Advanced Systems Lab
Spring 2024
Lecture: Roofline model

Instructor: Markus Püschel
TA: Tommaso Pegolotti, 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 \( O(\log(n)) \)
- Matrix-matrix product: \( C = AB + C \) \( O(n) \)

\[ \gamma = \text{size LLC (last level cache)} \]
Known to be optimal

\[ O(\sqrt{\gamma}) \] (not explained)
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

- **mem**
  - Bandwidth $\beta$ [bytes/cycle]
  - carefully measured
  -raw bandwidth from manual unattainable (maybe 60% is)
  - Stream benchmark may be conservative
- **cache**
- $p$ cores
- Each with peak performance $\pi$ [flops/cycle]

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

- **Work** $W(n)$ [flops]
- **Data movement** $Q(n)$ [bytes]
- **Runtime** $T(n)$ [cycles]

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

$$P(n) \leq \pi \quad \text{bound based on } \pi$$

$$\beta \geq Q/T \Rightarrow (W/T)/(W/Q) = P/I$$

In log scale:

$$\log_2(P) \leq \log_2(\beta) + \log_2(I)$$

---

$log$-$log$ scale!
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: $y = \alpha x + y$ \hspace{1cm} $W = 2n$ \hspace{1cm} $Q = 3n$ doubles = 24n bytes \hspace{1cm} $I = 1/12$
- dgemv: $y = Ax + y$ \hspace{1cm} $W = 2n^2$ \hspace{1cm} $Q = n^2$ doubles = 8n$^2$ bytes \hspace{1cm} $I = 1/4$
- dgemm: $C = AB + C$ \hspace{1cm} $W = 2n^3$ \hspace{1cm} $Q \geq 4n^2$ doubles = 32n$^2$ bytes \hspace{1cm} $I \leq n/16$
- 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

Changes the definition of Q to count all loads from the memory hierarchy, i.e., from memory and all levels of caches.

Shows throughput bounds for all levels of caches

Intel Advisor:
- Uses CARM as standard
- Provides the standard model, called MLR, but has to be used properly, e.g., you have to ensure cold cache and not include initialization
- Examples: next slides

In your project:
- Do measurements yourself, or
- Use Intel Advisor but choose MLR and use properly to reflect the standard roofline model, and explain in report how you did it

CARM, warm cache, including initialization

not good

4 roofs for L1, L2, L3 cache, memory

daxpy: yellow
dgemv: green
dgemm: red

CARM, cold cache, excluding initialization

don’t use

additional loads from caches decrease I

similar to standard model since all loads are from memory
MLR, cold cache, excluding initialization
standard roofline model, done well

Generalized Roofline Model

Website and tool: https://acl.inf.ethz.ch/research/ERM/

Summary

Roofline plots distinguish between memory and compute bound

Can be instantiated for different scenarios (e.g., computation on GPU, data in CPU memory)

Can be used on for back-of-the-envelope computations on paper

Measurements difficult (performance counters) but doable

Easier variant just consider reads: $Q_{\text{read}}, \beta_{\text{read}}$

Interesting insights: use in your project, but if so, do properly!