Advanced Systems Lab
Spring 2021
Lecture: Roofline model

Instructor: Markus Püschel, Ce Zhang
TA: Joao Rivera, several more

Operational Intensity Again

Definition: Given a program P, assume cold (empty) cache

Operational intensity: \[ I(n) = \frac{W(n)}{Q(n)} \]

Asymptotic bounds on \( I(n) \)
- Vector sum: \( y = x + y \) \( O(1) \)
- Matrix-vector product: \( y = Ax \) \( O(1) \)
- Fast Fourier transform \( O(\log(n)) \) \( O(\log(y)) \) (not explained)
- Matrix-matrix product: \( C = AB + C \) \( O(n) \) \( O(\sqrt{\gamma}) \)

Cache lecture
\( \gamma = \text{size LLC (last level cache)} \)
Known to be optimal
Compute/Memory Bound

A function/piece of code is:
- **Compute bound** if it has high operational intensity
- **Memory bound** if it has low operational intensity

The roofline model makes this more precise

---

Roofline model/plot **(Williams et al. 2008)**

Platform model

<table>
<thead>
<tr>
<th>mem</th>
<th>Bandwidth ( \beta ) ← carefully measured (bytes/cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>( \beta ) raw bandwidth from manual unattainable (maybe 60% is)</td>
</tr>
</tbody>
</table>

Each with peak performance \( \pi \) [flops/cycle]

Algorithm/program model (\( n \) is the input size)

<table>
<thead>
<tr>
<th>Work</th>
<th>( W(n) ) [flops]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data movement</td>
<td>( Q(n) ) [bytes]</td>
</tr>
<tr>
<td>Runtime</td>
<td>( T(n) ) [cycles]</td>
</tr>
</tbody>
</table>

Derived:

- Operational intensity: \( I(n) = W(n)/Q(n) \) [flops/byte]
- Performance: \( P(n) = W(n)/T(n) \) [flops/cycle]

Example: one core, \( \pi = 2 \), \( \beta = 1 \), no SIMD

Bound based on \( \beta \):

\[ \beta \geq \frac{Q}{T} = \frac{(W/T)/(W/Q)}{P/I} = P/I \]

In log scale: \( \log_2(P) \leq \log_2(\beta) + \log_2(I) \)
Roofline Plots

What happens if we introduce 4-way SIMD?

If $\beta$ does not change: more programs become memory bound.

Roofline Plots

What if a program has an uneven mix of operations (e.g., 20% mults and 80% adds)?

A tighter roof may hold for this program (depends on units and ports).
Roofline Measurements

Tool developed in our group (code may need an update)
(G. Ofenbeck, R. Steinmann, V. Caparros-Cabezas, D. Spampinato)
http://www.spiral.net/software/roofline.html

Example plots follow

Estimate operational intensity $I = W/Q$ (cold cache):

- **daxpy**: $y = \alpha x + y$  
  $W = 2n$  
  $Q = 3n$ doubles = $24n$ bytes  
  $I = 1/12$

- **dgemv**: $y = Ax + y$  
  $W = 2n^2$  
  $Q = n^2$ doubles = $8n^2$ bytes  
  $I = 1/4$

- **dgemm**: $C = AB + C$  
  $W = 2n^3$  
  $Q \geq 4n^2$ doubles = $32n^2$ bytes  
  $I \leq n/16$

- **FFT**

Note:

- For daxpy and dgemv, $Q$ is determined by compulsory misses.
- For dgemm, more misses than compulsory misses occur for larger sizes.
  If $3n^2 \leq \gamma$ (cache size), equality should hold above.

Core i7 Sandy Bridge, 6 cores
Code: Intel MKL, sequential
Cold cache

What happens when we go to parallel code?
Roofline Measurements

What happens when we measure with warm cache?

Core i7 Sandy Bridge, 6 cores
Code: Intel MKL, parallel
Cold cache

Core i7 Sandy Bridge, 6 cores
Code: Intel MKL, sequential
Warm cache
Roofline Measurements

**Performance [Flops/Cycle]**

- Peak mp: (280 Flops/Cycle)
- Peak seq: (80 Flops/Cycle)
- Roofline 1: (44 Flops/Cycle)
- Roofline 2 seq: (35 Flops/Cycle)
- Roofline 3: (20 Flops/Cycle)
- Roofline 4: (10 Flops/Cycle)

**Operational Intensity [Flops/Byte]**

- Cold cache

**MMM: Different implementations**

---

Generalized Roofline Model

**Performance [Flops/Cycle]**

**Operational Intensity [Flops/Byte]**

Website and tool: [https://acl.inf.ethz.ch/research/ERM/](https://acl.inf.ethz.ch/research/ERM/)

Summary

Roofline plots distinguish between memory and compute bound
Can be used on for back-of-the-envelope computations on paper
Measurements difficult (performance counters) but doable
Interesting insights: *use in your project!*