Lecture 5: Announcements

- HW1

- HW2: this w/e/early next + 2 w

- talk assignment
  - submit Flow
  - assignment
Case study: Matrix-Matrix Mult. (‘MMM’): Code Structure

- How would you structure a high-performance MMM?
- What are sources of concurrency?
- What should you consider your working set?

Why 4 memory ops?

Loop slicing?
- multiple cores/task
- SIMD/data
- L1P
Case study: Matrix-Matrix Mult. (’MMM’) via Latency

Come up with a simple cost model for MMM in a two-level hierarchy based on latency:

\[ \text{Miss rate:} \quad \frac{3}{4 N_b \cdot CLS} \]

[Yotov et al. ’07]
Case study: Matrix-Matrix Mult. ('MMM') via Bandwidth

Come up with a cost model for MMM in a two-level hierarchy based on bandwidth:

Reg. mem. bw. \( \frac{128}{N_0} \) flops/cycle

[Yotov et al. '07]
Case study: Matrix-Matrix Mult. ('MMM'): Discussion

**Discussion:** What are the main simplifications in each model?

![Diagram showing miss rate/latency and bandwidth simplifications. Miss rate/latency:
- high bw

Bandwidth:
- latency
- prefetch

[Yotov et al. '07]

**General Q:** How can we analyze cache cost of algorithms in general?
Hong/Kung: Red/Blue Pebble Game

Simple means of I/O cost analysis: “Red/blue pebble game”

► A way to quantify I/O cost on a DAG (why a DAG?)
► “Red Hot” pebbles: data that can be computed on
► “Blue Cool” pebbles: data that is stored, but not available for computation without I/O

Note: Can allow “Red/Purple/Blue/Black”: more levels

Q: What are the cost metrics in this model?

- Heating up a pebble
- Cooling down a pebble
- max # of red pebbles

[Hong/Kung ‘81]
Cache-Oblivious Algorithms

Annoying chore: Have to pick multiple machine-adapted block sizes in cache-adapted algorithms, one for each level in the memory hierarchy, starting with registers.

Idea:

- Step 1: Express algorithm recursively in divide & conquer manner
- Step 2: Pick a strategy to decrease block size

Give examples of block size strategies, e.g. for MMM:

\[
\begin{align*}
\text{All dimensions} & \rightarrow \text{Split} \\
\text{longest dim.} & \rightarrow \text{Split}
\end{align*}
\]

Result:

- Asymptotically optimal on Hong/Kung metric
Cache-Oblivious Algorithms: Issues

What are potential issues on actual hardware?

- function call overhead
- register allocation

With good base case:
- Instruction cache
- Instruction

[Yotov et al. ’07]
Recall: Big-O Notation

Classical Analysis of Algorithms (e.g.):

\[ \text{Cost}(n) = O(n^3). \quad (\text{as } n \to \infty) \]

Precise meaning? Anything missing from that statement?

There exists a \( C \),

\[
\text{for all } n \geq k \quad \text{Cost}(n) \leq C \cdot n^3
\]
Comments on asymptotic statements about cost in relation to high performance?

- No statement about finite $n$
- No statement about the constant

Net effect: Having an understanding of asymptotic cost is necessary, but not sufficient for high performance.

HPC is in the business of minimizing $C$ in:

$$\text{Cost}(n) \leq C \cdot n^3 \quad (\text{for all } n)$$
Alignment describes the process of matching the base address of:

- Single word: double, float
- SIMD vector
- Larger structure

To machine granularities:

Q: What is the performance impact of misalignment?
Performance Impact of Misalignment

- Matched structure

[Diagram showing misaligned and matched structures]
SIMD: Basic Idea

What’s the basic idea behind SIMD?

What architectural need does it satisfy?

- Insufficient decode/dispatch bw

Typically characterized by width of data path:

- SSE: 128 bit (4 floats, 2 doubles)
- AVX-2: 256 bit (8 floats, 4 doubles)
- AVX-512: 512 bit (16 floats, 8 doubles)
SIMD: Architectural Issues

Realization of inter-lane comm. in SIMD? Find instructions:

- broadcast
- permute
- reduce
- interleave

Name tricky/slow aspects in terms of expressing SIMD:

x86 SIMD suffixes: What does the “ps” suffix mean? “sd”?
SIMD: Transposes

Why are transposes important? Where do they occur?

Example implementation aspects:
- HPTT: [Springer et al. ‘17]
- github: springer13/hptt 8x8 transpose microkernel
- Q: Why 8x8?