Lecture 6: Announcements

- Talk assignment
- HW 1
- Computing
SIMD: Basic Idea

What’s the basic idea behind SIMD?

What architectural need does it satisfy?

- Flow control (Masking, recombine collectives)
- Gather/scaffer

- 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.

Name tricky/slow aspects in terms of expressing SIMD:

(see prev. slide)

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

ps → packed single
sd → scalar double
SIMD: Transposes

Why are transposes important? Where do they occur?

- Whenever there is a mismatch in (regular) data layout preventing SIMD, transposes help.

Example implementation aspects:

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

Introduction
   Notes
   About This Class
   Why Bother with Parallel Computers?
   Lowest Accessible Abstraction: Assembly
   Architecture of an Execution Pipeline
   Architecture of a Memory System
   **Shared-Memory Multiprocessors**

Machine Abstractions

Performance: Expectation, Experiment, Observation

Performance-Oriented Languages and Abstractions

Program Representation and Transformation
Multiple Cores vs Bandwidth

Assume (roughly right for Intel):

- memory latency of 100 ns
- peak DRAM bandwidth of 50 GB/s (per socket)

How many cache lines should be/are in flight at one time?

\[
100 \times 80 \frac{\text{GB}}{s} = 5000 \text{ B}
\]

\[
\approx 80 \text{ cache lines}
\]

\[
10 \text{ cache lines}
\]

[McCalpin ‘18]
Topology and NUMA

[SuperMicro Inc. ‘15]
Demo: Show lstopo on porter, from hwloc.
Placement and Pinning

Who decides on what core my code runs? How?

OMP_PLACES= cores
pthreads

Who decides on what NUMA node memory is allocated?

'first-touch policy'

Demo: intro/NUMA and Bandwidths
What is the main expense in NUMA?

Latency
Cache Coherence

What is cache coherence?
- Caching hits' inconsistency
- Coherence describes guarantees

How is cache coherence implemented?
- Old directory
- Newer snooping
  M E S I

What are the performance impacts?
- **Demo: intro/Threads vs Cache**
- **Demo: intro/Lock Contention**
'Conventional' vs Atomic Memory Update

Read → Increment → Write
Interruptible! → Interruptible!

Read → Increment → Write
Protected → Protected
Outline

Introduction

Machine Abstractions
  C
  OpenCL/CUDA
  Convergence, Differences in Machine Mapping
  Lower-Level Abstractions: SPIR-V, PTX

Performance: Expectation, Experiment, Observation

Performance-Oriented Languages and Abstractions

Program Representation and Transformation

Polyhedral Representation and Transformation
Outline

Introduction

Machine Abstractions
  C
  OpenCL/CUDA
  Convergence, Differences in Machine Mapping
  Lower-Level Abstractions: SPIR-V, PTX

Performance: Expectation, Experiment, Observation

Performance-Oriented Languages and Abstractions

Program Representation and Transformation

Polyhedral Representation and Transformation
Atomic Operations: Compare-and-Swap

```c
#include <stdatomic.h>
_Bool atomic_compare_exchange_strong(
  volatile A* obj,
  C* expected, C desired
);
```

What does `volatile` mean?

What does this do?

How might you use this to implement atomic FP multiplication?