Lecture 18:

Accessing Memory +
Course Wrap Up

Parallel Computing
Stanford CS149, Fall 2023
Today

- Technical material: accessing memory / how DRAM works
- Course wrap up
- A few comments on possibilities for next steps
Accessing Memory
(a basic tutorial on how DRAM works)
The memory system

- CPU
  - issues memory requests to memory controller
  - issues loads and store instructions
- Memory Controller
  - sends commands to DRAM
- Last-level cache (LLC)
- DRAM
  - 64 bit memory bus

Stanford CS149, Fall 2023
DRAM array

1 transistor + capacitor per “bit” (recall from physics: a capacitor stores charge)

2 Kbits per row

Row buffer (2 Kbits)

Data pins (8 bits)

(to memory controller...)

Stanford CS149, Fall 2023
DRAM operation (load one byte)

We want to read this byte

1. Precharge: ready bit lines (~10 ns)
2. Row activation (~10 ns)
3. Column selection (~10 ns)
4. Transfer data onto bus (to memory controller...)

Data pins (8 bits)
Row buffer (2 Kbits)

Estimated latencies are in units of memory clocks: DDR3-1600 (Kayvon's laptop at the time of making this slide)
Load next byte from (already active) row

Lower latency operation: can skip precharge and row activation steps

2 Kbits per row

1. Column selection
2. Transfer data onto bus

~ 10 ns

(to memory controller...)
DRAM access latency is not fixed

- **Best case latency: read from active row**
  - Column access time (CAS)

- **Worst case latency: bit lines not ready, read from new row**
  - Precharge (PRE) + row activate (RAS) + column access (CAS)

Precharge readies bit lines and writes row buffer’s contents back into DRAM array (reading a row is destructive)
Problem: low pin utilization due to latency of access

Data pins in use only a small fraction of time
(red = data pins busy)

This is bad since they are the scarcest resource!
DRAM burst mode

Idea: amortize latency over larger transfers

Each DRAM command describes bulk transfer

Bits placed on output pins in consecutive clocks

Data pins (8 bits)
DRAM chip consists of multiple banks

- All banks share same pins (only one transfer at a time)
- Banks allow for pipelining of memory requests
  - Precharge/activate rows/send column address to one bank while transferring data from another
  - Achieves high data pin utilization
Organize multiple chips into a DIMM

Example: Eight DRAM chips (64-bit memory bus)

Note: DIMM appears as a single, higher capacity, wider interface DRAM module to the memory controller. Higher aggregate bandwidth, but minimum transfer granularity is now 64 bits.
Reading one 64-byte (512 bit) cache line (the wrong way)

Assume: consecutive physical addresses mapped to same row of same chip
Memory controller converts physical address to DRAM bank, row, column
Reading one 64-byte (512 bit) cache line (the wrong way)

All data for cache line serviced by the same chip
Bytes sent consecutively over same pins (8 bits per clock —> cache line takes 64 cycles to transfer!)
Reading one 64-byte (512 bit) cache line (the wrong way)

All data for cache line serviced by the same chip
Bytes sent consecutively over same pins (8 bits per clock —> cache line takes 64 cycles to transfer!)
Reading one 64-byte (512 bit) cache line (efficient way)

Memory controller converts physical address to DRAM bank, row, column
Here: physical addresses are interleaved across DRAM chips at byte granularity
DRAM chips transmit first 64 bits in parallel (cache line takes 8 clocks to transfer)

Memory controller

Last-level cache (LLC)

CPU
Reading one 64-byte (512 bit) cache line (efficient way)

DRAM controller requests data from new column *
DRAM chips transmit next 64 bits in parallel

* Recall modern DRAM’s support burst mode transfer of multiple consecutive columns, which would be used here
Memory controller is a scheduler of memory requests

- Receives load/store requests from LLC

- Conflicting scheduling goals
  - Maximize throughput, minimize latency, minimize energy consumption
  - Common scheduling policy: FR-FCFS (first-ready, first-come-first-serve)
    - Service requests to currently open row first (maximize row locality)
    - Service requests to other rows in FIFO order
  - Controller may coalesce multiple small requests into large contiguous requests (to take advantage of DRAM “burst modes”)

64 bit memory bus (to DRAM)

Requests from system’s last level cache (e.g., L3)
Dual-channel memory system

- Increase throughput by adding memory channels (effectively widen memory bus)
- Below: each channel can issue independent commands
  - Different row/column is read in each channel
  - Simpler setup: use single controller to drive same command to multiple channels
Example: DDR4 memory

Processor: Intel® Core™ i7-7700K Processor (in Myth cluster)

Memory system details from Intel's site:

<table>
<thead>
<tr>
<th>Memory Specifications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Memory Size (dependent on memory type)</td>
<td>64 GB</td>
</tr>
<tr>
<td>Memory Types</td>
<td>DDR4-2133/2400, DDR3L-1333/1600 @ 1.35V</td>
</tr>
<tr>
<td>Max # of Memory Channels</td>
<td>2</td>
</tr>
</tbody>
</table>

**DDR4 2400**
- 64-bit memory bus x 1.2GHz x 2 transfers per clock* = 19.2 GB/s per channel
- 2 channels = 38.4 GB/sec
- ~13 nanosecond CAS

* DDR stands for “double data rate”
DRAM summary

- DRAM access latency can depend on many low-level factors
  - Discussed today:
    - State of DRAM chip: row hit/miss? is recharge necessary?
    - Buffering/reordering of requests in memory controller

- Significant amount of complexity in a modern multi-core processor has moved into the design of memory controller
  - Responsible for scheduling ten’s to hundreds of outstanding memory requests
  - Responsible for mapping physical addresses to the geometry of DRAMs
  - Area of active computer architecture research
Modern architecture challenge: improving memory performance:

Decrease distance data must move by locating memory closer to processors

(enables shorter, but wider interfaces)
Increase bandwidth, reduce power by chip stacking

Enabling technology: 3D stacking of DRAM chips

- DRAMs connected via through-silicon-vias (TSVs) that run through the chips
- TSVs provide highly parallel connection between logic layer and DRAMs
- Base layer of stack “logic layer” is memory controller, manages requests from processor
- Silicon “interposer” serves as high-bandwidth interconnect between DRAM stack and processor

Technologies:
Micron/Intel Hybrid Memory Cube (HBC)
High-bandwidth memory (HBM) - 1024 bit interface to stack
GPUs are adopting HBM technologies

**AMD Radeon Fury GPU (2015)**
- 4096-bit interface: 4 HBM chips x 1024 bit interface per chip
- 512 GB/sec BW

**NVIDIA P100 GPU (2016)**
- 4096-bit interface: 4 HBM2 chips x 1024 bit interface per chip
- 720 GB/sec peak BW
- 4 x 4 GB = 16 GB capacity
The memory bottleneck is being addressed in many ways

- **By the application programmer**
  - Schedule computation to maximize locality, increase arithmetic intensity (minimize required data movement)

- **By new hardware architectures**
  - Intelligent DRAM request scheduling
  - Bringing data closer to processor (deep cache hierarchies, 3D stacking)
  - Increase bandwidth (wider memory systems)
  - Ongoing research in locating limited forms of computation “in” or near memory
  - Ongoing research in hardware accelerated compression (not discussed today)

- **General design principles**
  - Locate data storage near processor
  - Move computation to data storage
  - Data compression (trade-off extra computation for less data transfer)
Course Wrap Up

(Students)
For the foreseeable future, the primary way to obtain higher performance computing hardware is through a combination of increased parallelism and hardware specialization.

- **Intel Xeon Phi**
  - 72 cores
  - 16-wide SIMD
  - 4-way multi-threading

- **NVIDIA Maxwell GPU**
  - (single SMM core)
  - 32 wide SIMD
  - 2048 CUDA/core threads per SMM

- **Apple A9**
  - Heterogeneous SoC
  - multi-core CPU + multi-core GPU + media ASICs

- **FPGA**
  - (reconfigurable logic)
Today’s software is surprisingly inefficient compared to the capability of modern machines

A lot of performance is currently left on the table (increasingly so as machines get more complex, and parallel processing capability grows)

Extracting this performance stands to provide a notable impact on many compute-intensive fields (or, more importantly enable new applications of computing!)

Given current software programming systems and tools, understanding how a parallel machine works is important to achieving high performance.

A major challenge going forward is making it simpler for programmers to extract performance on these complex machines.
This is very important given how exciting (and efficiency-critical) the next generation of computing applications are likely to be.
Key issues we have addressed in this course

Identifying parallelism
(or conversely, identifying dependencies)

Efficiently scheduling parallelism

1. Achieving good workload balance
2. Overcoming communication constraints:
   - Bandwidth limits, dealing with latency, synchronization
   - Exploiting data/computation locality = efficiently managing state!
3. Scheduling under heterogeneity (using the right processor for the job)

We discussed these issues at many scales and in many contexts

- Heterogeneous mobile SoC
- Single chip, multi-core CPU
- Multi-core GPU
- CPU+GPU connected via bus
- Clusters of machines
- Large scale, multi-node supercomputers
Key issues we have addressed in this course

Abstractions for thinking about efficient code

Data parallel thinking
Functional parallelism
Transactions
Tasks

How throughput-oriented hardware works

Multiple cores, hardware-threads, SIMD
Specialization to key domains
Next steps
Two cool classes

CS 217: Hardware Accelerators for Machine Learning (Winter, Kunle)

Focuses on design of specialized hardware architectures for ML (understanding the workload and building efficient hardware for that workload)

CS 348K: Visual Computing Systems (Spring, Kayvon)

Design of high-performance hardware/software systems for processing images and video (ray tracing, video analysis, smartphone camera processing, NeRF/Al-based graphics, fast data labeling, etc)
After taking this course, you might be able to play a role in ongoing Stanford research in parallel computing!

Come talk to us!
Why research (or independent study)?

▪ You will learn way more about a topic than in any class.

▪ Think your undergrad/MS peers are amazingly smart? So are Ph.D. students! (you get to work side-by-side with them and with faculty). Imagine what level you might rise to.

▪ It’s fun to be on the cutting edge. Industry might not even know about what you are working on. (imagine how much more valuable you are if you can teach them)

▪ It widens your mind as to what might be possible
A parallel programming problem in my lab:
AI Agents Learn by Huge Amounts of Trial and Error!

Szot et al, NeurIPS 2021
Large Scale Agent Training Is Expensive!

OpenAI Five

Home Robotics

Hide and Seek

Learning Dota 2: Months of training

Billions of samples

High-level strategies emerge after billions of timesteps

100K+ CPUs, 100s GPUs

64 GPUs
Claim:
We Need a Game Engine for Building Batch Simulators!
Zoom Out Example Simulator:
HYPOTHESIS:

CS classes alone may not be the most effective way to maximize your experience at Stanford and opportunities afterward.

It may not be the best way to get a competitive job.

It may not be the best way to get the coolest jobs.

It may not be the best way to prepare yourself to have the most impact in a future career.
A conventional path…

CS student

DOES NOT SLEEP in order to do well in MANY CS classes

Even more impressive resume handed out at CS job fair

Resume gets student first-round interview

Student knows their stuff in interview (aces fine-grained linked list locking question)

GOOD JOB
Woot!
An alternative path…

Amazing CS student

Takes fewer classes, but does some crazy extra credits in CS149. (really interested in parallel programming)

Student: “Hey Kayvon, I liked your class, is there anything I can help with in your research group next semester?”

Kayvon: “Yo! You did great in the class. I loved that extra credit you did. You should totally come help with this project in my group.”

Student gets awesome experience working side-by-side with Stanford Ph.D. students and professors. Learns way more than in class.

Kayvon, to friend in industry: “Hey, you’ve got to hire this student, they know more about parallel architecture than any undergrad in the country. They’ve been doing publishable research on it.”

WICKED GOOD JOB

Woot!

Stanford CS149, Fall 2023
HOW TO GET STARTED:

Common scenario:
Student: “I’m really interested in parallel systems research, is there anything I could do in your lab?”

Kayvon’s reaction: will this be a good fit for the student?
To do well in a research lab setting, the student must be both highly motivated, have some background in the area to help them succeed, and be willing to at least put real time into it for the quarter.

 Have they taken CS149?  
Are there examples of them going beyond expectations on programming projects in CS149?  
Have they worked on anything of the sort before in related classes or internships?
Common pitfalls 😞

- **Pitfall:** Student emails me after last class: “I’m really interested in going farther” — then never gets a response from me
  - Solution: keep emailing! And especially email again in the week before the next quarter.

- **Pitfall:** Can’t find a project fit right now. Typically the best way to get started in a lab is play an engineering role on a project led by one of my Ph.D. students
  - e.g., implement a feature that the Ph.D. student wants to see tried, but doesn’t have time to do
  - Implement an existing algorithm and carry out a measurement/experiment
  - Not all projects are in a phase where these types of side projects are possible
Think bigger + broader

You are fortunate.
You are smart, talented, and hard-working.
You are in an amazing environment at Stanford.

How can you maximize that opportunity while you are here?
The mechanisms are in place (or we’ll help you create them):

- Course projects
- Research
- Independent study
- Entrepreneurship

The biggest sign you are in the “real-world” isn’t when you are paying your own bills, showing up to work on time, or ensuring your code passes regressions... it is asking your own questions and making your own decisions.

And there’s a lot more to decide on than classes.
Or in other words... there are “grades” you can get at Stanford that are much higher than A’s and A+’s *

* taken from Dave Eckhardt
Thanks for being a great class!
Thanks for putting in the work.

Have a great break! Send me a post card!