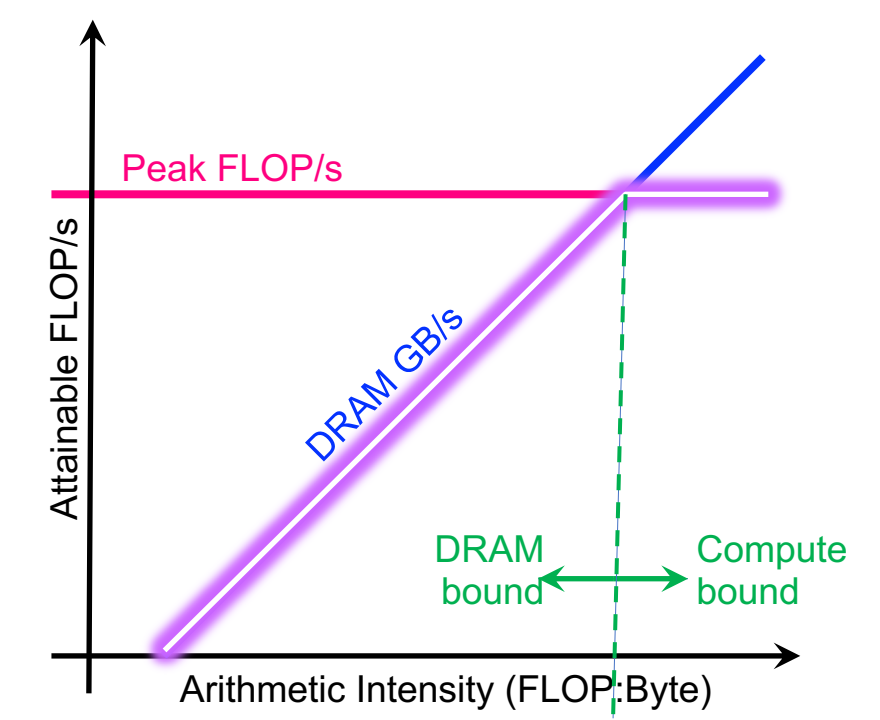
Samuel Williams (SWWilliams@lbl.gov), Charlene Yang, Khaled Ibrahim, Thorsten Kurth, Nan Ding, Jack Deslippe, Leonid Oliker
CRD/NERSC, Lawrence Berkeley National Laboratory

Introduction

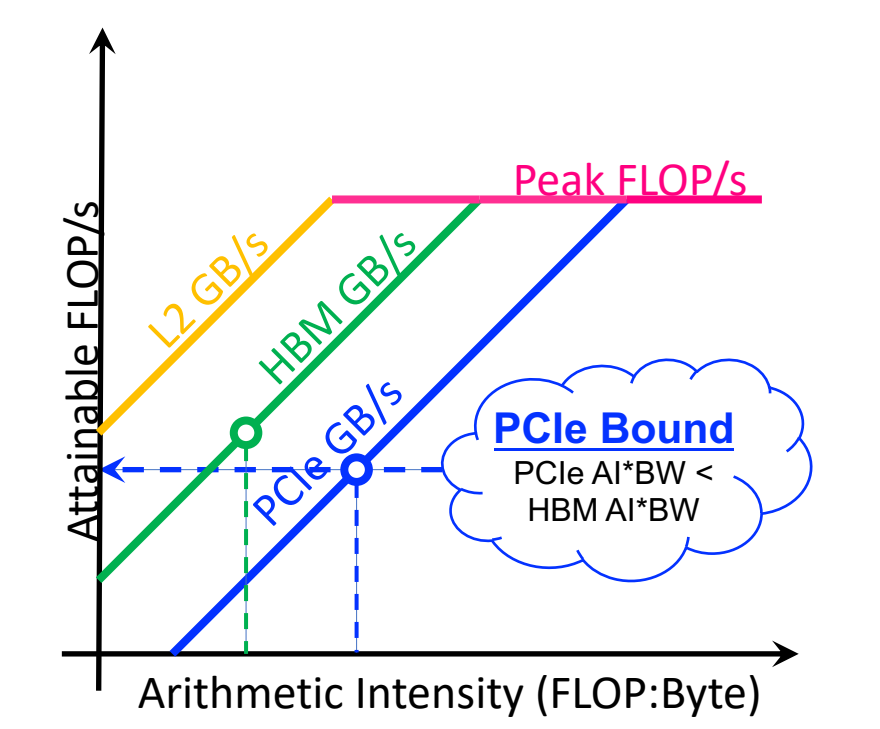
- Roofline is a throughput-oriented performance model
- Tracks **rates** not times
- Independent of ISA and architecture
- applies to CPUs, GPUs, Google TPUs, FPGAs, etc...
- Defines **Good Performance**



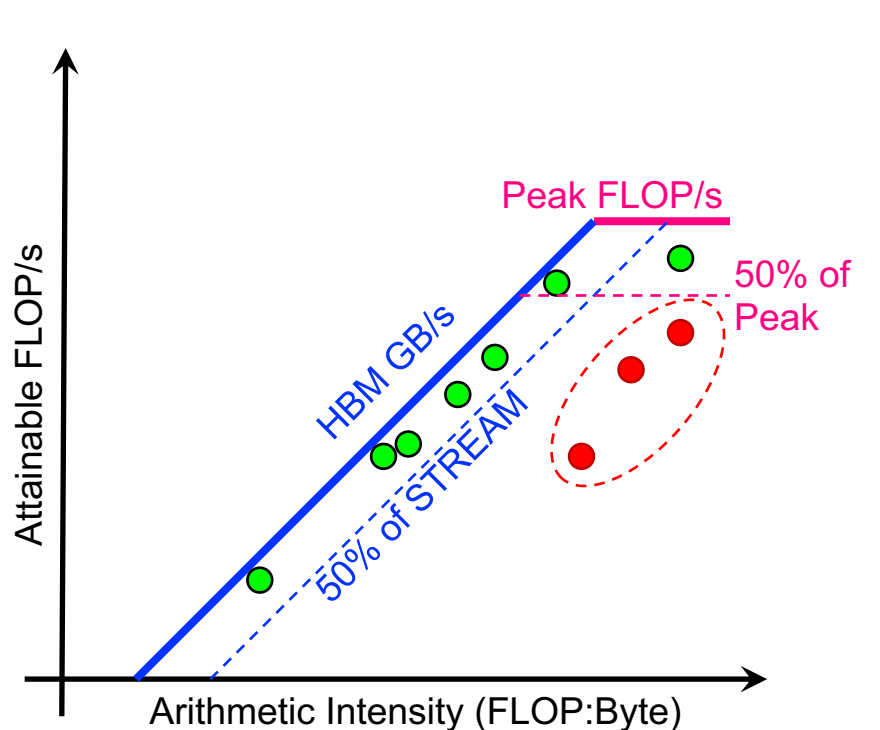
- Arithmetic Intensity is a measure of data locality
 - Ratio of **Total Flops** to **Total Bytes**
 - Includes cache and prefetcher effects
 - Can be very different from total loads/stores (bytes requested)
 - Equal to ratio of sustained GFLOPs to sustained GB/s (time cancels)



- Hierarchical Roofline
 - Applies to all levels of memory hierarchy on both CPUs and GPUs
 - Different data movements for L2/HBM/PCIe imply different arithmetic intensities
 - Differences in L2/HBM/PCIe intensity highlight differences in locality (similar AI's imply streaming)

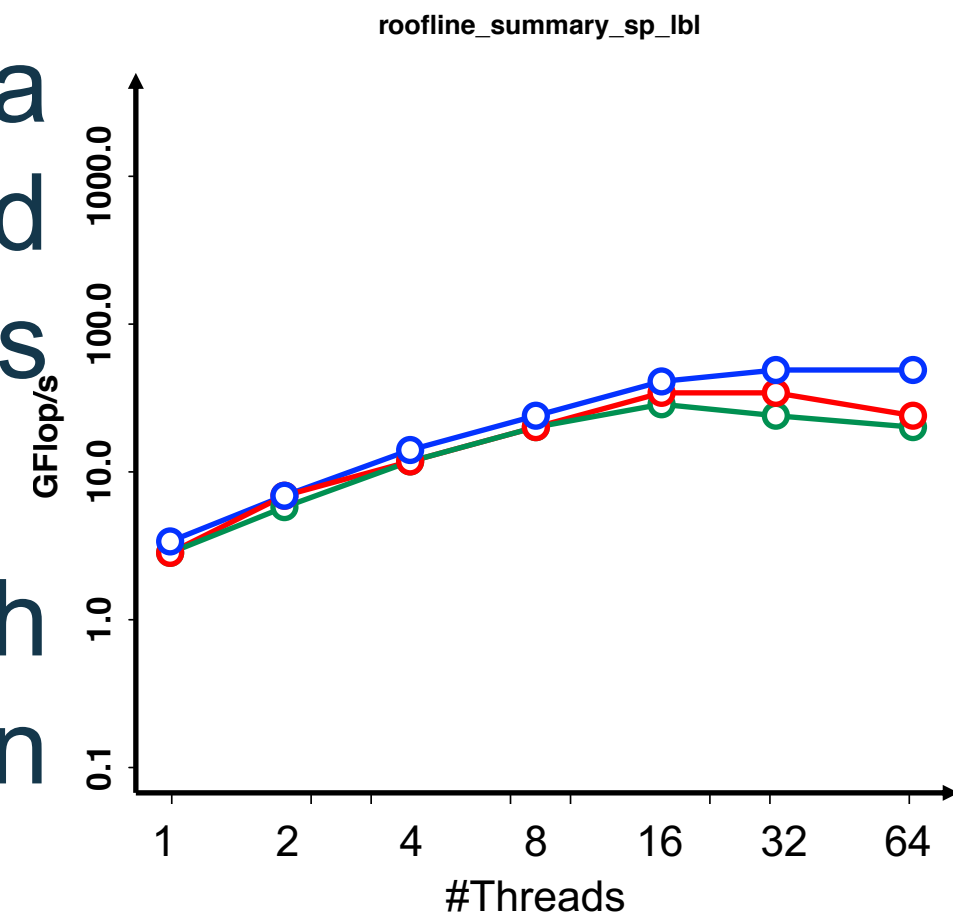


- Focus on important Loops, Kernels, Applications, ...
 - loops/kernels/apps attaining better than 50% of Roofline will see limited benefit from optimization
 - Users can use Roofline to identify underperforming loops/kernels/apps

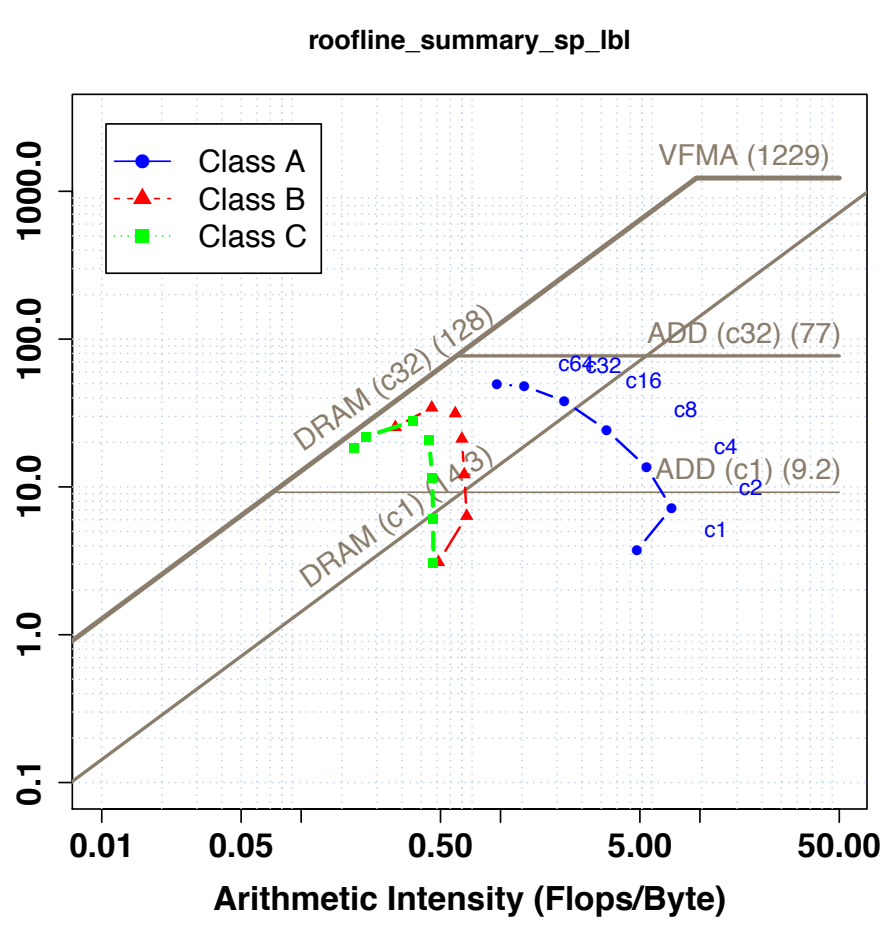


Scaling Trajectories

- Performance as a function of thread concurrency provides little insight
- Need better approach to understand turn overs in performance
- Use Roofline to analyze thread scalability
- "Roofline Scaling Trajectories"
 - 2D scatter plot of performance as a function of intensity and concurrency
 - Identify loss in performance due to increased cache pressure (data movement)



- NAS Parallel Benchmarks
- Intensity (data movement) varies with concurrency and problem size
- Large problems (green and red) move more data per thread, and exhaust cache capacity
- Falling Intensity → hit the bandwidth ceiling quickly and degrade.
- Useful for understanding locality/BW contention induced scaling bottlenecks

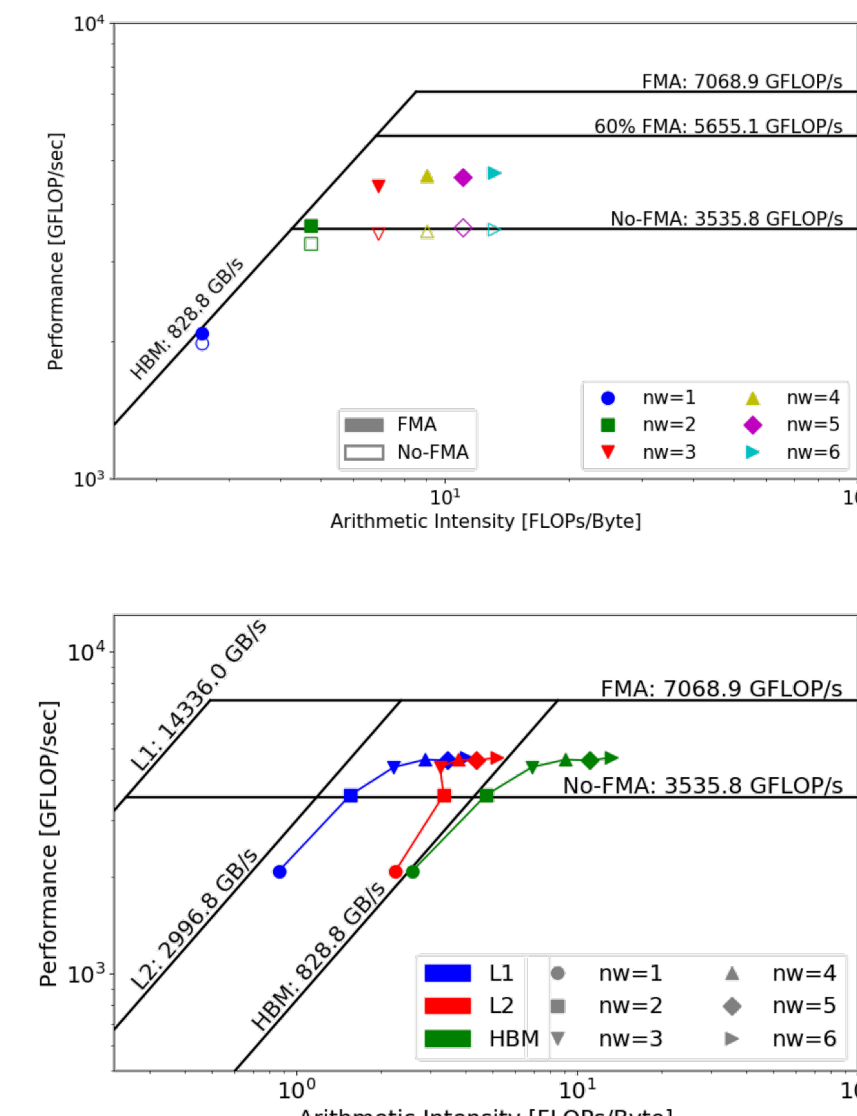


Roofline on GPUs

- Developed a Roofline methodology POC for analyzing applications running on NVIDIA GPUs
- Use NVProf to collect Roofline-related metrics (FLOPs, cache/DRAM data movement, etc...)

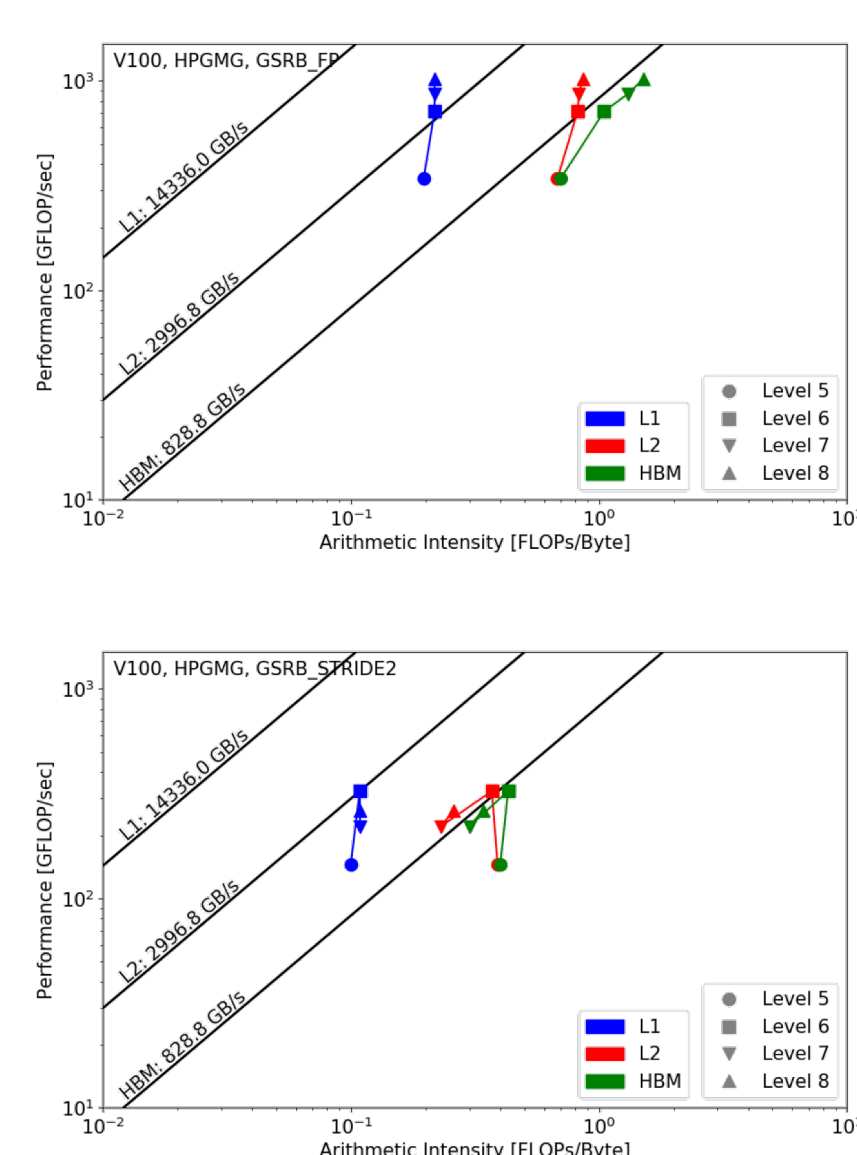
BerkeleyGW (Materials)

- <https://github.com/cyanquwa/BerkeleyGW-GPP>
- rw** increases data reuse in inner loop
 - More flops for fixed data movement
 - Understand cache effects
 - Quantify effects of FMA:MUL ratio (disable FMA in compiler)
- Observations...
 - High correlation with HBM BW
 - FMA doesn't hit FMA ceiling
 - High RF and L2 Locality
 - Minimal increases in L1 locality



HPGMG (Multigrid)

- <https://bitbucket.org/hpgmg/hpgmg>
- Multiple variants of GSRB smoother...
 - GSRB_FP does 2x the work but is trivial to implement
 - STRIDE2 requires more complex memory access and predication
- Observations...
 - High correlation with HBM BW for large problem sizes (level>5)
 - Moderate L1 cache locality
 - Low reuse in the L2 cache for GSRB_FP variant
 - STRIDE2 performance crashes due to decline in intensity



Roofline for TensorFlow

- Demonstrate methodology using conv2d from TensorFlow+cuDNN on V100 GPU

Setup...

```
input_image = tf.random_uniform(shape=input_size, minval=0., maxval=1., dtype=dtype)
output_result = conv2d(input_image, 'NHWC', kernel_size, stride_size, dtype)
```

Forward Pass (2D conv)

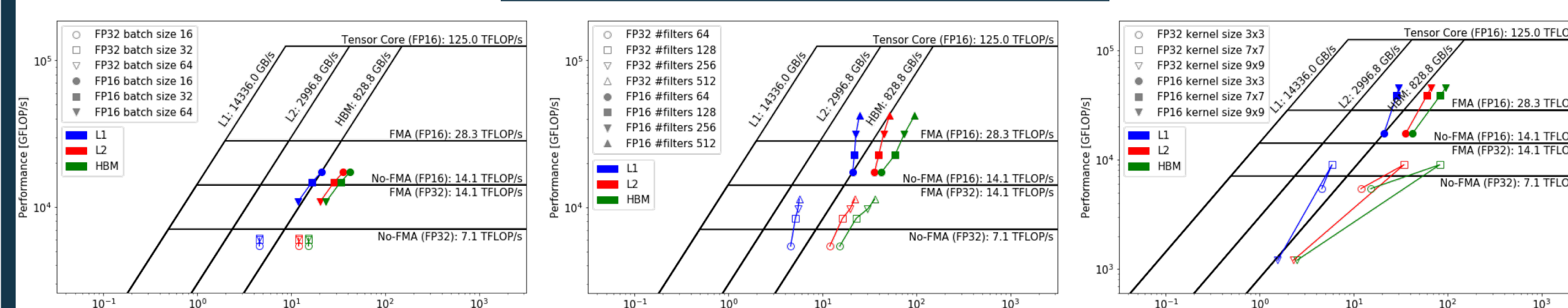
```
exec_op = output_result
```

Backward Pass (2D conv + derivative)

```
opt = tf.train.GradientDescentOptimizer(0.5)
exec_op = opt.compute_gradients(output_result)
```

- Each kernel includes multiple sub-kernels
 - Padding, permutations, conversions, compute, etc...
 - Should include all of them when analyzing performance
- TensorFlow also includes an autotuning step
 - Ignore autotuning when profiling/modeling
 - nvprof --profile-from-start off
 - run 5 warmup iterations (autotuning / not profiled)
 - start profiler (pyc.driver.start_profiler), run 20 iter, stop profiler
- Vary parameters to understand performance

conv2d Forward Pass



Batch Size

- Constant performance(?)
- FP16 performance anti-correlated with batch size
- Performance << TC peak
- Transformation kernels
- Low L2 locality

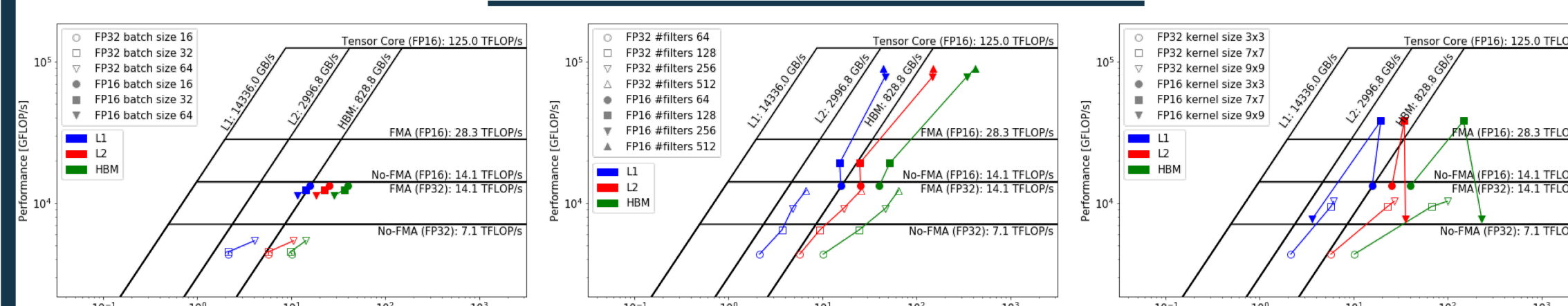
#Filters

- Intensity ∝ #Filters
- Low L2 data locality
- Some use of TC's (>FP16 FMA)... **partial TC ceiling**

Kernel Size

- Intensity ∝ kernel size
- Low L2 data locality
- Autotuner switched FP32 algorithm to FFT at 9x9
- Some use of TC's (>FP16 FMA)... **partial TC ceiling**

conv2d Backward Pass



Batch Size

- Autotuner chose different (better) algorithm for FP32 with batch size = 64 (boost)

#Filters

- Close to FP16 TC peak
- Close to FP32 FMA peak

Kernel Size

- Good FP32 performance trend (almost peak)
- Autotuner chose to run 9x9 FP16 in FP32 !!

Integration in Intel Advisor

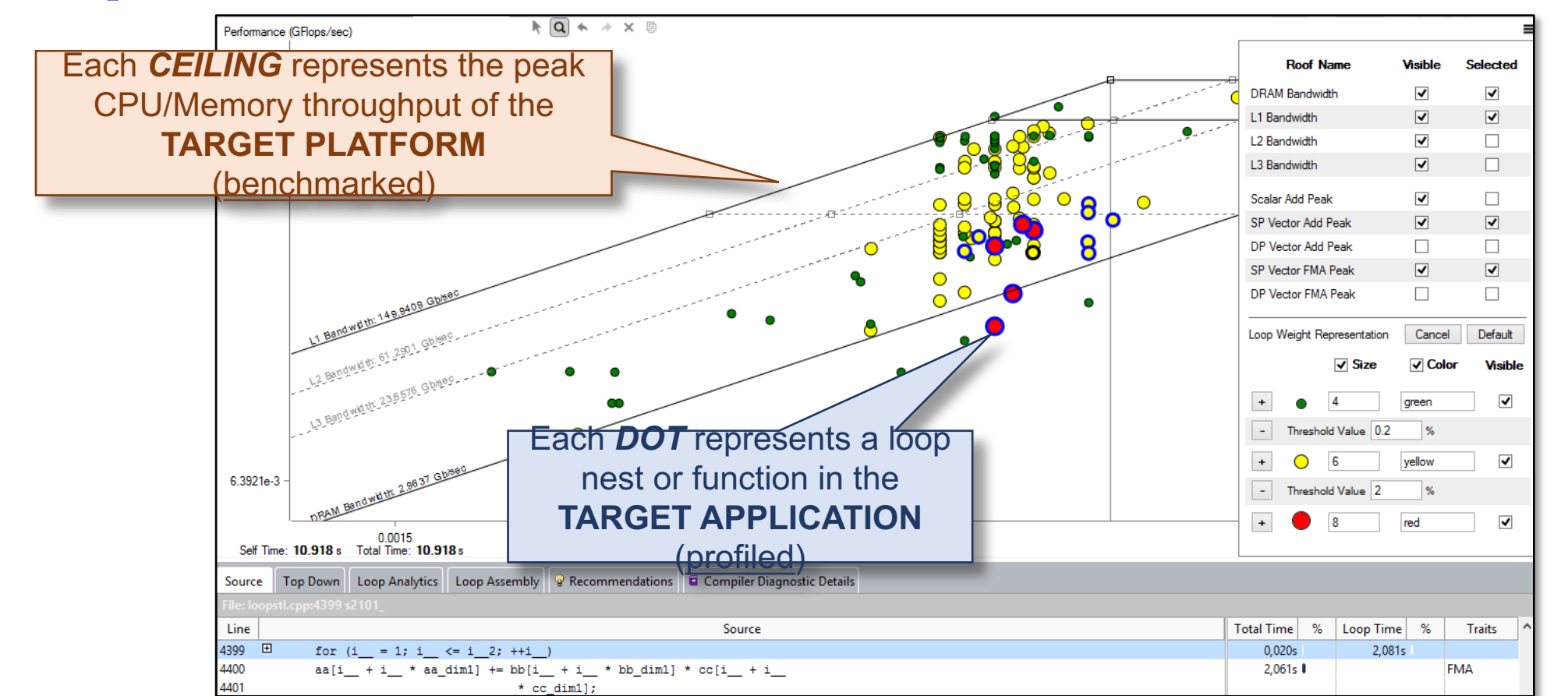
- Roofline has been integrated into Intel's Advisor Performance Tool...

- Automatically instruments applications (one dot per loop nest/function)
- Computes FLOPs and AI for each function / loop nest
- Integrated Cache Simulator (hierarchical roofline)
- Automatically benchmarks target system (calculates ceilings)
- AVX-512 support including vector masks
- Full integration with existing Advisor capabilities

- Fully supported on NERSC's Edison and Cori (Haswell and Knights Landing) Systems

- <http://www.nersc.gov/users/software/performance-and-debugging-tools/advisor/>

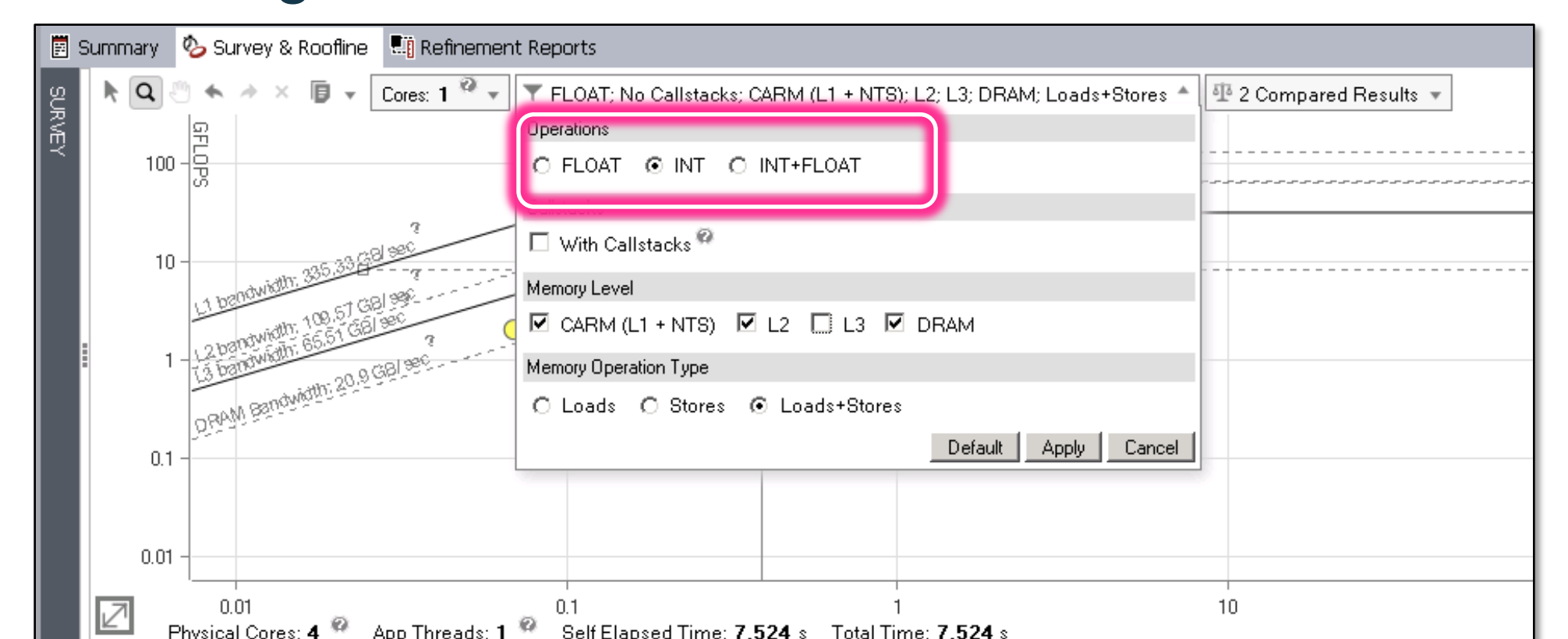
```
% module load advisor/2018.integrated_roofline
% cc -g -dynamic -openmp -O2 -o mycode.exe mycode.c
% source advixe-vars.sh
% advixe-cl -collect survey --project-dir ./your_project --
% <your-executable-with-parameters>
% advixe-cl -collect tripcounts -enable-cache-simulation -
% flop --project-dir ./your_project -- <your-executable-with-
% parameters>
```



- Increasingly, many applications have large, non-floating-point components (e.g. Genomics, Graphs, etc...)

- Traditional FLOP Roofline is irrelevant (no FLOPs)

- Advisor Roofline support expanded to include Integer and Integer+FLOP Rooflines



Community Engagement

- Strong collaboration with NERSC, Intel, and NVIDIA
- We've run Roofline tutorials at SC'17, SC'18, SC'19, ECP'18, ECP'19, ISC'18, ISC'19, NERSC, etc...

Publications

- <https://crd.lbl.gov/roofline/publications>
- C. Yang, T. Kurth, S. Williams, "Hierarchical Roofline Analysis for GPUs: Accelerating Performance Optimization for the NERSC-9 Perlmutter System", CUG, 2019.
- C. Yang, S. Williams, "Performance Analysis of GPU-Accelerated Applications using the Roofline Model", GTC, 2019.
- C. Yang, et al., "An Empirical Roofline Methodology for Quantitatively Assessing Performance Portability", P3HPC, 2018.
- K. Ibrahim, S. Williams, L. Oliker, "Roofline Scaling Trajectories: A Method for Parallel Application and Architectural Performance Analysis", HPBench, 2018.
- T. Koskela, et al., "A Novel Multi-Level Integrated Roofline Model Approach for Performance Characterization", ISC, 2018.