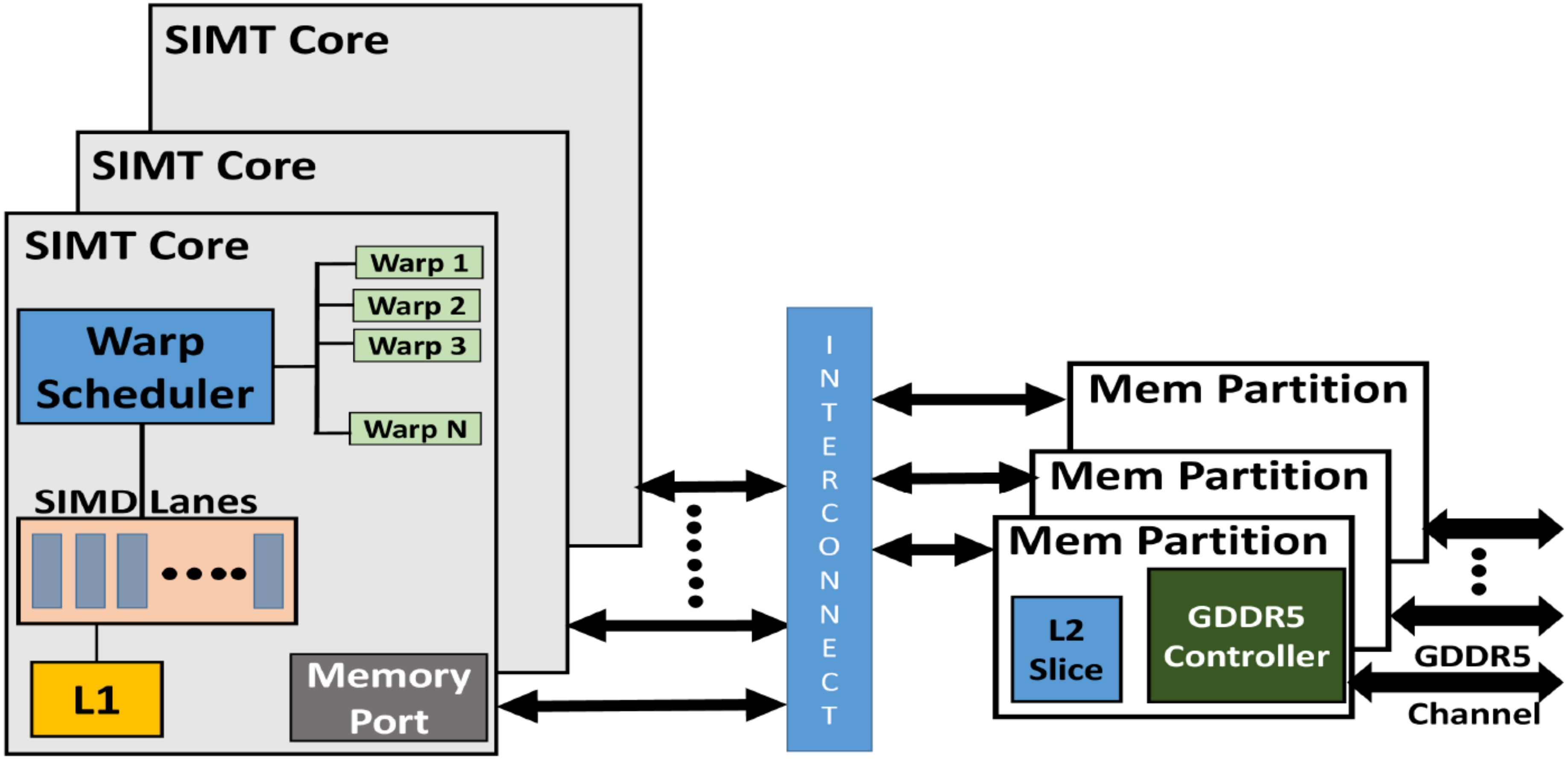
Lecture 28: Reliability

- Today's topics:
  - GPU wrap-up
  - Disk basics
  - RAID
  - Research topics
  - Review

The GPU Architecture



Architecture Features

- Simple in-order pipelines that rely on thread-level parallelism to hide long latencies

- Many registers (~1K) per in-order pipeline (lane) to support many active warps

- When a branch is encountered, some of the lanes proceed along the “then” case depending on their data values; later, the other lanes evaluate the “else” case; a branch cuts the data-level parallelism by half (branch divergence)

- When a load/store is encountered, the requests from all lanes are coalesced into a few 128B cache line requests; each request may return at a different time (mem divergence)

GPU Memory Hierarchy

- Each SIMT core has a private L1 cache (shared by the warps on that core)
- A large L2 is shared by all SIMT cores; each L2 bank services a subset of all addresses
- Each L2 partition is connected to its own memory controller and memory channel
- The GDDR5 memory system runs at higher frequencies, and uses chips with more banks, wide IO, and better power delivery networks
- A portion of GDDR5 memory is private to the GPU and the rest is accessible to the host CPU (the GPU performs copies)

Role of Disks

- Activities external to the CPU/memory are typically orders of magnitude slower
- Example: while CPU performance has improved by 50% per year, disk latencies have improved by 10% every year
- Typical strategy on I/O: switch contexts and work on something else
- Other metrics, such as bandwidth, reliability, availability, and capacity, often receive more attention than performance

Magnetic Disks

- A magnetic disk consists of 1-12 *platters* (metal or glass disk covered with magnetic recording material on both sides), with diameters between 1-3.5 inches

- Each platter is comprised of concentric *tracks* (5-30K) and each track is divided into *sectors* (100 – 500 per track, each about 512 bytes)

- A movable arm holds the read/write heads for each disk surface and moves them all in tandem – a *cylinder* of data is accessible at a time

Disk Latency

- To read/write data, the arm has to be placed on the correct track – this *seek time* usually takes 5 to 12 ms on average – can take less if there is spatial locality
- *Rotational latency* is the time taken to rotate the correct sector under the head – average is typically more than 2 ms (15,000 RPM)
- *Transfer time* is the time taken to transfer a block of bits out of the disk and is typically 3 – 65 MB/second
- A disk controller maintains a disk cache (spatial locality can be exploited) and sets up the transfer on the bus (*controller overhead*)

Defining Reliability and Availability

- A system toggles between
    - Service accomplishment: service matches specifications
    - Service interruption: service deviates from specs
- The toggle is caused by *failures* and *restorations*
- Reliability measures continuous service accomplishment and is usually expressed as mean time to failure (MTTF)
- Availability measures fraction of time that service matches specifications, expressed as MTTF / (MTTF + MTTR)

RAID

- Reliability and availability are important metrics for disks
- RAID: redundant array of inexpensive (independent) disks
- Redundancy can deal with one or more failures
- Each sector of a disk records check information that allows it to determine if the disk has an error or not (in other words, redundancy already exists within a disk)
- When the disk read flags an error, we turn elsewhere for correct data

RAID 0 and RAID 1

- RAID 0 has no additional redundancy (misnomer) – it uses an array of disks and stripes (interleaves) data across the arrays to improve parallelism and throughput

- RAID 1 mirrors or shadows every disk – every write happens to two disks

- Reads to the mirror may happen only when the primary disk fails – or, you may try to read both together and the quicker response is accepted

- Expensive solution: high reliability at twice the cost

RAID 3

- Data is bit-interleaved across several disks and a separate disk maintains parity information for a set of bits
- For example: with 8 disks, bit 0 is in disk-0, bit 1 is in disk-1, …, bit 7 is in disk-7; disk-8 maintains parity for all 8 bits
- For any read, 8 disks must be accessed (as we usually read more than a byte at a time) and for any write, 9 disks must be accessed as parity has to be re-calculated
- High throughput for a single request, low cost for redundancy (overhead: 12.5%), low task-level parallelism

RAID 4 and RAID 5

- Data is block interleaved – this allows us to get all our data from a single disk on a read – in case of a disk error, read all 9 disks

- Block interleaving reduces thruput for a single request (as only a single disk drive is servicing the request), but improves task-level parallelism as other disk drives are free to service other requests

- On a write, we access the disk that stores the data and the parity disk – parity information can be updated simply by checking if the new data differs from the old data

RAID 5

- If we have a single disk for parity, multiple writes can not happen in parallel (as all writes must update parity info)
- RAID 5 distributes the parity block to allow simultaneous writes

RAID Summary

- RAID 1-5 can tolerate a single fault – mirroring (RAID 1) has a 100% overhead, while parity (RAID 3, 4, 5) has modest overhead
- Can tolerate multiple faults by having multiple check functions – each additional check can cost an additional disk (RAID 6)
- RAID 6 and RAID 2 (memory-style ECC) are not commercially employed

Memory Protection

- Most common approach: SECDED – single error correction, double error detection – an 8-bit code for every 64-bit word -- can correct a single error in any 64-bit word – also used in caches

- Extends a 64-bit memory channel to a 72-bit channel and requires ECC DIMMs (e.g., a word is fetched from 9 chips instead of 8)

- Chipkill is a form of error protection where failures in an entire memory chip can be corrected

Computation Errors – TMR

- Errors in ALUs and cores are typically handled by performing the computation n times and voting for the correct answer

- n=3 is common and is referred to as triple modular redundancy

Future Innovations

- Accelerators
- Handling big data applications with near-data processing
- New memory technologies
- Security

Review Topics

- Finite state machines
- Pipelines: performance, control hazards, data hazards
- Out-of-order execution
- Caches
- Memory system, virtual memory
- Cache coherence
- Synchronization, consistency, programming models
- GPUs
- Reliability