The Lecture Contains:

Prev topic Next topic Prev page
Next page

Memory organisation

- Faster storage is more costly
- Gap between CPU and main memory speed is increasing
- Good programs exploit locality of data: both spatially and temporally

Memory organisation

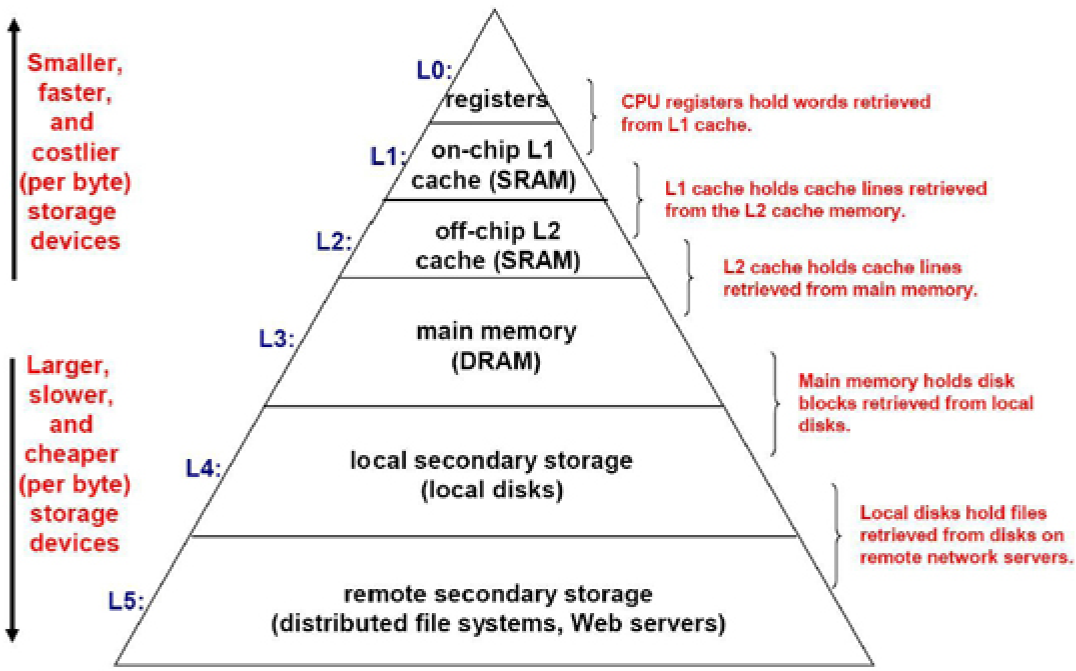
- Faster storage is more costly
- Gap between CPU and main memory speed is increasing
- Good programs exploit locality of data: both spatially and temporally
- Hence, memory and storage systems should be organized as a memory hierarchy

Module 1: Basics and Background

Lecture 4: Memory and Disk Accesses

Prev topic Next topic Prev page
Next page

Example of memory hierarchy



Prev topic Next topic Prev page
Next page

Memory hierarchy

- Smaller capacity but faster access memory acts as a subset of storage for larger capacity but slower access memory
- Generalization of the concept of "cache"
- Why does memory hierarchy work?

Module 1: Basics and Background

Lecture 4: Memory and Disk Accesses

Prev topic Next topic Prev page
Next page

Memory hierarchy

- Smaller capacity but faster access memory acts as a subset of storage for larger capacity but slower access memory
- Generalization of the concept of "cache"
- Why does memory hierarchy work?
    - Locality of access: Programs tend to access data higher up in the hierarchy much more often
    - Intelligent caching algorithms
    - Net effect: Large capacity of memory with very fast access

Prev topic Next topic Prev page
Next page

Disks

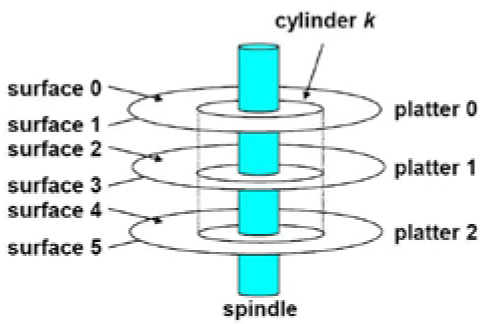
- Physically, disks consist of circular platters
- Both surfaces of a platter can be accessed
- Each surface contains concentric tracks
- Tracks are divided into sectors separated by gaps
- Aligned tracks form a cylinder

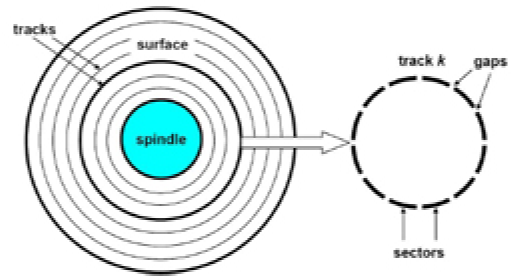
Module 1: Basics and Background

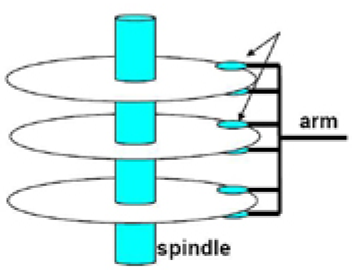
Lecture 4: Memory and Disk Accesses

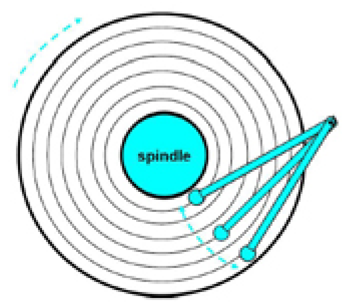
Prev topic Next topic Prev page
Next page

Disk access









Prev topic Next topic Prev page
Next page

Disk capacity

Capacity=(#bytes/sector)x(#sectors/track)x(#tracks/surface)x(#surfaces/platter)x(#platters/disk)

- Example
    - 1024 bytes / sector
    - 400 sectors / track
    - 20000 tracks / surface
    - 2 surfaces / platter
    - 5 platters / disk
    - Capacity $= 1024 \times 400 \times 20000 \times 2 \times 5$ bytes $=$ 80 GB

Disk access time

- Smallest unit of information that can be read from or written to disk is a sector
- Block or page is a logical unit read from or written to by O/S
    - Block consists of a contiguous sequence of sectors

Disk access time

- Smallest unit of information that can be read from or written to disk is a sector
- Block or page is a logical unit read from or written to by O/S
  - Block consists of a contiguous sequence of sectors
- Access time T_{access}: Time to access a particular sector

$$T_{access} = T_{seek} + T_{rotation} + T_{transfer}$$

- Seek time T_{seek} : Time to position arm heads over cylinder containing the target sector
  - Typical seek time: 8 ms
- Rotational latency $T_{rotation}$: (Average) time to rotate r/w head to the first bit of the sector
  - $T_{rotation} =$ (1 / 2) $\times$ (1 / rpm) $\times$ (60 s / 1 min)
- Transfer time $T_{transfer}$: Time to read bits from the sector
  - $T_{transfer} =$ (1 / (#sectors / track)) $\times$ (60 / rpm)

Typical disk parameters

- Average seek times from 4-10 ms
- Rotational speeds are 60, 90, 120, 250 revolutions per second, i.e. ,3600, 5400, 7200, 15000 rpm respectively
- Sector sizes vary between 512 bytes and 1024 bytes
- 400 to 1000 sectors per track
- 20,000 to 50,000 tracks per surface
- 1 to 5 platters per disk

Typical disk parameters

- Average seek times from 4-10 ms
- Rotational speeds are 60, 90, 120, 250 revolutions per second, i.e.,3600, 5400, 7200, 15000 rpm respectively
- Sector sizes vary between 512 bytes and 1024 bytes
- 400 to 1000 sectors per track
- 20,000 to 50,000 tracks per surface
- 1 to 5 platters per disk
- Example
    - Rotational speed = 7200 rpm
    - Average seek time T_{seek} = 8 ms
    - Average #sectors / track = 400
    - $T_{rotation} = (1 / 2) \times (1 / 7200) \times 60 = 4.17$ ms
    - $T_{transfer} = (1 / 400) \times (1 / 7200) \times 60 = 0.02$ ms
    - $\therefore T_{access} = 8 + 4.17 + 0.02 = 12$ ms

Prev topic Next topic Prev page
Next page

Access times

- This disk access time is for random I/O

Module 1: Basics and Background

Lecture 4: Memory and Disk Accesses

Prev topic Next topic Prev page
Next page

Access times

- This disk access time is for random I/O
- Once the first bit is read, the rest (sequential I/O) is almost free (only 0.02 ms)
- Ratio of random I/O to sequential I/O is, therefore, (12 / 0.02) $=$ 600 times
- Bulk transfer rates are calculated more precisely using gaps

Access times

- This disk access time is for random I/O
- Once the first bit is read, the rest (sequential I/O) is almost free (only 0.02 ms)
- Ratio of random I/O to sequential I/O is, therefore, (12 / 0.02) $=$ 600 times
- Bulk transfer rates are calculated more precisely using gaps
- Time to access a byte $=$ 60 ns for DRAM (used for main memory) and 4 ns for SRAM (used for cache memory)
- Ratio of disk access time to memory access time $=$ 40000 for SRAM and 2500 for DRAM

Access times

- This disk access time is for random I/O
- Once the first bit is read, the rest (sequential I/O) is almost free (only 0.02 ms)
- Ratio of random I/O to sequential I/O is, therefore, (12 / 0.02) $=$ 600 times
- Bulk transfer rates are calculated more precisely using gaps
- Time to access a byte $=$ 60 ns for DRAM (used for main memory) and 4 ns for SRAM (used for cache memory)
- Ratio of disk access time to memory access time $=$ 40000 for SRAM and 2500 for DRAM
- Disk access time is dominated by seek time and rotational latency
- Sequential access algorithms exploit the (almost) free access time of later bits heavily
- Most algorithms aim to avoid random I/Os