Data Storage and Query Answering

Data Storage and Disk Structure (1)

Review of DBS & DBMS

- A Database Management System (DBMS) is a software package designed to store, manage and retrieve databases.
- A Database System (DBS) consists of two components:
  - The DBMS;
  - The database.

Review of DBS & DBMS (cont.)

- Why use a DBS?
  - Logical data independence.
  - Physical data independence.
  - Efficient access.
  - Reduced application development time.
  - Data integrity and security.
  - Concurrent access / concurrency control.
  - Recovery from crashes.

A Simple Implementation of DBMS

- One file per table
  - Students(name, id, dept) in a file Students
  - A meta symbol "#" to separate attributes

    Smith#123#CS

    Johnson#522#EE

    …
- Database schema in a special file Schema

    Students#name#STR#id#INT#dept#STR

    Depts#name#STR#office#Str

    …

Naïve Query Answering

SELECT * FROM Students WHERE dept = 'CS' | CSStud

- Read file Schema to determine the attributes of relation Student and their types
- Check that condition dept = 'CS' is semantically valid for Students
- Create a new file CSStud
- Read file Students, for each line
  - Check condition dept = 'CS', if it is true then write the line as a tuple to file CSStud
- Add to the file Schema a line about CSStud
- Problems
  - If we change EE to ECON in one tuple in Students, the entire file has to be rewritten
  - Even if we look for one student, we have to read the whole file
  - If multiple users read/write file Students simultaneously, what would happen?

Handling Joins

SELECT office FROM Students, Depts
WHERE Students.name = ‘Smith’ AND Students.dept = Depts.name;

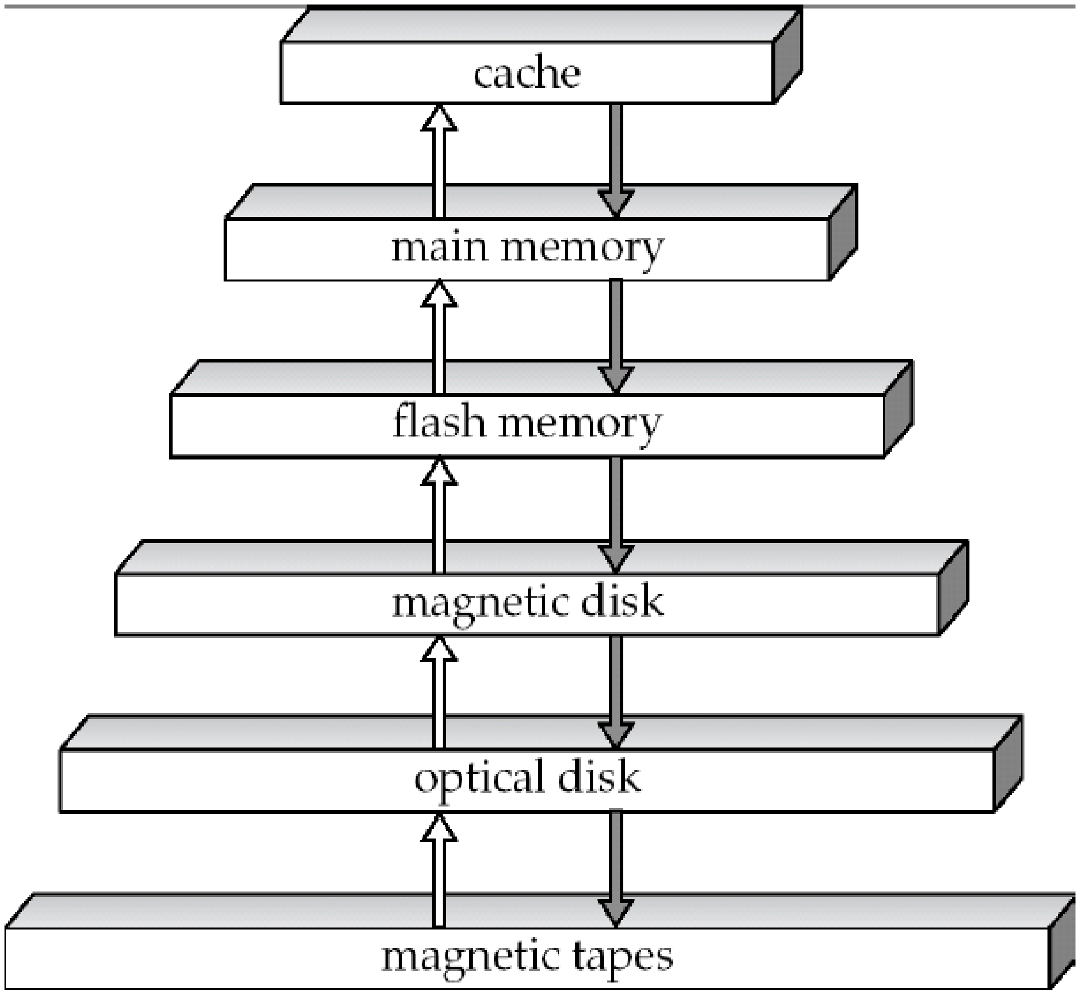
- Algorithm

```
FOR each tuple s in Students DO
  FOR each tuple d in depts DO
    IF s.name = ‘Smith’ AND s.dept = d.dept THEN
         write d.office as a tuple to the output
```

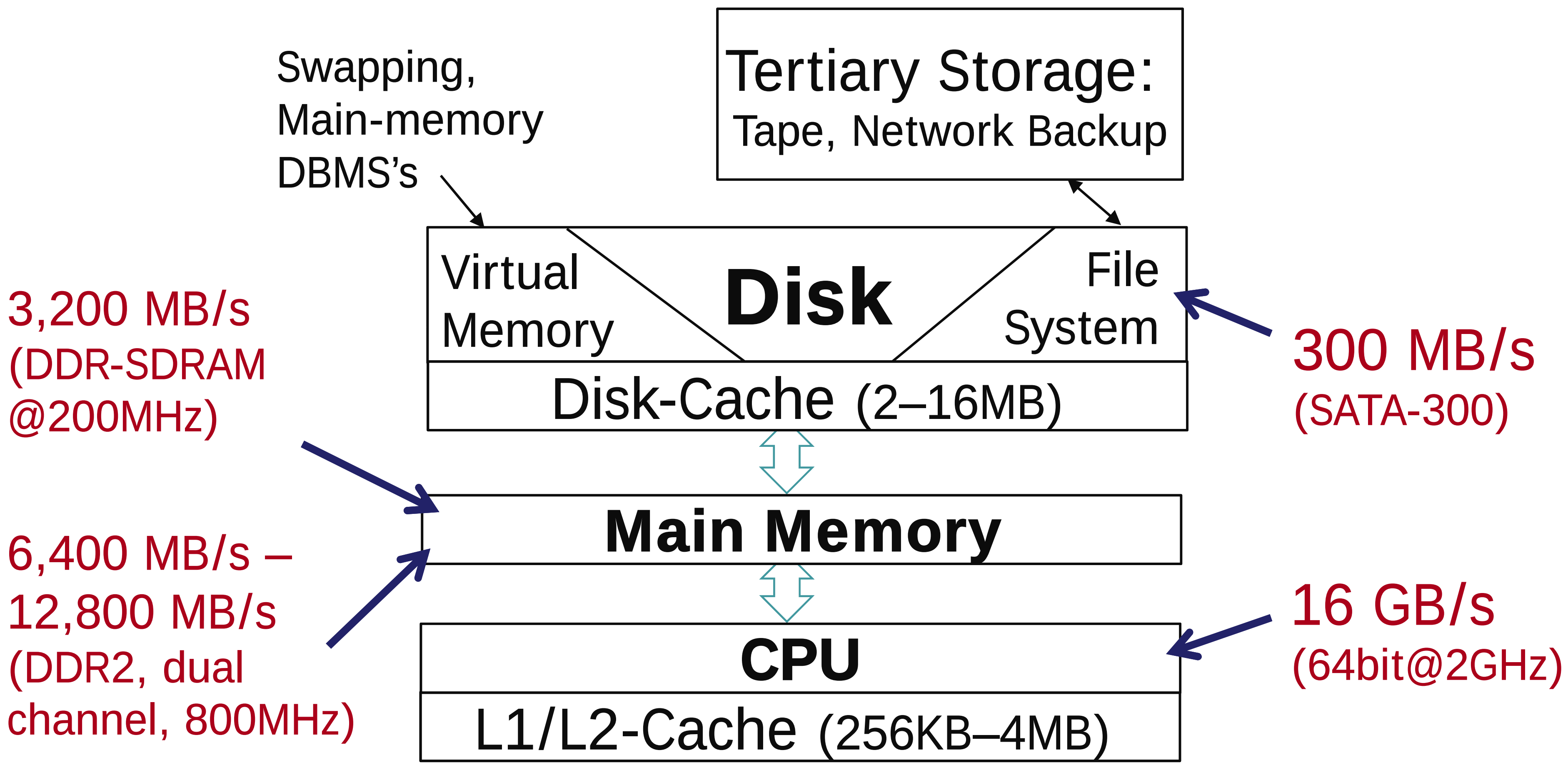
- More problems
  - Why do we need to match a student “Cindy” with all departments?
  - I/O Complexity: $O(n^2)$, costly!
  - What if the system crashes?

Storage Device Hierarchy

- How should we store data on disks so that queries can be answered efficiently?
- How can we organize disks effectively so that a database built on top can be more efficient and robust?



The Memory Hierarchy



<u>CPU-to-Main-Memory:</u>
~200 cycles latency

<u>CPU-to-L1-Cache:</u>
~5 cycles initial latency,
then "burst" mode

The Memory Hierarchy (cont.)

- Cache
  - Data and instructions in cache when needed by CPU.
  - On-board (L1) cache on same chip as CPU, L2 cache on separate chip.
  - Capacity ~ 1MB, access time a few nanoseconds.
- Main memory
  - All active programs and data need to be in main memory.
  - Capacity ~ 1 GB, access time 10-100 nanoseconds.

The Memory Hierarchy (cont.)

- Secondary storage
  - Secondary storage is used for permanent storage of large amounts of data, typically a magnetic disk.
  - Capacity up to 1 TB, access time ~ 10 milliseconds.
- Tertiary storage
  - To store data collections that do not fit onto secondary storage, e.g. magnetic tapes or optical disks.
  - Capacity ~ 1 PB, access time seconds / minutes.

Volatile / Non-Volatile Device

- Trade-off
  - The larger the capacity of a storage device, the slower the access (and vice versa).
- A volatile storage device forgets its contents when power is switched off, a non-volatile device remembers its content.
- Secondary storage and tertiary storage is non-volatile, all others are volatile.
- DBS needs non-volatile (secondary) storage devices to store data permanently.

Memory / Disk

- RAM (main memory) for subset of database used by current transactions.
- Disk to store current version of entire database (secondary storage).
- Tapes for archiving older versions of the database (tertiary storage).

Virtual Memory

- Typically programs are executed in virtual memory of size equal to the address space of the processor.
- Virtual memory is managed by the operating system, which keeps the most relevant part in the main memory and the rest on disk.
- A DBS manages the data itself and does not rely on the virtual memory.
- However, main memory DBS do manage their data through virtual memory.

Moore's Law

- Gordon Moore in 1965 observed that the density of integrated circuits (i.e., number of transistors per unit) increased at an exponential rate, thus roughly doubles every 18 months.
- Parameters that follow Moore's law:
  - Number of instructions per second that can be exceuted for unit cost;
  - Number of main memory bits that can be bought for unit cost;
  - Number of bytes on a disk that can be bought for unit cost.

Moore’s Law (cont.)

- But some other important hardware parameters do not follow Moore’s law and grow much slower.
- Theses are, in particular,
  - Speed of main memory access;
  - Speed of disk access.
- For example, disk latencies (seek times) have almost stagnated for past 5 years.
- Thus, moving data from one level of the memory hierarchy to the next becomes progressively larger.