The Roofline Model:
A pedagogical tool for program analysis and optimization

ParLab Summer Retreat
Samuel Williams, David Patterson

samw@cs.berkeley.edu
Motivation

- Performance and scalability of multicore architectures can be extremely non-intuitive to novice programmers
- Success of the multicore paradigm should be premised on augmenting the abilities of the world’s programmers
Goals & Audience

- Focused on:
  - rates and efficiencies (Gflop/s, % of peak),

- Goals for Roofline:
  - Provide everyone with a graphical aid that provides:
    - realistic expectations of performance and productivity
  - Show inherent hardware limitations for a given kernel
  - Show potential benefit and priority of optimizations

- Who’s not the audience for the Roofline:
  - Not for those interested in fine tuning (+5%)
  - Not for those challenged by parallel kernel correctness
Principal Components of Performance
Components

- There are three principal components to performance:
  - Computation
  - Communication
  - Locality

- Each architecture has a different balance between these
- Each kernel has a different balance between these

- Performance is a question of how well an kernel’s characteristics map to an architecture’s characteristics
For us, floating point performance ($\text{Gflop/s}$) is the metric of interest (typically double precision)

Peak in-core performance can only be attained if:
- fully exploit ILP, DLP, FMA, etc…
- non-FP instructions don’t sap instruction bandwidth
- threads don’t diverge (GPUs)
- transcendental/non pipelined instructions are used sparingly
- branch mispredictions are rare

To exploit a form of in-core parallelism, it must be:
- Inherent in the algorithm
- Expressed in the high level implementation
- Explicit in the generated code
Communication

- For us, DRAM bandwidth (GB/s) is the metric of interest.

- Peak bandwidth can only be attained if certain optimizations are employed:
  - Few unit stride streams
  - NUMA allocation and usage
  - SW Prefetching
  - Memory Coalescing (GPU)
Locality

- Computation is free, Communication is expensive.
- Maximize locality to minimize communication
- **There is a lower limit to communication: compulsory traffic**

- Hardware changes can help minimize communication
  - Larger cache capacities minimize capacity misses
  - Higher cache associativities minimize conflict misses
  - Non-alloacting caches minimize compulsory traffic

- Software optimization can also help minimize communication
  - Padding avoids conflict misses
  - Blocking avoids capacity misses
  - Non-alloacting stores minimize compulsory traffic
Roofline Model
Integrating Components...

- Goal: integrate in-core performance, memory bandwidth, and locality into a single readily understandable performance figure
- Also, must graphically show the penalty associated with not including certain software optimizations

- Roofline model will be unique to each architecture
- Coordinates of a kernel are ~unique to each architecture
What relates GB/s to GFlop/s? 

- Through dimensional analysis, it's clear that Flops:Bytes is the parameter that allows us to convert bandwidth (GB/s) to performance (GFlop/s).

- This is a well known quantity: Arithmetic Intensity (discussed later).

- When we measure total bytes, we incorporate all cache behavior (the 3C’s) and Locality.
Basic Roofline

- Performance is upper bounded by both the peak flop rate, and the product of streaming bandwidth and the flop:byte ratio

\[
\text{Gflop/s} = \min \left\{ \begin{array}{l}
\text{Peak Gflop/s} \\
\text{Stream BW} \times \text{actual flop:byte ratio}
\end{array} \right. 
\]
• *Bandwidth #’s collected via micro benchmarks*

• *Computation #’s derived from optimization manuals (pencil and paper)*

• *Assume complete overlap of either communication or computation*
Roofline model for Opteron

(adding ceilings)

- Peak roofline performance
- Based on manual for single precision peak
- And a hand tuned stream read for bandwidth

AMD Opteron 2356
(Barcelona)

Peak SP

![Graph showing the roofline model for AMD Opteron 2356. The x-axis represents the flop:DRAM byte ratio on a log scale, and the y-axis represents the attainable Gflop/s on a log scale. The graph includes a line showing the peak SP performance.](image-url)
Roofline model for Opteron
(adding ceilings)

AMD Opteron 2356
(Barcelona)

- Peak roofline performance
- Based on manual for single precision peak
- And a hand tuned stream read for bandwidth
Roofline model for Opteron
(adding ceilings)

AMD Opteron 2356
(Barcelona)

- Opterons have separate multipliers and adders
- ‘functional unit parallelism’
- This is a ceiling beneath the roofline
Roofline model for Opteron
(adding ceilings)

AMD Opteron 2356
(Barcelona)

- In single precision, SIMD is 4x32b.
- If only the _ss versions are used, performance is 1/4.
Roofline model for Opteron
(adding ceilings)

If 4 independent instructions are kept in the pipeline, performance will fall.
Roofline model for Opteron
(adding ceilings)

AMD Opteron 2356
(Barcelona)

- If SW prefetching is not used, performance will degrade
- These act as ceilings below the bandwidth roofline

<table>
<thead>
<tr>
<th>flop:DRAM byte ratio</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>attainable Gflop/s</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

- peak SP
- mul / add imbalance
- w/out SIMD
- w/out ILP
- peak stream bandwidth
- w/out SW prefetching
- w/out ILP
- w/out SIMD
- peak SP
Roofline model for Opteron
(adding ceilings)

AMD Opteron 2356 (Barcelona)

Without NUMA optimizations, the memory controllers on the second socket can’t be used.
Roofline model for Opteron
(adding ceilings)

AMD Opteron 2356
(Barcelona)

- Bandwidth is much lower without unit stride streams
Roofline model for Opteron
(adding ceilings)

AMD Opteron 2356
(Barcelona)

- Its difficult for any architecture to reach the raw DRAM bandwidth
Roofline model for Opteron
(adding ceilings)

Partitions the regions of expected performance into three optimization regions:
- Compute only
- Memory only
- Compute+Memory
Uniqueness

- There is no single ordering or roofline model
- The order of ceilings is generally (bottom up):
  - What is inherent in algorithm
  - What a compiler is likely to provide
  - What a programmer could provide
  - What can never be exploited for this kernel
- For example,
  - FMA or mul/add balance is inherent in many linear algebra routines and should be placed at the bottom.
  - However, many stencils are dominated by adds, and thus the multipliers and FMA go underutilized.
Arithmetic Intensity in HPC

- **Arithmetic Intensity (AI)** ~ Total Flops / Total DRAM Bytes
- Some HPC kernels have an arithmetic intensity that’s constant, but on others it scales with problem size (increasing temporal locality)
- Actual arithmetic intensity is capped by cache/local store capacity
Accurately Determining the true Flop:DRAM Byte ratio

- Remember the 3C’s of caches

- Calculating the Flop:DRAM byte ratio is:
  - **Compulsory misses**: straightforward
  - **Capacity misses**: pencil and paper (maybe performance counters)
  - **Conflict misses**: must use performance counters

- Flop:actual DRAM Byte ratio < Flop:compulsory DRAM Byte ratio

- One might place a range on the arithmetic intensity ratio
- Thus performance is limited to an area between the ceilings and between the upper (compulsory) and lower bounds on arithmetic intensity
Roofline model for Opteron
(powerpoint doodle)

AMD Opteron 2356
(Barcelona)

- Final Roofline

- peak SP
- mul / add imbalance
- w/out SIMD
- w/out ILP
- w/out SW prefetching
- w/out NUMA optimizations
- peak stream bandwidth
Some arbitrary kernel has a flop:compulsory byte ratio of 4.

- Overlaid on the roofline.
- Defines upper bound on range of expected performance.
- Also shows which optimizations are likely.
Roofline model for Opteron
(powerpoint doodle)

- Capacity misses reduce the actual flop:byte ratio
- Also reduces attainable performance
- AI is unique to each combination of kernel and architecture

**AMD Opteron 2356 (Barcelona)**

- Peak SP
- Mul / add imbalance
- W/out SIMD
- W/out ILP
- Compulsory misses
- W/out NUMA optimizations
- W/out SW prefetching
- W/out stream bandwidth

- 2
- 4
- 6
- 8
- 16
- 2^1
- 2^2
- 2^3
- 2^4
- 2^5
- 2^6
- 2^7
- 2^8
- 2^9
- 2^10
- 2^11
- 2^12
- 2^13
- 2^14
- 2^15
- 2^16

- 1
- 2
- 4
- 8
- 16
- 32
- 64
- 128

- 1/8
- 1/4
- 1/2
- 1
- 2
- 4
- 8
- 16

flop:DRAM byte ratio

attainable Gflop/s
Roofline model for Opteron
(powerpoint doodle)

AMD Opteron 2356
(Barcelona)

- Conflict misses may destroy performance
- AI is unique to each combination of kernel and architecture

Peak SP

Conflict misses may destroy performance
AI is unique to each combination of kernel and architecture
Roofline model for Opteron
(powerpoint doodle)

AMD Opteron 2356
(Barcelona)

Conflict misses may destroy performance

Doubling cache capacity doesn't double the arithmetic intensity!
Three Categories of Software Optimization
Maximizing Attained in-core Performance

软件优化，如显式SIMD化，可以突破水平天花板（编译器可以预期的）。其他例子包括循环展平、重排序和长运行循环。

- AMD Opteron 2356 (Barcelona)

- 软件优化，如显式SIMD化，可以突破水平天花板（编译器可以预期的）。其他例子包括循环展平、重排序和长运行循环。
Maximizing Attained Memory Bandwidth

- Compilers won’t give great out-of-the-box bandwidth
- Punch through bandwidth ceilings:
  - Maximize MLP
  - Long unit stride accesses
  - NUMA aware allocation and parallelization
  - SW prefetching

![Graph showing memory bandwidth scalability](image)
Minimizing Memory Traffic

- Use performance counters to measure flop:byte ratio (AI)
- Out-of-the-box code may have an AI ratio much less than the compulsory ratio
  - Be cognizant of cache capacities, associativities, and threads sharing it
  - Pad structures to avoid conflict misses
  - Use cache blocking to avoid capacity misses
- These optimizations can be imperative
Before optimization, traffic, and limited bandwidth optimization limits performance to a very narrow window.
Effective Roofline (after)

After optimization, ideally, performance is significantly better.
Applicable to Other Architectural Paradigms ?
Four Architectures

**AMD Barcelona**
- 667MHz DDR2 DIMMs
- 2x64b memory controllers
- Hyper Transport (64b/s each direction)
- SRI / crossbar
- 512KB victim
- 512KB victim
- 512KB victim
- 512KB victim
- 2MB Shared quasi-victim (32 way)
- Hyper Transport
- 10.66 GB/s

**Sun Victoria Falls**
- 667MHz FBDIMMs
- 21.33 GB/s (each direction)
- 10.66 GB/s
- 2x64b memory controllers
- Hyper Transport (1 per hub per direction)
- 179 GB/s
- 90 GB/s
- 4 Coherency Hubs
- 2x128b controllers
- Crossbar
- 4MB Shared L2 (64b interleaved)

**IBM Cell Blade**
- 512MB XDR DRAM
- XDR memory controllers
- 10.66 GB/s
- 25.6 GB/s
- EIB (ring network)
- VMT PPE
- 512K L2

**NVIDIA G80**
- 768MB 900MHz GDDR3 Device DRAM
- 6x64b memory controllers
- 38.4 GB/s
Four Architectures

**AMD Barcelona**
- Opteron
- 512KB victim
- 2MB Shared quasi-victim (32 way)
- 2x64b memory controllers
- 667MHz DDR2 DIMMs
- 10.66 GB/s
- 2MB Shared quasi-victim (32 way)
- SRI / crossbar
- HyperTransport

**IBM Cell Blade**
- VMT PPE
- 512KB L2
- EIB (ring network)
- XDR memory controllers
- 512MB XDR DRAM
- 25.6 GB/s

**Sun Victoria Falls**
- MT SPARC
- Crossbar
- 179 GB/s
- 190 GB/s
- 196 GB/s
- 4 Coherency Hubs
- 2x128b controllers
- Sun Victoria Falls
- 21.33 GB/s
- 10.66 GB/s
- 667MHz FBDIMMs

**NVIDIA G80**
- Thread Cluster
- Thread Cluster
- Thread Cluster
- Thread Cluster
- Thread Cluster
- Thread Cluster
- Thread Cluster
- Thread Cluster
- 192KB L2 (Textures only)
- 24 ROPs
- 6 x 64b memory controllers
- 768MB 900MHz GDDR3 Device DRAM
- 21.33 GB/s
- 10.66 GB/s
- 667MHz FBDIMMs

One Thread/Core
Multithreaded Cores
Four Architectures

**AMD Barcelona**
- Opteron
  - 512KB victim
  - 2MB Shared quasi-victim (32 way)
  - 667MHz DDR2 DIMMs
- XDR memory controllers
  - SRI / crossbar
- HyperTransport
  - 10.66 GB/s
- Crossbar

**Sun Victoria Falls**
- Opteron
  - 512KB victim
  - 2MB Shared quasi-victim (32 way)
  - 667MHz DDR2 DIMMs
- XDR memory controllers
  - SRI / crossbar
- HyperTransport
  - 10.66 GB/s
- Crossbar

**IBM Cell Blade**
- VMT PPE
  - 512KB L2
  - EIB (ring network)
  - XDR memory controllers
  - 25.6 GB/s
- 512MB XDR DRAM

**NVIDIA G80**
- Thread Cluster
  - 192KB L2 (Textures only)
  - 24 ROPs
  - 6x64 memory controllers
  - 186.4 GB/s
- 768MB 900MHz GDDR3 Device DRAM
32b Rooflines for the Four
(in-core parallelism)

- Single Precision Roofline models for the SMPs used in this work.
- Based on micro-benchmarks, experience, and manuals
- Ceilings = in-core parallelism
- Can the compiler find all this parallelism?
- NOTE:
  - log-log scale
  - Assumes perfect SPMD
32b Rooflines for the Four
(diverged threads)

- AMD Barcelona
  - G80 dynamically finds DLP (shared instruction fetch)
  - SIMT
  - If threads of a warp diverge from SIMD execution, performance is limited by instruction issue bandwidth
  - Ceilings on G80 = number of unique PCs when threads diverge

- Sun Victoria Falls

- IBM Cell Blade

- NVIDIA G80
  - peak SP
  - w/out SIMD
  - w/out ILP
  - w/out FMA
  - w/out memory coalescing
  - w/out NUMA
  - w/out SW prefetch
  - w/out DMA concurrency
  - w/out NUMA
  - w/out SW prefetch
  - w/out DMA concurrency
  - peak SP
  - w/out SIMD
  - w/out ILP
  - w/out FMA
  - w/out memory coalescing
  - w/out NUMA
  - w/out SW prefetch
  - w/out DMA concurrency
  - peak SP
  - w/out SIMD
  - w/out ILP
  - w/out FMA
  - w/out memory coalescing
  - w/out NUMA
  - w/out SW prefetch
  - w/out DMA concurrency

- flop:DRAM byte ratio

- attainable Gflop/s (32b)
32b Rooflines for the Four
(FP fraction of dynamic instructions)

- Some kernels have large numbers of non FP instructions
- Saps instruction issue bandwidth
- Ceilings = FP fraction of dynamic instruction mix

NOTE:
  - Assumes perfect in-core parallelism
32b Rooflines for the Four

(ridge point)

- Some architectures have drastically different ridge points
- VF may be compute bound on many kernels
- Clovertown has \( \frac{1}{3} \) the BW of Barcelona = ridge point to the right
Using Roofline when Auto-tuning HPC Kernels
Multicore SMPs Used

Intel Xeon E5345 (Clovertown)

Core Core Core Core
4MB shared L2
4MB shared L2
4MB shared L2
4MB shared L2

FSB
10.66 GB/s
21.33 GB/s (read)
10.66 GB/s (write)

Chipset (4x64b controllers)

667MHz FBDIMMs

AMD Opteron 2356 (Barcelona)

Opteron Opteron Opteron Opteron
Opteron Opteron Opteron Opteron
512KB victim 512KB victim 512KB victim 512KB victim
512KB victim 512KB victim 512KB victim 512KB victim
2MB Shared quasi-victim (32 way) 2MB Shared quasi-victim (32 way)

SRI / crossbar

2x64b memory controllers

667MHz DDR2 DIMMs

Sun T2+ T5140 (Victoria Falls)

MT SPARC MT SPARC MT SPARC MT SPARC
MT SPARC MT SPARC MT SPARC MT SPARC
MT SPARC MT SPARC MT SPARC MT SPARC
MT SPARC MT SPARC MT SPARC MT SPARC

Crossbar

Crossbar

4MB Shared L2 (16-way)
(64b interleaved)

4 Coherency Hubs
2x128b controllers

4MB Shared L2 (16-way)
(64b interleaved)

4 Coherency Hubs
2x128b controllers

179 GB/s
21.33 GB/s
10.66 GB/s
667MHz FBDIMMs

IBM QS20 Cell Blade

VMT SPE
MT SPE
MFC SPE
MT SPE
MT SPE
VMT SPE
MT SPE
MT SPE
MFC SPE
MT SPE
MT SPE
VMT SPE
MT SPE
MT SPE
MFC SPE
MT SPE
MT SPE

512K L2
512K L2
512K L2
512K L2

EIB (ring network)

XDR memory controllers

512MB XDR DRAM

512MB XDR DRAM
Sparse Matrix-Vector Multiplication (SpMV)

Sparse Matrix Vector Multiplication

- **Sparse Matrix**
  - Most entries are 0.0
  - Performance advantage in only storing/operating on the nonzeros
  - Requires significant metadata

- **Evaluate** $y = Ax$
  - $A$ is a sparse matrix
  - $x$ & $y$ are dense vectors

- **Challenges**
  - Difficult to exploit ILP (bad for superscalar),
  - Difficult to exploit DLP (bad for SIMD)
  - Irregular memory access to source vector
  - Difficult to load balance
  - **Very low arithmetic intensity** (often $<0.166$ flops/byte)
    $\Rightarrow$ likely memory bound
Roofline model for SpMV

- Double precision roofline models
- FMA is inherent in SpMV
  (place at bottom)
Roofline model for SpMV

- Two unit stride streams
- Inherent FMA
- No ILP
- No DLP
- FP is 12-25%
- Naïve compulsory flop:byte < 0.166

Cloverown

- Peak DP
- w/out SIMD
- w/out ILP
- mul/add imbalance
- fits within snoop filter

Barcelona

- Peak DP
- w/out SIMD
- w/out ILP
- mul/add imbalance

Victoria Falls

- Peak DP
- 25% FP
- 12% FP

Cell Blade

- None of the previous limitations
- No naïve Cell implementation
Roofline model for SpMV
(out-of-the-box parallel)

- Two unit stride streams
- Inherent FMA
- No ILP
- No DLP
- FP is 12-25%
- Naïve compulsory flop:byte < 0.166
- For simplicity: dense matrix in sparse format

Clovertown

Barcelona

Victoria Falls

Cell Blade

No naïve Cell implementation
Roofline model for SpMV (NUMA & SW prefetch)

- **Clovertown**
  - peak DP
  - w/out SIMD
  - w/out ILP
  - mull/add imbalance
  - fits within snoop filter

- **Barcelona**
  - peak DP
  - w/out SIMD
  - w/out ILP

- **Victoria Falls**
  - peak DP
  - 25% FP
  - w/out SIMD
  - w/out ILP

- **Cell Blade**
  - peak DP
  - 12% FP
  - w/out SIMD
  - w/out ILP

- compulsory flop:byte ~ 0.166
- utilize all memory channels
Roofline model for SpMV (matrix compression)

- Inherent FMA
- Register blocking improves ILP, DLP, flop:byte ratio, and FP% of instructions

Graphs showing the attainable Gflop/s for Clovrtown, Barcelona, Victoria Falls, and Cell Blade with different flop:DRAM byte ratio and various optimizations.
Lattice-Boltzmann Magneto-Hydrodynamics (LBMHD)


Best Paper, Application Track
Plasma turbulence simulation via Lattice Boltzmann Method

Two distributions:
- momentum distribution (27 scalar components)
- magnetic distribution (15 vector components)

Three macroscopic quantities:
- Density
- Momentum (vector)
- Magnetic Field (vector)

Must read 73 doubles, and update 79 doubles per point in space
Requires about 1300 floating point operations per point in space
Just over 1.0 flops/byte (ideal)
Roofline model for LBMHD

- Huge datasets
- NUMA allocation/access
- Little ILP
- No DLP
- Far more adds than multiplies (imbalance)
- Essentially random access to memory
- Flop:byte ratio ~0.7
- High conflict misses
Roofline model for LBMHD
(out-of-the-box code)

- Huge datasets
- NUMA allocation/access
- Little ILP
- No DLP
- Far more adds than multiplies (imbalance)
- Essentially random access to memory
- Flop:byte ratio ~0.7
- High conflict misses
- Peak VF performance with 64 threads (out of 128) - high conflict misses
Roofline model for LBMHD
(Padding, Vectorization, Unrolling, Reordering)

- Vectorize the code to eliminate TLB capacity misses
- Ensures unit stride access (bottom bandwidth ceiling)
- Tune for optimal VL
- Clovertown pinned to lower BW ceiling

![Diagram of Roofline model for Clovertown, Barcelona, Victoria Falls, and Cell Blade]
Roofline model for LBMHD
(SIMDization + cache bypass)

- Make SIMDization explicit
- Technically, this swaps ILP and SIMD ceilings
- Use cache bypass instruction: *movntpd*
- Increases flop:byte ratio to ~1.0 on x86/Cell
The Heat Equation Stencil

The Heat Equation Stencil

- Explicit Heat equation on a regular grid
- Jacobi
- One double per point in space
- 7-point nearest neighbor stencil
- Must:
  - read every point from DRAM
  - perform 8 flops (linear combination)
  - write every point back to DRAM
- Just over 0.5 flops/byte (ideal)
- Cache locality is important
Roofline model for Stencil
(out-of-the-box code)

- Large datasets
- 2 unit stride streams
- No NUMA
- Little ILP
- No DLP
- Far more adds than multiplies (imbalance)
- Ideal flop:byte ratio $\frac{1}{3}$
- High locality requirements
- Capacity and conflict misses will severely impair flop:byte ratio

**Clovertown**

- Peak DP
- Mul/add imbalance
- W/out SIMD
- W/out ILP
- W/out NUMA
- Unaligned DMA
- W/out SW prefetch

**Barcelona**

- Peak DP
- Mul/add imbalance
- W/out SIMD
- W/out ILP
- W/out NUMA
- Without FMA
- W/out ILP
- W/out SIMD

**Victoria Falls**

- Peak DP
- Mul/add imbalance
- W/out SIMD
- W/out ILP
- W/out NUMA
- W/out SW prefetch
- W/out NUMA

**Cell Blade**

- Peak DP
- Mul/add imbalance
- W/out SIMD
- W/out ILP
- W/out NUMA
- W/out SW prefetch
- W/out NUMA

**No naïve Cell implementation**
Roofline model for Stencil
(out-of-the-box code)

- Large datasets
- 2 unit stride streams
- No NUMA
- Little ILP
- No DLP
- Far more adds than multiplies (imbalance)
- Ideal flop:byte ratio $\frac{1}{3}$
- High locality requirements
- Capacity and conflict misses will severely impair flop:byte ratio

Clovertown

Barcelona

Victoria Falls

Cell Blade

No naïve Cell implementation
## Roofline model for Stencil

(NUMA, cache blocking, unrolling, prefetch, …)

- **Clovertown**
  - Cache blocking helps ensure flop:byte ratio is as close as possible to $\frac{1}{3}$
  - Clovertown has huge caches but is pinned to lower BW ceiling
  - Cache management is essential when capacity/thread is low

<table>
<thead>
<tr>
<th>flop:DRAM byte ratio</th>
<th>1</th>
<th>1/2</th>
<th>1/4</th>
<th>1/8</th>
<th>1/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flop:DRAM ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victoria Falls</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Cell Blade</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Barcelona</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Clovertown</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Cell Blade</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Barcelona</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Clovertown</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Cell Blade</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Barcelona</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Clovertown</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Cell Blade</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

- **No naïve Cell implementation**
Roofline model for Stencil
(SIMDization + cache bypass)

- Make SIMDization explicit
- Technically, this swaps ILP and SIMD ceilings
- Use cache bypass instruction: \textit{movntpd}
- Increases flop:byte ratio to \( \sim 0.5 \) on x86/Cell
Refining the Roofline
Other performance metrics

- There is no reason either floating point (Gflop/s) must be the performance metric

- Could also use:
  - Graphics (Pixels, Vertices, Textures)
  - Crypto
  - Integer
  - Bitwise
  - etc…
For our kernels, DRAM bandwidth is the key communication component.

For other kernels, other bandwidths might be more appropriate:
- L2 bandwidth (e.g. DGEMM)
- PCIe bandwidth (offload to GPU)
- Network bandwidth

The example below shows zero overhead double buffered transfers to/from a GPU over PCIe x16:
- How bad is a SP stencil?
- What about SGEMM?

No overlap / high overhead tends to smooth performance:
- Performance is half at ridge point

![Diagram showing the relationship between flop:PCIe byte ratio and attainable Gflop/s.](attachment:diagram.png)
Mix and match

- In general, you can mix and match as the kernel/architecture requires:
- e.g. all possibilities is the cross product of performance metrics with bandwidths

\[
\{\text{Gflop/s, GIPS, crypto, \ldots}\} \times \{\text{L2, DRAM, PCIe, Network}\}
\]
Conclusions

- The Roofline Model provides an intuitive graph for kernel analysis and optimization.
- Easily extendable to other architectural paradigms.
- Easily extendable to other communication or computation metrics.
Questions?
BACKUP SLIDES
Trends

- ILP is decreasing (shorter pipelines, multithreading)
- SIMD is becoming wider
- Bandwidth isn’t keeping up with #cores
- Application flop:byte is decreasing
- Cache/Local Store management is becoming critical
- ILP is decreasing (shorter pipelines, multithreading)
- SIMD is becoming wider
- Bandwidth isn’t keeping up with #cores
- Application flop:byte is decreasing
- Cache/Local Store management is becoming critical

% of peak Gflop/s vs flop:DRAM byte ratio

- peak SP
- mul / add imbalance
- w/out ILP
- w/out SIMD
- peak stream bandwidth
- w/out memory optimizations
Trends

- ILP is decreasing (shorter pipelines, multithreading)
- SIMD is becoming wider
- Bandwidth isn’t keeping up with #cores
- Application flop:byte is decreasing
- Cache/Local Store management is becoming critical
Trends

- ILP is decreasing (shorter pipelines, multithreading)
- SIMD is becoming wider
- Bandwidth isn’t keeping up with #cores
- Application flop:byte is decreasing
- Cache/Local Store management is becoming critical
Trends

- ILP is decreasing (shorter pipelines, multithreading)
- SIMD is becoming wider
- Bandwidth isn’t keeping up with #cores
- Application flop:byte is decreasing
- Cache/Local Store management is becoming critical
Same formalism can be plotted on different axis

Difficult to plot AI
No Overlap

- What if computation or communication isn’t totally overlapped
- At ridgepoint 50% of the time is spent in each, so performance is cut in half
- In effect, the curves are smoothed
- Common for bulk synchronous MPI communication, atypical for DRAM access on modern architectures
Roofline model for Opteron
(non-overlapped)

- Not typical of multi-thread/core architectures
- Not typical of architectures with ooo or HW prefetchers
- More common in network accesses
Auto-tuned Performance
(+Cell/SPE version)

- Wrote a double precision Cell/SPE version
- DMA, local store blocked, NUMA aware, etc...
- Only 2x1 and larger BCOO
- Only the SpMV-proper routine changed

- About 12x faster (median) than using the PPEs alone.

Intel Xeon (Clovertown)

AMD Opteron (rev.F)

Sun Niagara2 (Huron)

IBM Cell Blade (SPEs)

- More DIMMs(opteron), FW fix, array padding(N2), etc...
- Cache/TLB Blocking
- Compression
- SW Prefetching
- NUMA/Affinity
- Naïve Pthreads
- Naïve
Auto-tuned Performance
(Local Store Implementation)

- First attempt at cell implementation.
- VL, unrolling, reordering fixed
- No NUMA
- Exploits DMA and double buffering to load vectors
- Straight to SIMD intrinsics.
- Despite the relative performance, Cell's DP implementation severely impairs performance

Intel Xeon (Clovertown)

AMD Opteron (rev.F)

Sun Niagara2 (Huron)

IBM Cell Blade (SPEs)*

*collision() only

- +SIMDization
- +SW Prefetching
- +Unrolling
- +Vectorization
- +Padding
- Naïve+NUMA
Where do GPUs fit in?

<table>
<thead>
<tr>
<th>Instruction Level Parallelism</th>
<th>Data Level Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressed at compile time</td>
<td>VLIW</td>
</tr>
<tr>
<td>Discovered at run time</td>
<td>superscalar, SMT, etc…</td>
</tr>
<tr>
<td>Data Level Parallelism</td>
<td>SIMD, Vector</td>
</tr>
<tr>
<td></td>
<td>G80</td>
</tr>
</tbody>
</table>

- GPUs discover data level parallelism from thread level parallelism at runtime.