What programmers should know about Memory

Shankar Viswanathan
BLU July, 2023
Agenda

• Memory types & hierarchy
  - DRAM
  - SRAM (used in caches)

• Brief description of:
  - Coherency
  - Consistency
What’s with the title?

- **Paper** by Ulrich Drepper from 2007
- Riff on earlier **paper** titled: “What Every Computer Scientist Should Know About Floating-Point Arithmetic” by David Goldberg
Computer Memory Hierarchy

- Power on
  - Processor registers very fast, very expensive
  - Processor cache very fast, very expensive
- Power on
  - Random access memory fast, affordable
- Power off
  - Flash / USB memory slower, cheap
- Power off
  - Hard drives slow, very cheap
- Power off
  - Tape backup very slow, affordable
- Power off
  - Large size very large capacity
- Power off
  - Medium size medium capacity
- Power off
  - Small size small capacity
- Power on
  - Immediate term
- Power on
  - Small size small capacity
Agenda

• Memory types & hierarchy
  – DRAM
  – SRAM (used in caches)

• Brief description of:
  – Coherency
  – Consistency
DRAM

1T-1C DRAM cell

Multi-bank DRAM array

Source: https://www.allaboutcircuits.com/technical-articles/introduction-to-dram-dynamic-random-access-memory/
Types of DRAM

- DDR
- LPDDR
- GDDR
- HBM
Memory interleaving

- Channel interleave
- Rank interleave
- Bank interleave
UMA vs NUMA
DRAM timings

- tCL
- tRCD
- tRP
- tRAS
- tCCD
- tRRD
- tRTW
- ...
- Alphabet soup continues!
DRAM Write Operation

DRAM Read Operation

DDR DRAM Evolution

Memory Capacity Improvement

Memory Interface Improvement

JEDEC

7
Challenge in Memory Controller Design

- Goal is to maximize utilization of DRAM data bus
- Manage all these timing constraints
- Balance requirements of read latency vs. throughput
  - CPUs are latency sensitive, GPUs are BW hungry
- Satisfy QoS requirements of any real-time clients
  - Audio & Display controllers have strict requirements
- Manage other high-level goals: fairness, power efficiency, etc.
Agenda

- Memory types & hierarchy
  - DRAM
  - SRAM (used in caches)
- Brief description of:
  - Coherency
  - Consistency
SRAM

6T SRAM cell

SRAM array
Caches

- SRAM array used to hold most frequently accessed data in the processor
- Large area devoted to caches
Cache organization

- Small fast L1 instruction and data caches
- Larger L2 cache per CPU core
- Shared L3 cache over multiple cores
- Misses from L3 go to DRAM
- Some designs can have separate LLC (last level cache) that can be shared by other subsystems in the chip (e.g. GPU, NPU etc.)
Cache structure

Source: Computer Architecture by Hennessey & Patterson
Side bar: TLBs are also caches

Source: https://www.geeksforgeeks.org/caching-page-tables/
3D stacked cache

AMD 3D CHIPLET TECHNOLOGY

- Structural silicon
- 64MB L3 cache die
- Direct copper-to-copper bond
- Through Silicon Vias (TSVs) for silicon-to-silicon communication
- Up to 8-core "Zen 3" CCD

A PACKAGING BREAKTHROUGH FOR HIGH-PERFORMANCE COMPUTING

© WikiChip
Hybrid bonding

Source: https://www.techpowerup.com/review/amd-ryzen-7-5800x3d/2.html
Agenda

• Memory types & hierarchy
  – DRAM
  – SRAM (used in caches)

• Brief description of:
  – Coherency
  – Consistency
Cache coherency

- Hardware ensures caches are in sync with each other when one processor updates the value at a particular address.
Memory Consistency

- Coherence deals with multiple accesses to the same address.
- Consistency has to do with ordering of accesses to different addresses.

```plaintext
CPU0
data = 1;
flag = 1;
print(data);

CPU1
while(flag != 1);
print(data);
```

If data = flag = 0 initially, what gets printed?
Consistency explained

- A consistency model specifies a contract between the programmer and a system, wherein the system guarantees that if the programmer follows the rules for operations on memory, memory will be consistent and the results of reading, writing, or updating memory will be predictable.

- L. Lamport defined a basic consistency model called “Sequential Consistency” in which every thread executes operations in strict order. This greatly limits performance and thus no major ISA implements it.

- Different consistency models therefore define the extent to which R → R, W → R, R → W, and W → W orderings are maintained.
Popular ISA Consistency models

- x86/AMD64: a flavor of TSO (total store ordering) – stores from the same thread appear in order
  - Previous example code would always print data = 1
- ARM: weak ordering, allows any reordering
  - data = 0 or data = 1 are both possible
- RISC-V: RVWMO (weak memory ordering), also allows any reordering
I’m a programmer and I sat through your boring talk so far.

Why does any of this matter?
I’m a programmer: why does any of this matter?

• Code performance is very dependent on caches and memory system. C code:
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < M; j++) {
      C[i][j] = A[i][j] + B[i][j]; → good performance
      // C[j][i] = A[j][i] + B[j][i]; → bad! (good in Fortran though)
    }
  }

• LLMs have billions of parameters, partitioning the problem and knowing what should and should not be cached is crucial to achieve high performance

• Graphics shaders or GPGPU compute kernels have to be very aware of DRAM layout and timings to extract maximum BW utilization
I’m a programmer: why does any of this matter? (2)

- Compilers and any hand-optimized code have to deal with consistency model
  - Fences have to be inserted based on target architecture
  - But unnecessary fences will kill performance

- Language runtimes and virtual machines (like JVM) can impose their own behavior
And please don’t orphan your memory channels

Thank you!