2.2. Internals

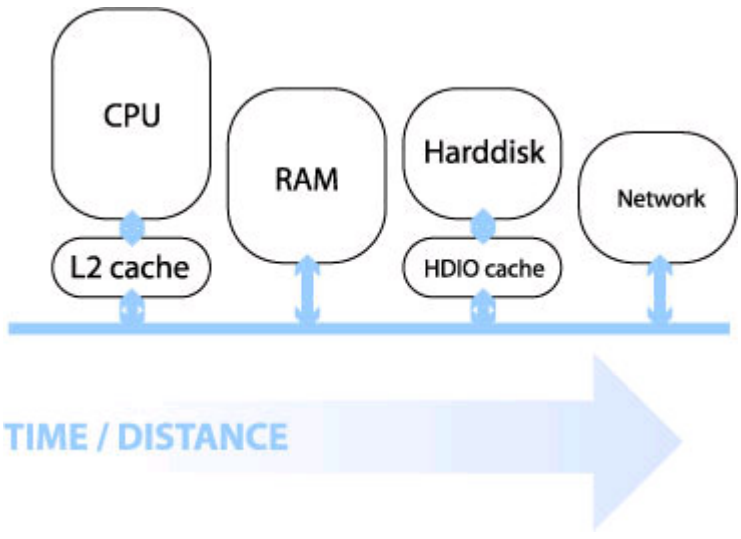
In this lecture we look at...

2.2.01. Introduction

- Database internals (base tier)
- RAID technology
  - Reliability and performance improvement
- Record and field basics
- Headers to hashing
- Index structures

2.2.01b. Machine architecture (by distance)

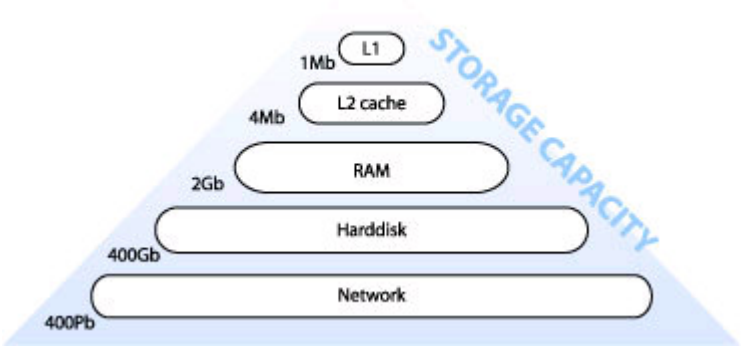
- Distance from chip determines minimum latency
- Speed of light is a constant
- Impact of bus frequencies
  - IDE (66,100,133 Hz)
  - PCI, PCI-X (66,100,133 Hz)
  - PCI Express (1Ghz to 12Ghz)
- Impact of bus bandwidths
  - PCI (32/64 bit/cycle, 133MB/s)
  - PCI Express (x16 8.0GB/s)



Here's a link from Intel showing a machine architecture with signal bandwidths: Intel diagram

2.2.01c. Machine architecture (by capacity)

- Capacity increased with distance
- Staged architecture as compromise
- Speed, time/distance
- Also cost, heat, usage scale



2.2.02. Database internals

- Stored as files of records (of data values)
  - Auxiliary data structures/indices
- 1y and 2y storage
  - memory hierarchy (pyramid diagram)
  - volatility
- Online and offline devices
- Primary file organisation, records on disk
  - Heap - unordered
  - Sorted - ordered, sequential by sort key
  - Hashed - ordered by hash key
  - B-trees - more complex

2.2.03. Disk fundamentals

- DBMS task
  - linked to backup
- 1y, 2y and 3y
  - e.g. DLT tape
- Changing face of current technology
  - Impact of inexpensive harddisks
  - Flash memory devices (CF, USB)
- Random versus sequential access
- Latency (rotational delay) and
- Bandwidth (data transfer rate)

2.2.04. RAID technology

- Redundant Array of Independent Disks

- Data striping
  - Blocks (512 bytes), bits and transparency
- Reliability (1/n)
  - Mirroring/shadowing
  - Error correction codes/parity
- Performance (n)
  - Mirroring (2x read access)
  - Multiple parallel access

2.2.05. RAID levels

- 0 No redundant data
- 1 Disk mirrors (performance gain)
- 2 Hamming codes (also detect)
- 3 Single parity disk
- 4 Block level striping
- 5 and parity/data distribution
- 6 Reed-Soloman codes

2.2.06. Records and fields

- DBMS specific, generally
- Records (tuples) comprise fields (attributes)
- File is a sequence of records
- Variable length records
  - Variable length fields
  - Multi-valued attributes/repeating fields
  - Optional fields
  - Mixed file of different record types

2.2.07. Fields

- records -> files -> disks
- Fixed length for efficient access
- Networking issues
- Delimit variable length fields (max)
- Explicit record/field lengths
- Separators (,;,:,$,?,%)
- Record headers and footers
- Spanning
  - block boundaries and redundancy

2.2.08. Primary organisation

- Bias data manipulation to 1y memory
  - Load record to 1y, write back
  - Cache theorem
- Data storage investment, rapidity of access
  - optimisations based on frequent algorithmic use
- Ordering, ordering field/key field
- Hashing

2.2.09. Indexes/indices

- Auxiliary structures/secondary access path
- Single level indexes (Key, Pointer)
- File of records
- Ordering key field
- Primary, Secondard and Clustering

2.2.09b. Primary index example

- Primary index on simple table
- Ordering key field (primary key) is Integer
- Pointers as addresses
- Sparse, not dense

Primary Index (sparse)				*File*
OKF	**Address**		**Address**	**Surname**
Daniels	0x1F00		0x1F00	Daniels
McBane	0x2200		0x2000	Jameson
Simpson	0x2500		0x2100	Jones
			0x2200	McBane
			0x2300	O'Rourke
			0x2400	Riddesh
			0x2500	Simpson
			0x2600	Smith

2.2.10. Primary Index file (as pairs list)

- Two fields <K(i),P(i)>
- Ordering key field and pointer to block
- Second example, indexing candidate key *Surname*
  - K(1)="Barnes",P(1) -> block 1
    - Barnes record is first/anchor entry in block 1
  - K(2)="Smith",P(2) -> block 6
  - K(3)="Zeta",P(3) -> block 8
- Dense (K(i) for every record), or Sparse
- Enforce key constraint

2.2.10b. Clustering index example

- Clustering index
- Ordering key field (OKF) is non-key
- Each entry points to multiple records

Clustering Index				*File*	
OKF	**Address**		**Address**	**Surname**	**Firstname**
Sanderson	0x1F00		0x1F00	Sanderson	James
Smith	0x2000		0x2000	Smith	Abdul
Sopley	0x2300		0x2100	Smith	Jill
Stanhope	0x2400		0x2200	Smith	Rajir
Szechl	0x2500		0x2300	Sopley	John
			0x2400	Stanhope	Alex
			0x2500	Szechl	Ikir
			0x2600	Szechl	Ptolemy

2.2.11. Clustering Index (as pairs list)

- Two fields <K(i),P(i)>
- Ordering non-key field and pointer to block
  - Internal structure e.g. linked list of records
- Each block may contain multiple records
  - K(1)="Barnes",P(1) -> block 1
  - K(2)="Bates",P(2) -> block 2
  - K(3)="Zeta",P(3) -> block 3
- K(i) not required to have
  - a distinct value for each record
  - non-dense, sparse

2.2.11b. Secondary Index example

- Independent of primary ordering
- Can't use block anchors
- Needs to be dense

Secondary Index			*File*	
OKF	**Address**	**Address**	**Surname**	**Firstname**
Abdul	0x2000	0x1F00	Sanderson	James
Alex	0x2400	0x2000	Smith	Abdul
Ikir	0x2500	0x2100	Smith	Jill
James	0x1F00	0x2200	Smith	Rajir
Jill	0x2100	0x2300	Sopley	John
John	0x2300	0x2400	Stanhope	Alex
Ptolemy	0x2600	0x2500	Szechl	Ikir
Rajir	0x2200	0x2600	Szechl	Ptolemy

2.2.12. More indices

- Single level index
  - ordered index file
  - limited by binary search
- Multi level indices
  - based on tree data structures (B+/B-trees)
    - faster reduction of search space ($\log_{fo}b_i$)

2.2.13. Indices

- Database architecture
  - Intension/extension
- Indexes separated from data file
  - Created/disgraded dynamically
  - Typically 2y to avoid reordering records on disk

2.2.14. Query optimisation

- Faster query resolution
  - improved performance
  - lower load
  - hardware cost:performance ratio
- Moore's law
- Query process chain
- Query optimisation

2.2.15. Query processing

- Compile-track familiarity
  - Scanner/tokeniser - break into tokens
  - Parser - semantic understanding, grammar
  - Validated - check attribute names
    - Query tree
    - Execution strategy, heuristic
  - Query optimisation
    - In (extended relational) canonical algebra form

2.2.16. Query optimisation

- SQL query
  - SELECT lname, fname
  - FROM employee

    - WHERE salary > (
        - SELECT MAX(salary)
        - FROM employee
        - WHERE dno=5
    -);
- Worst-case
    - Process inner for each outer
- Best-base
- Canonical algrebraic form

2.2.16b. Query optimisation implementation

- Indexing accelerates query resolution
- Closed comparison (intra-tuple)
    - all variables/attributes within single tuple
    - e.g. x < 100
- Open comparison (inter-tuple)
    - variables span multiple tuples
- Essentially a sorting problem
- Internal sorting covered (pre-requisites)
- Need external sort for non-cached lists

2.2.17. Query optimisation

- External sorting
    - Stems from large disk (2y), small memory (1y)
    - Sort-merge strategy
        - Sort runs (small sets of total data file)
        - Then merge runs back together
    - Used in
        - SELECT, to accelerate selection (by index)
        - PROJECT, to eliminate duplicates
        - JOIN, UNION and INTERSECTION