Databases

- Data Models
  - Conceptual representation of the data
- Data Retrieval
  - How to ask questions of the database
  - How to answer those questions
- Data Storage
  - How/where to store data, how to access it
- Data Integrity
  - Manage crashes, concurrency
  - Manage semantic inconsistencies
Query Processing/Storage

- Query Processing Engine
  - Given a input user query, decide how to “execute” it
  - Specify sequence of pages to be brought in memory
  - Operate upon the tuples to produce results

- Buffer Management
  - Bringing pages from disk to memory
  - Managing the limited memory

- Space Management on Persistent Storage (e.g., Disks)
  - Storage hierarchy
  - How are relations mapped to files?
  - How are tuples mapped to disk blocks?

Outline

- Storage hierarchy
- Disks
- RAID
- File Organization
- Etc....
Storage Hierarchy

- Tradeoffs between speed and cost of access

- Volatile vs nonvolatile
  - Volatile: Loses contents when power switched off

- Sequential vs random access
  - Sequential: read the data contiguously
    - select * from employee
  - Random: read the data from anywhere at any time
    - select * from employee where name like '__a__b'

- Why care?
  - Need to know how data is stored in order to optimize, to understand what’s going on

How important is this today?

- Trade-offs shifted drastically over last 10-15 years
  - Especially with fast network, SSDs, and high memories
  - However, the volume of data is also growing quite rapidly

- Some observations:
  - Cheaper to access another computer’s memory than local disk
  - Cache is playing more and more important role
  - Data often fits in memory of a single machine, or cluster of machines

  - “Disk” considerations less important
    - Still: Disks are where most of the data lives today
    - Similar reasoning/algorithms required though
Storage Hierarchy: Cache

- Cache
  - Super fast; volatile; Typically on chip
  - L1 vs L2 vs L3 caches ???
    - L1 about 64KB or so; L2 about 1MB; L3 8MB (on chip) to 256MB (off chip)
    - Huge L3 caches available now-a-days
  - Becoming more and more important to care about this
    - Cache misses are expensive
  - Similar tradeoffs as were seen between main memory and disks
  - Cache-coherency ??
Storage Hierarchy: Cache

K8 core in the AMD Athlon 64 CPU

source: http://cse1.net/recaps/4-memory.html
Storage Hierarchy

- Main memory
  - 10s or 100s of ns; volatile
  - Pretty cheap and dropping: 1GB << $100
  - Main memory databases feasible now-a-days
- Flash memory (EEPROM)
  - Limited number of write/erase cycles
  - Non-volatile, slower than main memory (especially writes)
  - Examples?
- Question
  - How does what we discuss next change if we use flash memory only?
  - Key issue: Random access as cheap as sequential access

Storage Hierarchy

- Magnetic Disk (Hard Drive)
  - Non-volatile
  - Sequential access much much faster than random access
  - Discuss in more detail later
- Optical Storage - CDs/DVDs; Jukeboxes
  - Used more as backups… Why?
  - Very slow to write (if possible at all)
- Tape storage
  - Backups; super-cheap; painful to access
  - IBM just released a secure tape drive storage solution
Storage...

- Primary
  - e.g. Main memory, cache; typically volatile, fast
- Secondary
  - e.g. Disks; Solid State Drives (SSD); non-volatile
- Tertiary
  - e.g. Tapes; Non-volatile, super cheap, slow

Storage Hierarchy

<table>
<thead>
<tr>
<th>Storage type</th>
<th>Access time</th>
<th>Relative access time</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache</td>
<td>0.5 ns</td>
<td>Blink of an eye</td>
</tr>
<tr>
<td>L2 cache</td>
<td>7 ns</td>
<td>4 seconds</td>
</tr>
<tr>
<td>1MB from RAM</td>
<td>0.25 ms</td>
<td>5 days</td>
</tr>
<tr>
<td>1MB from SSD</td>
<td>1 ms</td>
<td>23 days</td>
</tr>
<tr>
<td>HDD seek</td>
<td>10 ms</td>
<td>231 days</td>
</tr>
<tr>
<td>1MB from HDD</td>
<td>20 ms</td>
<td>1.25 years</td>
</tr>
</tbody>
</table>

source: http://cse1.net/recaps/4-memory.html
Outline

- Storage hierarchy
- Disks
- RAID
- File Organization
- Etc....

1956 IBM RAMAC
24” platters
100,000 characters each
5 million characters
1979
SEAGATE
5MB

1998
SEAGATE
47GB

2006
Western Digital
500GB
Weight (max. g): 600g

2000-ish:
Single hard drive:
  Seagate Barracuda 7200.10 SATA
  750 GB
  7200 rpm
  weight: 720g
  Uses “perpendicular recording”

Microdrives

Now:
  • 6 TB internal $145
  • 2 TB internal $205
  • delivered next day
“Typical” Values

Diameter: 1 inch → 15 inches
Cylinders: 100 → 2000
Surfaces: 1 or 2
(Tracks/cyl) 2 (floppies) → 30
Sector Size: 512B → 50K
Capacity → 360 KB to 2TB (as of Feb 2010)
Rotations per minute (rpm) → 5400 to 15000

Accessing Data

- Accessing a sector
  - Time to seek to the track (seek time)
    - average 4 to 10ms
  - Waiting for the sector to get under the head (rotational latency)
    - average 4 to 11ms
  - Time to transfer the data (transfer time)
    - very low
  - About 10ms per access
    - So if randomly accessed blocks, can only do 100 block transfers
      100 x 512bytes = 50 KB/s
- Data transfer rates
  - Rate at which data can be transferred (w/o any seeks)
    30-50MB/s to up to 200MB/s (Compare to above)
  - Seeks are bad!
Seagate Barracuda: 1TB

- Heads 8, Disks 4
- Bytes per sector: 512 bytes
- Default cylinders: 16,383
- Defaults sectors per track: 63
- Defaults read/write heads: 16
- Spindle speed: 7200 rpm
- Internal data transfer rate: 1287 Mbits/sec max
- Average latency: 4.16msec
- Track-to-track seek time: 1msec-1.2msec
- Average seek: 8.5-9.5msec
- We also care a lot about power now-a-days
  - Why?

Reliability

- Mean time to/between failure (MTTF/MTBF):
  - 57 to 136 years
- Consider:
  - 1000 new disks
  - 1,200,000 hours of MTTF each
  - On average, one will fail 1200 hours = 50 days!
Disk Controller

- Interface between the disk and the CPU
- Accepts the commands
- *checksums* to verify correctness
- Remaps bad sectors

![Diagram of system bus, disk controller, and disks]

Optimizing block accesses

- Typically sectors too small
- Block: A contiguous sequence of sectors
  - 4k to 16k
  - All data transfers done in units of blocks
- Scheduling of block access requests?
  - Considerations: *performance* and *fairness*
  - *Elevator algorithm*
Solid State Drives

- Essentially flash that emulates hard disk interfaces
- No seeks → Much better random reads performance
- Writes are slower, the number of writes at the same location limited
  - Must write an entire block at a time
- About a factor of 10 more expensive right now

- Will soon lead to perhaps the most radical hardware configuration change in a while

Outline

- Storage hierarchy
- Disks
- RAID
- File Organization
- Etc....
RAID

- Redundant array of independent disks
- Goal:
  - Disks are very cheap
  - Failures are very costly
  - Use “extra” disks to ensure reliability
    - If one disk goes down, the data still survives
  - Also allows faster access to data
- Many raid “levels”
  - Different reliability and performance properties

Redundant Array Independent Disks

RAID 0 – Blocks Striped. No Mirror. No Parity.

Fast!

RAID 1 – Blocks Mirrored. No Stripe. No parity.

Redundant!

RAID 2 – Blocks Striped. ( and stores ECC)

Weird!

RAID 5 – Blocks Striped. Distributed Parity.

thegeekstuff.com
RAID Level 5

- Distributed parity “blocks” instead of bits
- Normal operation:
  - “Read” directly from single disk.
  - Load distributed across all 5 disks
  - “Write”: Need to read and update the parity block
    - To update 9 to 9’
      - read 9 and P2
      - compute $P2' = P2 \oplus 9 \oplus 9'$
      - write 9’ and P2’