Performance Analysis using the Roofline Model

Samuel Williams (SWWilliams@lbl.gov), Charlene Yang, Khaleed 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 GFlop/s 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 A'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 turnover 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 NVPProf to collect Roofline-related metrics (FLOPs, cache/DRAM data movement, etc...)
- BerkeleyGW (Materials)
- nw increases data reuse in inner loop
  - More flops for fixed data movement
  - Understand cache effects
  - Quantity effects of FMA-MUL ratio (disabling FMA in compiler)
- Observations...
  - High correlation with HBM BW
  - PMA doesn’t hit FMA ceiling
  - High RF and L2 Locality
  - Minimal increases in L1 locality

HPMG (Multigrid)

- 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

- Development methodology using conv2d from TensorFlow+cudNN on V100 GPU

Setup...
- g++ -std=c++11 -O3 mycode_intel/imagenet/imagenet_train.cpp -o mycode_intel -L /opt/intel/compilers/0.3.4/lib/intel64 -L /opt/intel/compilers/0.3.4/lib/intel64 -l intel64_math -l intel64_math -l intel64_math -l intel64_math
- Forward Pass (2D conv)
- Backward Pass (2D conv + derivative)

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

Conv2d Forward Pass

Conv2d Forward Pass
- Convolution performed concurrently with batch size
- Transformation kernels
- Low L2 locality

Conv2d Backward Pass

Conv2d Backward Pass
- Autotuned performance (psyc) algorithm for FP32 with batch size = 100

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)

- % mobile load advisor/2014_integrated_roofline
- % cc -g -O3 -o mycode/mycode -g -o mycode.exe mycode.c
- % source advisor-vars.sh
- % Advisor-c /projects-dir ./your_project --your-executable-with-parameters
- % Advisor-c /projects-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://www.lbl.gov/roofline/publications