Lecture: Storage, GPUs

- Topics: disks, RAID, reliability, GPUs (Appendix D, Ch 4)
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*)
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 cannot happen in parallel (as all writes must update parity info)

- RAID 5 distributes the parity block to allow simultaneous writes
Other Reliability Approaches

• High reliability is also expected of memory systems; many memory systems offer SEC-DED support – single error correct, double error detect; implemented with an 8-bit code for every 64-bit data word on ECC DIMMs

• Some memory systems offer chipkill support – the ability to recover from complete failure in one memory chip – many implementations exist, some resembling RAID designs

• Caches are typically protected with SEC-DED codes

• Some cores implement various forms of redundancy, e.g., DMR or TMR – dual or triple modular redundancy
SIMD Processors

- Single instruction, multiple data
- Such processors offer energy efficiency because a single instruction fetch can trigger many data operations
- Such data parallelism may be useful for many image/sound and numerical applications
GPUs

- Initially developed as graphics accelerators; now viewed as one of the densest compute engines available

- Many on-going efforts to run non-graphics workloads on GPUs, i.e., use them as general-purpose GPUs or GPGPUs

- C/C++ based programming platforms enable wider use of GPGPUs – CUDA from NVidia and OpenCL from an industry consortium

- A heterogeneous system has a regular host CPU and a GPU that handles (say) CUDA code (they can both be on the same chip)
The GPU Architecture

• SIMT – single instruction, multiple thread; a GPU has many SIMT cores

• A large data-parallel operation is partitioned into many thread blocks (one per SIMT core); a thread block is partitioned into many warps (one warp running at a time in the SIMT core); a warp is partitioned across many in-order pipelines (each is called a SIMD lane)

• A SIMT core can have multiple active warps at a time, i.e., the SIMT core stores the registers for each warp; warps can be context-switched at low cost; a warp scheduler keeps track of runnable warps and schedules a new warp if the currently running warp stalls
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)
Advanced Courses

• For GPU architectures and programming, see Mary Hall’s CS 6235, Parallel Programming for Many-Core Arch

• Spr’13: CS 7810: Advanced Computer Architecture
  ▪ Mo/We 11:50am-1:10pm
  ▪ Core design, cache hierarchies, networks, memory systems, datacenters, etc.
  ▪ Major course project that evaluates original ideas with simulators (often leads to publications)
  ▪ One assignment
  ▪ Take-home final
Title

• Bullet