The Roofline Model:
A Pedagogical Tool for Program Analysis and Optimization

Sam Williams, David Patterson

EECS Electrical Engineering and Computer Sciences

Berkeley Par Lab

PARALLEL COMPUTING LABORATORY

Motivation, Goals and Audience

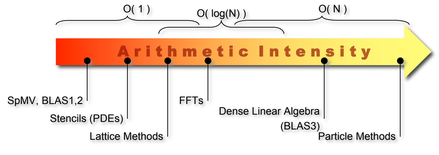
- Performance and scalability can be extremely non-intuitive on modern architectures
- Success of the multicore paradigm should be premised on extending the capabilities of the world's programmers
- Must provide a visually intuitive performance model that the bulk of the world's programmers (not just Ph.Ds) can use to optimize code
- Provide realistic performance and productivity expectations
- Focus on **kernel** performance and efficiency (e.g. Gflop/s)
- **Not intended for:**
  - those interested in fine tuning (+5%)
  - those challenged by program correctness

SPMD Performance Components

- Three performance components for SPMD kernels
- **Computation**
  - Typically floating point (single or double precision)
  - Could also be graphics, crypto, integer or bitwise operations
- **Communication**
  - Transfer of data from one level of the memory hierarchy to the next
  - Registers, L1, L2, DRAM, PCIe, Network
- **Locality**
  - Balance between communication and computation
  - Incorporates 3C's model for cache behavior
  - Results in a Flop:Byte ratio

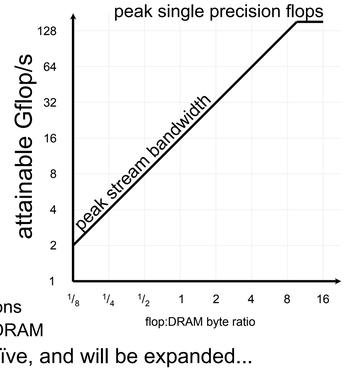
Arithmetic Intensity

- The flop per DRAM byte ratio is a well known quantity from HPC: **Arithmetic Intensity**
- Some kernels have arithmetic intensity that grows with the problem size (FFT, Matrix-Matrix multiplication, etc…)
- However, many interesting kernels have arithmetic intensity that remains constant with problem size (Matrix-Vector multiplication, Structured Grids, etc…)



Naïve Roofline Model

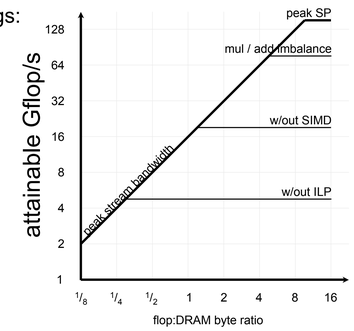
- We're primarily constrained by DRAM bandwidth.
- Naïvely, one may define a performance roofline based on the **minimum** of:
  - Peak (advertised) flops
  - Peak stream bandwidth x (flops per DRAM byte)
- Plot on **log-log** scale
- We can bound performance if we know, or can measure:
  - Total floating point operations
  - Total bytes transferred to DRAM
- In reality, this is far too naïve, and will be expanded...



Building the Roofline Model for Opteron

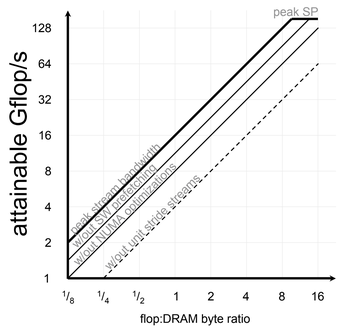
In-core Parallelism

- defines horizontal ceilings:
  - Instruction level (including pipelining)
  - SIMD
  - functional unit (separate units)
- Derived from optimization manuals
- Ordered (from bottom):
  - Inherent in the kernel
  - Exploitable by a compiler
  - Requires hand coding
  - Not inherent in the kernel



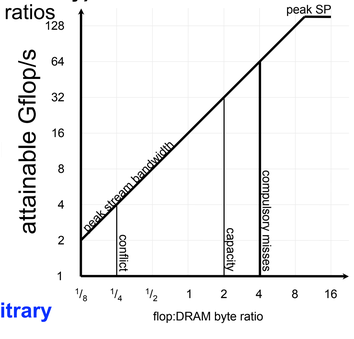
Memory Bandwidth

- Defines diagonal ceilings below peak stream bandwidth roofline:
  - Unit stride
  - NUMA
  - SW prefetching
- Obtained with microbenchmarks



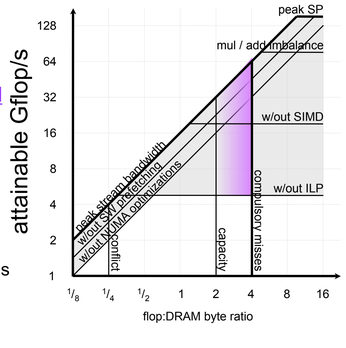
Locality (arithmetic intensity)

- Defines flop:DRAM byte ratios (walls)
- Can never do better than the compulsory flop:byte ratio
- Each architecture/kernel combination has a unique number of capacity and conflict misses
- Ideally obtained with performance counters
- **This example is an arbitrary kernel**



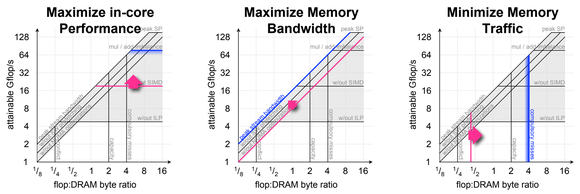
Roofline Model

- Integrate computation, communication, and locality into a single figure
- Performance is **bounded** by the optimizations implemented:
  - In-core optimizations (horizontal ceilings)
  - Bandwidth optimizations (diagonal ceilings)
  - Memory traffic optimizations (vertical walls)



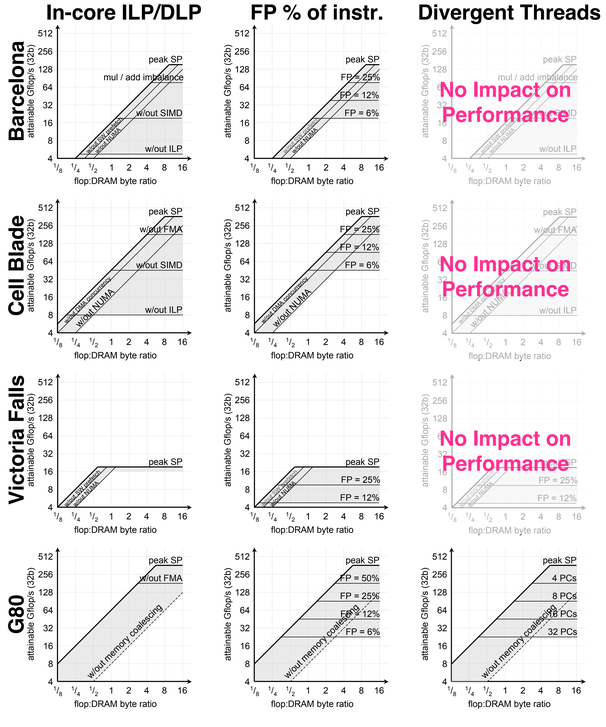
Three Types of Software Optimization

- Performance is limited by ceilings/walls
- In effect, a software optimization punches through them
- Performance at the kernel's (new) arithmetic intensity is limited by the remaining ceilings



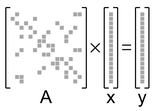
Other Architectural Paradigms ?

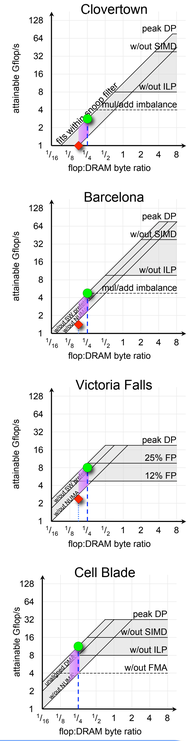
- Does this technique apply to other architectural paradigms?
- Create single precision (32b) roofline models for:
  - AMD Opteron 2356 (Barcelona)
  - IBM QS20 Cell Blade
  - Sun T2+ T5140 (Victoria Falls)
  - NVIDIA G80
- **In-core Performance**:
  - In-core parallelism is the primary challenge on superscalars
  - Non-FP instructions can sap the potentially limited instruction fetch/issue bandwidth
  - On NVIDIA, divergent threads consume more fetch bandwidth.
- **Bandwidth**:
  - Typically common (SW prefetch, NUMA, unit stride, …)
  - NVIDIA: if threads aren't all collaboratively loading a contiguous block, performance drops by more than 8x
- **Arithmetic intensity:**
  - Each architecture has unique cache/local store behavior (3C's)
- Each architecture has its own **Achilles' Heel(s)**



Using the Roofline to Analyze SpMV

- Sparse Matrix
  - Most entries are 0.0
  - Performance advantage in only storing/operating on the nonzeros
  - Requires significant meta data
- Evaluate **y=Ax**
  - A is a sparse matrix
  - **x** & **y** are dense vectors
- Challenges
  - Difficult to exploit ILP
  - Difficult to exploit DLP
  - Irregular memory access to **x**
  - Difficult to load balance
  - Very low arithmetic intensity (<0.166)
- Auto-tuning
  - NUMA, SW prefetch improve efficiency
  - Matrix compression eliminates compulsory misses
  - Original performance
  - Auto-tuned performance



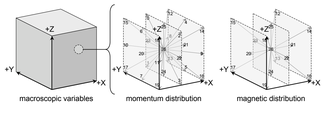


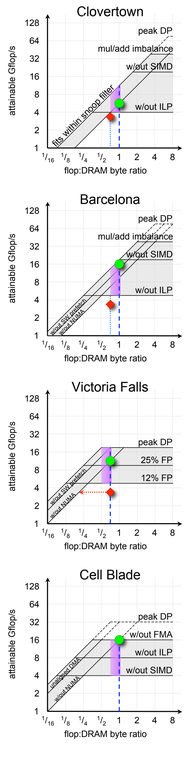
Reference

Samuel Williams, Leonid Oliker, Richard Vuduc, John Shalf, Katherine Yelick, James Demmel, "Optimization of Sparse Matrix-Vector Multiplication on Emerging Multicore Platforms", Supercomputing (SC), 2007.

Using the Roofline to Analyze LBMHD

- Plasma turbulence simulation
- Two distributions:
  - A is a sparse matrix
  - **x** & **y** are dense vectors
- Three macroscopic quantities:
  - Density
  - Momentum (cartesian vector)
  - Magnetic Field (cartesian vector)
- Lattice update:
  - Read 73 doubles
  - 1300 floating point operations
  - Write 79 doubles
  - arithmetic intensity ~ 1.0 (ideal)
- Auto-tuning
  - NUMA, unrolling, and SIMDization improve efficiency
  - Cache bypass eliminates compulsory misses
  - Original performance
  - Auto-tuned performance





Reference

Samuel Williams, Jonathan Carter, Leonid Oliker, John Shalf, Katherine Yelick, "Lattice Boltzmann Simulation Optimization on Leading Multicore Platforms", International Parallel & Distributed Processing Symposium (IPDPS) (to appear), 2008.

Best Paper, Application Track