Ch 9. Storing Data: Disks and Files
- Heap File Structure -

Sang-Won Lee

http://icc.skku.ac.kr/~swlee

SKKU VLDB Lab.

(http://vldb.skku.ac.kr/)

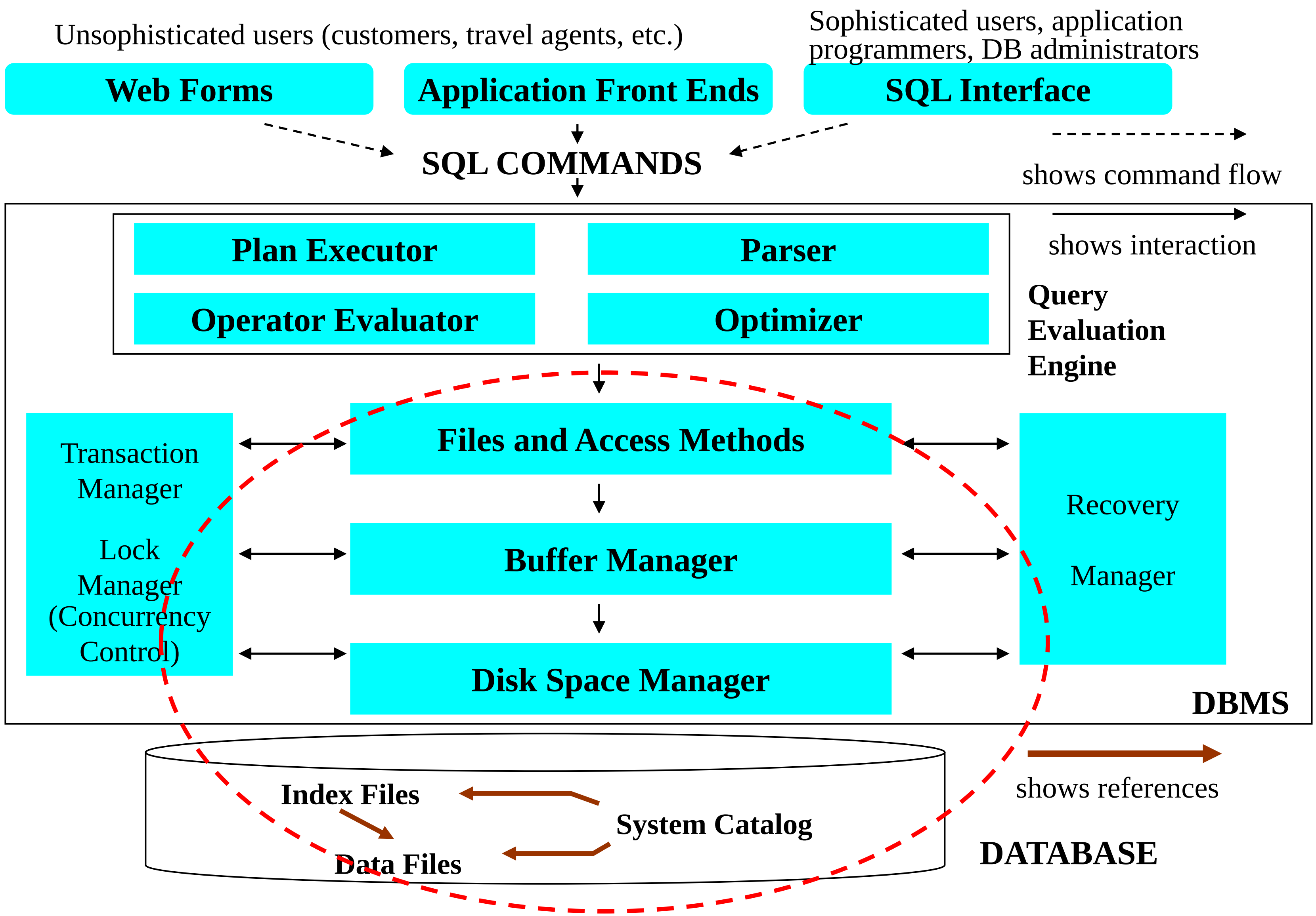
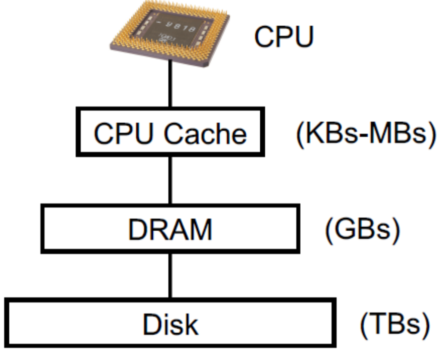


Figure 1.3 Anatomy of an RDBMS

Contents

9.0 Disks and Files



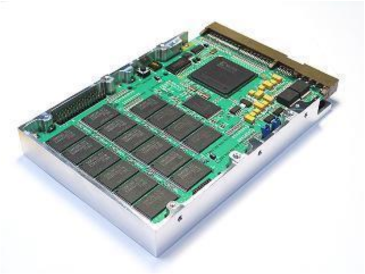
- DBMS stores information on harddisks or flash SSDs.
  - Electronic (CPU, DRAM) vs. Mechanical (harddisk) vs. Electronic (SSD)
- This has major implications for DBMS design!
  - READ: transfer data from disk to main memory (RAM).
  - WRITE: transfer data from RAM to disk.
  - Both are expensive operations, relative to in-memory operations, so must be planned carefully!
    - ✓ DRAM: ~ 10 ns
    - ✓ Harddisk: ~ 10ms
    - ✓ SSD: **80us** ~ 10ms
- CS (and DBMS) is a discipline about numerous trade-offs.
  - Space vs. time; cost vs. performance

Non-Volatile Secondary Storage: Flash SSD vs. Harddisk

60 years champion

VS

A new challenger!

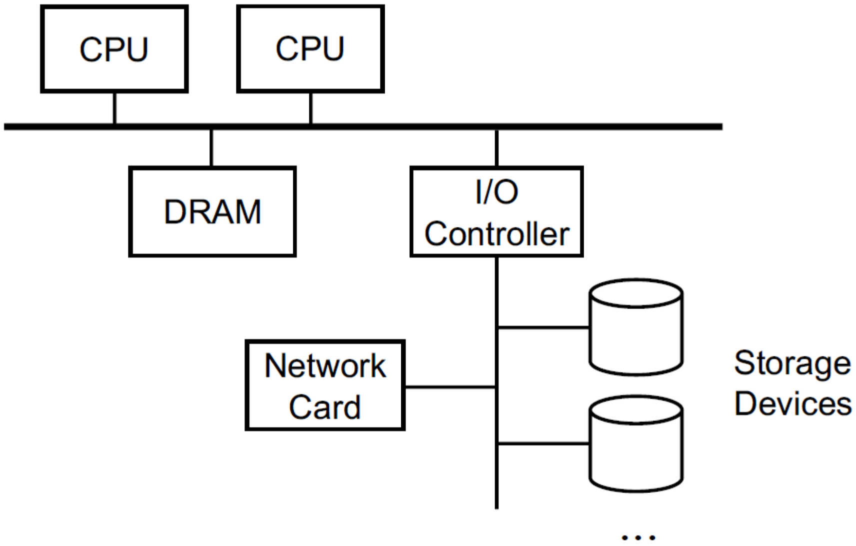


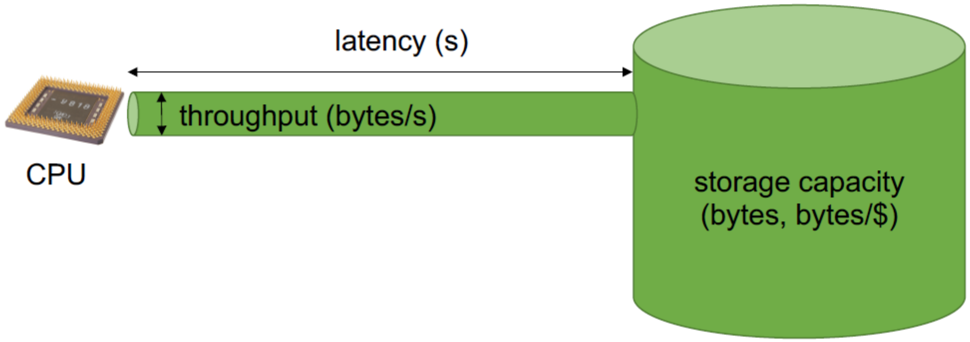
Identical Interface

≡

Flash SSD	HDD
Electronic	Mechanical
Read/Write Asymmetric	Symmetric
No Overwrite	Overwrite

Typical Server and Storage Performance Metrics





Max throughput
large reads (>> 1block):
- DRAM: 100GB/s
- NVMe SSD: 2GB/s
- HDD: 130MB/s

$1,000 @ NewEgg:
- 0.25 TB of DRAM
- 9TB of NVMe SSD
- 50TB of HDD

Source: http://web.stanford.edu/class/cs245/slides/03-System-Architecture-p2.pdf

Storage Performance Metrics

- Capacity ($/GB) : Harddisk >> Flash SSD
- Bandwidth (MB/sec): Harddisk < Flash SSD
- Latency (IOPS): Harddisk << Flash SSD
  - e.g. Harddisk
    - ✓ Commodity hdd (7200rpm): 50$ / 1TB / 100MB/s / 100 IOPS
    - ✓ Enterprise hdd(1.5Krpm): 250$ / 72GB / 200MB/s / 500 IOPS
    - ✓ The price of harddisks is said to be proportional to IOPS, not capacity.

Storage Media	4KB Random Throughput (IOPS)		Sequential Bandwidth (MB/sec)		Capacity in GB	Price in $ ($/GB)
	Read	Write	Read	Write		
MLC SSD†	28,495	6,314	251.33	242.80	256	450 (1.78)
MLC SSD‡	35,601	2,547	258.70	80.81	80	180 (2.25)
SLC SSD§	38,427	5,057	259.2	195.25	32	440 (13.75)
Single disk¶	409	343	156	154	146.8	240 (1.63)
8-disk¶ RAID-0	2,598	2,502	848	843	1,170	1,920 (1.63)

SSD: †Samsung 470 Series 256GB, ‡Intel X25-M G2 80GB, §Intel X25-E 32GB
¶Disk: Seagate Cheetah 15K.6 146.8GB

TABLE I

PRICE AND PERFORMANCE CHARACTERISTICS OF FLASH MEMORY SSDS AND MAGNETIC DISK DRIVES

Storage Device Metrics(2): HDD vs. Flash SSDs

- Other Metrics: Weight/shock resistance/heat & cooling, power(watt) , IOPS/watt, IOPS/$ ….
  - Harddisk << Flash SSD

Table 1. Basic disk and solid-state disk (SSD) performance and cost.

Drive	Cost	Capacity	Power draw while reading data	Read IOPS	Throughput
2.5'' disk	$40	320 Gbytes	2.1 watts/0.75 watt	240	80 Mbytes per second (spec sheet)
Solid-state disk	$220	80 Gbytes	0.1 watt	35,000	250 MBps

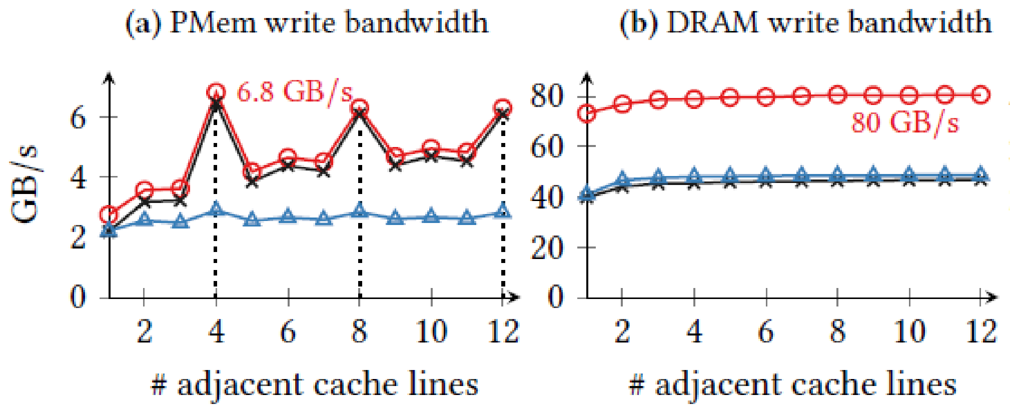
Table 2. Per-dollar and per-watt performance.

Drive	Gbytes per dollar	IOPS per dollar	IOPS per watt	Throughput per dollar	Throughput per watt
2.5'' disk	4	3	104	1 Mbytes/ second	40 MBps
Solid-state disk	0.36	159	220,000	1.13 MBps	2,500 MBps

[Source: Rethinking Flash In the Data Center, IEEE Micro 2010]

Intel Optane DC Persistent Memory vs. Z-SSD

	DRAM	Optane DC PMM	SSD
Read Latency	73 ns	300 ns	230 µs
Seq. Read BW	110 GB/s	36 GB/s	3.5 GB/s
Rand. Read BW	100 GB/s	10 GB/s	1.9 GB/s
Byte-addressable	Yes	Yes	No



Samsung SZ985 Z-NAND SSD	
Form Factor	HHHL
Interface	PCIe Gen3 x4
NAND	Z-NAND Technology
Port	Single
Data Transfer Rate (128KB data size)	
Sequential Read / Write (GB/s)	3.2 / 3.2
Data I/O Speed (4KB data Size, sustained)	
Random Read / Write (IOPs)	750K/ 170K
Latency (sustained random workload)	
Random Read	12 - 20µs
Random Write (Typical)	16µs
DWPD	30
Capacity	800GB

Bandwidth Crisis in AI/ML era?

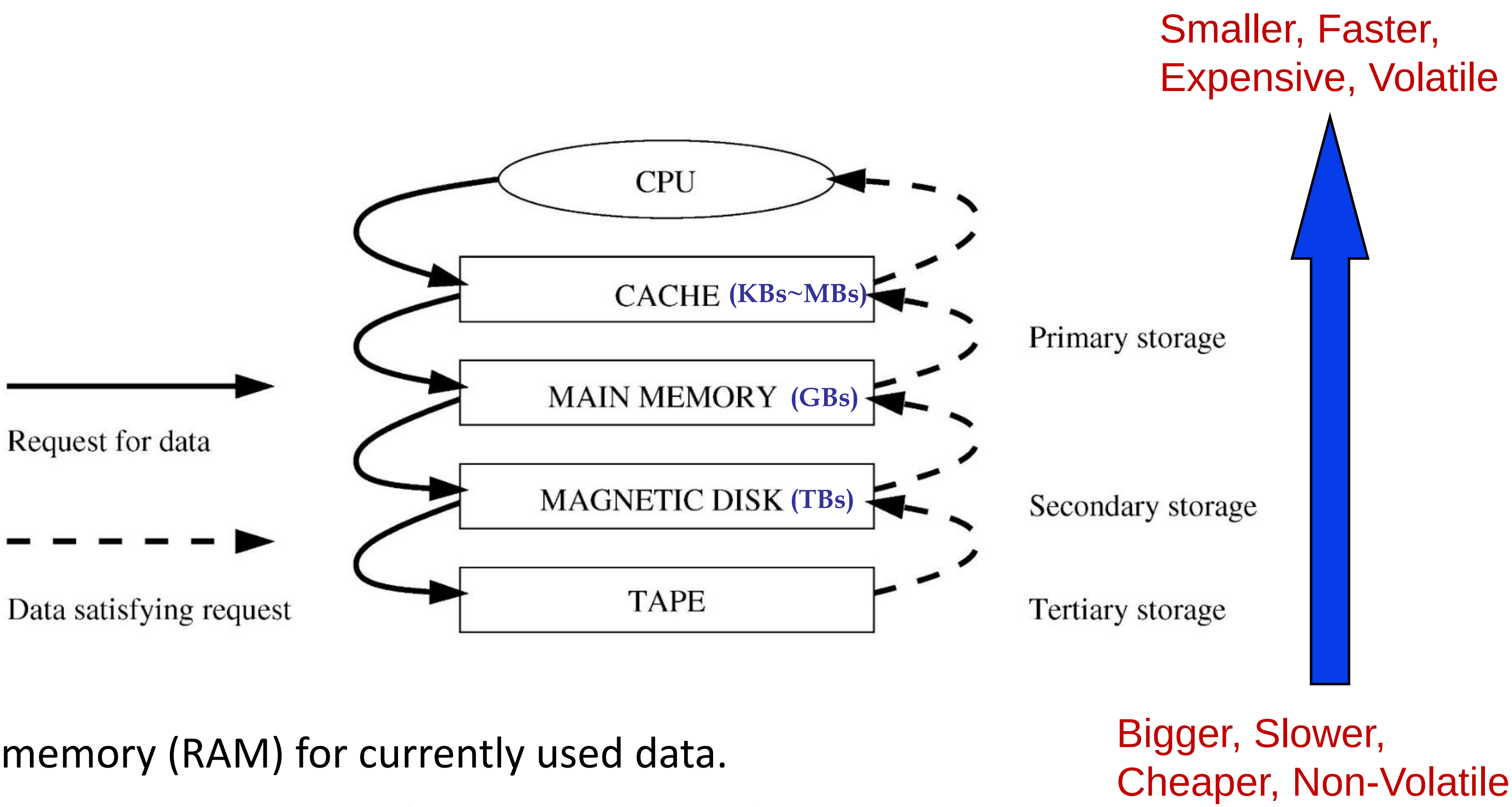
- Data-intensive ML algorithms. (source: Compressed Linear Algebra for Declarative Large-Scale Machine Learning)
  - Many ML algorithms are iterative, with repeated read-only data access. These algorithms often rely on matrix-vector multiplications, which require one complete scan of the matrix with only two floating point operations per matrix element. This low operational intensity renders matrix-vector multiplication, even in-memory, I/O bound.18 **Despite the adoption of flash-and NVM-based SSDs, disk bandwidth is usually 10x-100x slower than memory bandwidth, which is in turn 10x-40x slower than peak floating point performance.** Hence, it is crucial for performance to fit the matrix into available memory without sacrificing operations performance. This challenge applies to single-node in-memory computations, data-parallel frameworks with distributed caching like Spark,20 and accelerators like GPUs with limited device memory. Even in the face of emerging memory and link technologies, the challenge persists due to increasing data sizes, different access costs in the memory hierarchy, and monetary costs.
- One real problem
  - Solution?: 1) cheaper but higher BW memory devices (Intel Optane DC PM), 2) data compression (above link), 3) making ML algorithms more computation intensive (?), 4) offloading ML algorithms near to storage, 5) use multiple low-spec CPU

Storage Wars: File vs. Block vs. Object Storage

	File	Block	Object
Use cases	- File sharing - Local archiving - Data Protection	- DB - Email server - RAID - VM	- Big data - Web apps - Backup archives

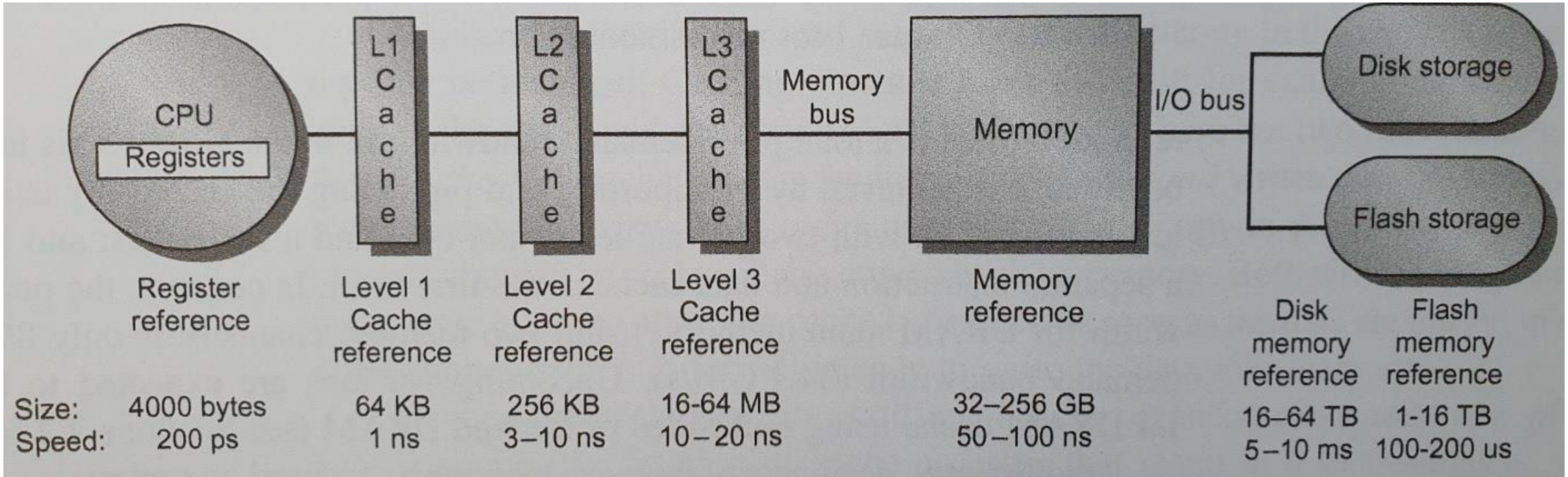
- Object storage is good for scalability for big data (https://blog.storagecraft.com/object-storage-systems/)
  - Scalability is where object-based storage does its most impressive work. Scaling out an object architecture is as simple as adding additional nodes to the storage cluster. Every server has its physical limitations. But thanks to location transparency and remarkable metadata flexibility, this type of storage can be scaled without the capacity limits that plague traditional systems.
- Amazon S3 (Simple Storage Service) and REST API
  - https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html

9.1 Memory Hierarchy



- Main memory (RAM) for currently used data.
- Disk for the main database (secondary storage).
- Tapes for archiving older versions of the data (tertiary storage)
- WHY MEMORY HIERARCHY?
- What if ideal storage appear? Fast, cheap, large, NV..: PCM, MRAM, FeRAM?

Memory Hierarchy for Server



Memory hierarchy for server

Source: Figure 1.2 in "Computer Architecture: A Quantitative Approach (6th Ed.)"

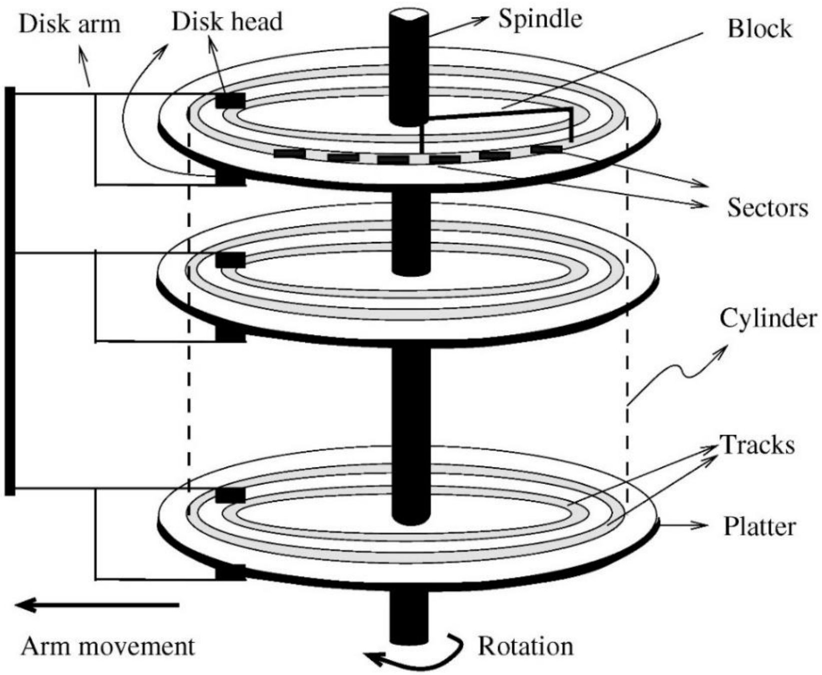
Very Large Data Bases

Disks

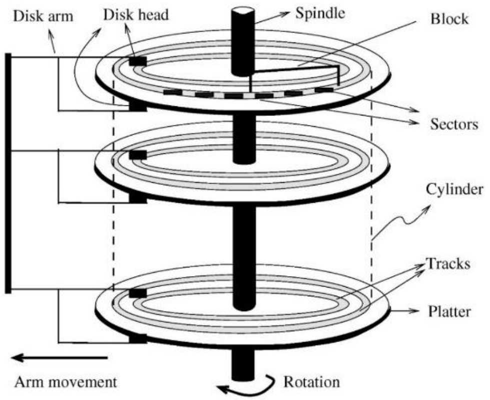
- Secondary storage device of choice. (non-volatile, durable)
- Main advantage over tapes: random access vs. sequential.
  - E.g. To find a record (with its address known) among 1 billion records:
- Data is stored and retrieved in disk blocks or pages unit.
- Unlike RAM, time to retrieve a disk page varies depending upon location on disk.
  - Thus, relative placement of pages on disk has big impact on DB performance!
    - ✓ e.g. adjacent allocation of the pages from the same tables.
  - We need to optimize both data placement and access
    - ✓ e.g. elevator disk scheduling algorithm

Anatomy of a Disk

- The platters spin
  - e.g. 5400 / 7200 / 15K rpm
- The arm assembly is moved in or out to position a head on a desired track. Tracks under heads make a cylinder
  - Mechanical storage -> low IOPS
- Only one head reads/writes at any one time.
  - Parallelism degree: 1
- Block size is a multiple of sector size
- Update-in-place: poisoned apple
- No atomic write
- Fsync for ordering / durability



Accessing a Disk Page



- Time to access (read/write) a disk block:
  1. Seek time (moving arms to position disk head on track)
  2. Rotational delay (waiting for block to rotate under head)
  3. Transfer time (actually moving data to/from disk surface)
- Mechanical devices: seek time and rotational delay dominate.
  - Seek time: about 1 to 20msec
  - Rotational delay: from 0 to 10msec
  - Transfer rate: about 1ms per 4KB page
- Key to lower I/O cost: reduce seek/rotational delays!
  - E.g. disk scheduling algorithm in OS, Linux 4 I/O schedulers

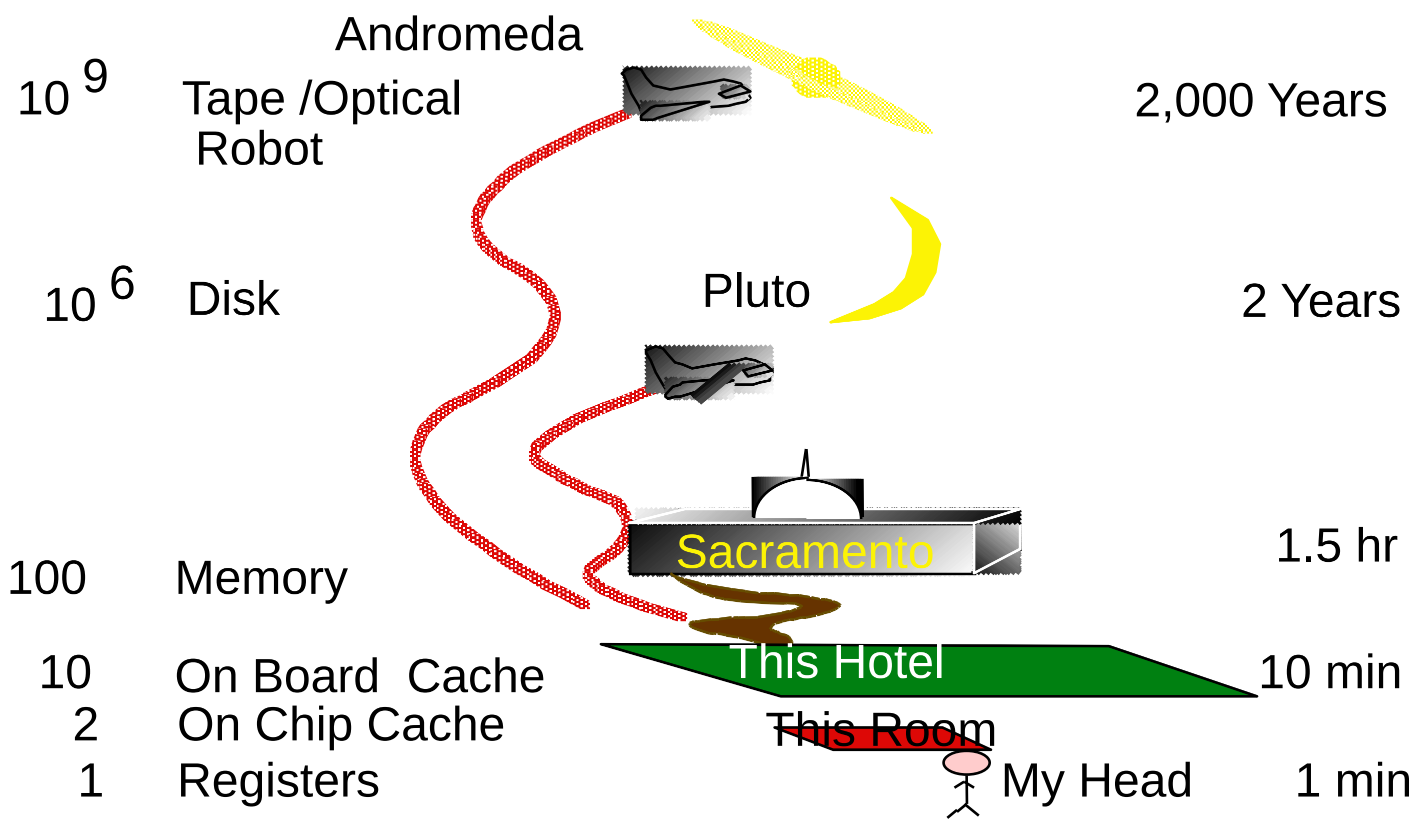
Arranging Pages on Disk

- `Next' block concept:
  - Blocks on same track, followed by
  - Blocks on same cylinder, followed by
  - Blocks on adjacent cylinder
- Blocks in a file should be arranged sequentially on disk (by `next'), to minimize seek and rotational delay.
  - E.g. extent-based allocation (Section 9.5)
- Disk fragmentation problem (see https://en.wikipedia.org/wiki/File_system_fragmentation)
  - Is disk fragmentation still problematic in flash storage?
  - Not a big deal in read, but a big deal in write?

Some Techniques to Hide IO Bottlenecks

- Pre-fetching: For a sequential scan, pre-fetching several pages at a time is a big win! Even cache / disk controller support prefetching
- Caching: modern disk controllers do their own caching.
- IO overlapping: CPU works while IO is performing
  - Double buffering, asynchronous IO
- Multiple threads
- And, don't do IOs, avoid IOs

Jim Gray's Storage Latency Analogy: How Far Away is the Data?



Latency Numbers Every Programmer Should Know

```
Typical one instruction ...................... 01 ns
L1 cache reference ......................... 0.5 ns
Branch mispredict ............................ 5 ns
L2 cache reference ........................... 7 ns
Mutex lock/unlock ........................... 25 ns
Main memory reference ...................... 100 ns
Compress 1K bytes with Zippy ............. 3,000 ns  =   3 µs
Send 2K bytes over 1 Gbps network ....... 20,000 ns  =  20 µs
SSD random read ........................ 150,000 ns  = 150 µs
Read 1 MB sequentially from memory ..... 250,000 ns  = 250 µs
Round trip within same datacenter ...... 500,000 ns  = 0.5 ms
Read 1 MB sequentially from SSD* ..... 1,000,000 ns  =   1 ms
Disk seek ........................... 10,000,000 ns  =  10 ms
Read 1 MB sequentially from disk .... 20,000,000 ns  =  20 ms
Send packet CA->Netherlands->CA .... 150,000,000 ns  = 150 ms
```

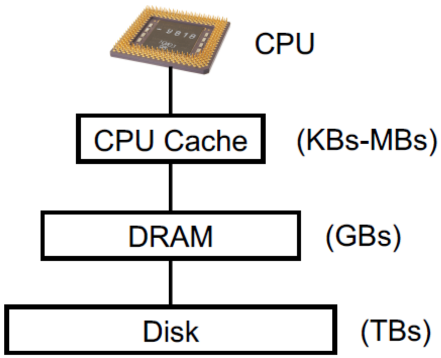
X 10^9

Assuming ~1GB/sec SSD

https://gist.github.com/hellerbarde/2843375
Data by Jeff Dean; Originally by Peter Norvig

Technology RATIOS Matter

[Source: Jim Gray's PPT]



- **Disk improvements ['88 – '04]:**
  - Capacity: 60%/y
  - Bandwidth: 40%/y
  - Access time: 16%/y

- Technology ratio change: 1980s vs. 2020s
  - If everything changes in the same way, then nothing really changes.
  - If some things get much cheaper/faster than others, then that is real change.
    - Some things are not changing much (e.g., cost of people, speed of light) while other things are changing a LOT(e.g., Moore's law, disk capacity)
  - Harddisk: "Latency lags behind bandwidth" and "bandwidth does behind capacity"

- Flash memory/NVRAMs and its role in the memory hierarchy?
  - Disruptive technology ratio change → new disruptive solution?

Evolution of DRAM, HDD, and SSD (1987 – 2017)

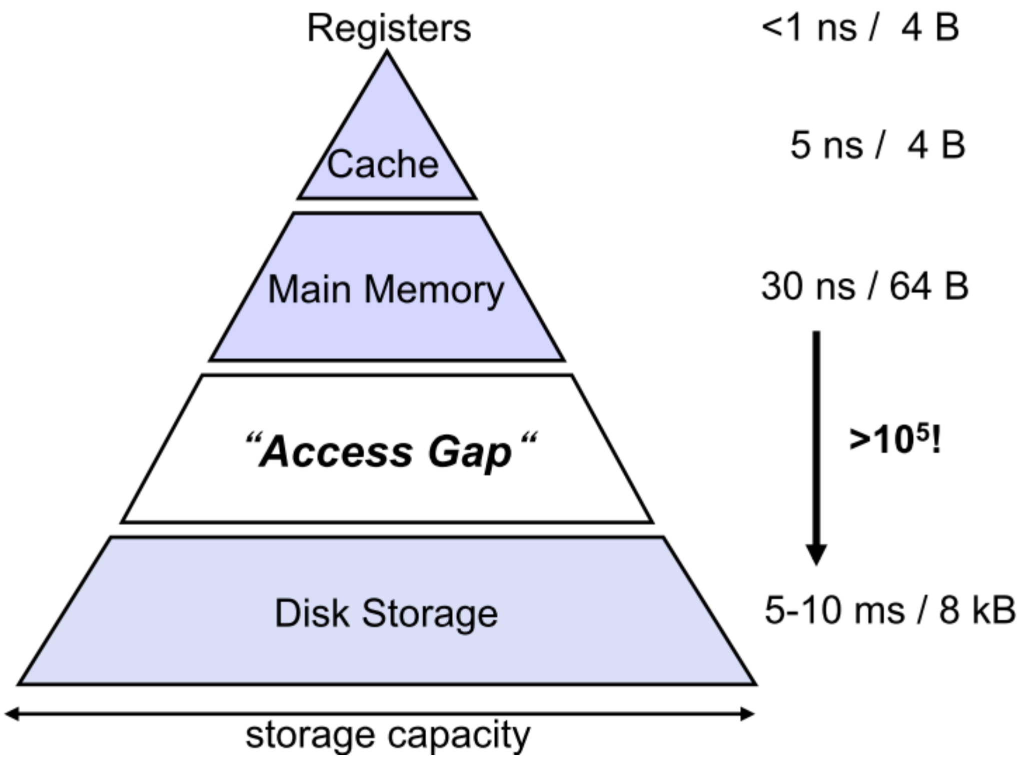
- Raja et. al., The Five-minute Rule Thirty Years Later and its Impact on Storage Hierarchy, ADMS '17
- **the Storage Hierarchy**

Metric	DRAM				HDD				SATA Flash SSD	
	1987	1997	2007	2017	1987	1997	2007	2017	2007	2017
Unit price($)	5k	15k	48	80	30k	2k	80	49	1k	560
Unit capacity	1MB	1GB	1GB	16GB	180MB	9GB	250GB	2TB	32GB	800GB
$/MB	5k	14.6	0.05	0.005	83.33	0.22	0.0003	0.00002	0.03	0.0007
Random IOPS	-	-	-	-	15	64	83	200	6.2k	67k (r)/20k (w)
Sequential b/w (MB/s)	-	-	-	-	1	10	300	200	66	500 (r)/460 (w)

Table 1: The evolution of DRAM, HDD, and Flash SSD properties

Latency Gap in Memory Hierarchy

Typical access latency & granularity



We need `gap filler'!

: Flash memory SSD!

[Source: Uwe Röhm's Slide]

- Latency lags behind bandwidth [David Patterson, CACM Oct. 2004]
  - Bandwidth problem can be cured with money, but latency problem is harder

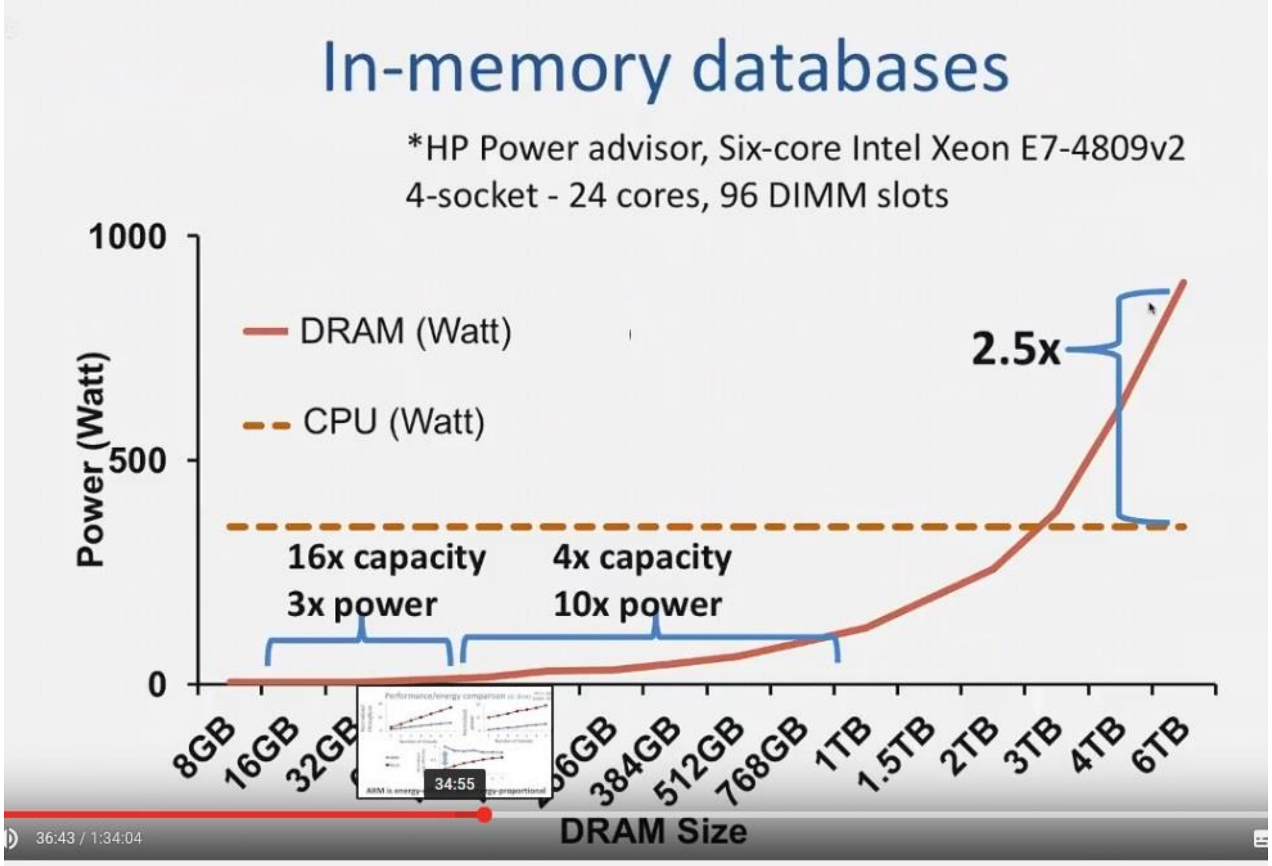
Why Not Store It All in Main Memory?

- Cost!: 20$ /1GB DRAM vs. 50$ / 150 GB of disk (EIDI/ATA) vs. 100$/30GB (SCSI).
  - High-end databases today are in the 10-100 TB range.
  - Approx. 60% of the cost of a production system is in the disks.
- Some specialized systems (e.g. Main Memory(MM) DBMS) store entire database in main memory.
  - Vendors claim 10x speed up vs. traditional DBMS in main memory.
  - Sap Hana, MS Hekaton, Oracle In-memory, Altibase ..
- Main memory is volatile: data should be saved between runs.
  - Disk write is inevitable: 1) log write for recovery and 2) periodic checkpoint

MM-DBMS

- Why MMDBMS has been recently popular since mid-2000s?
  - Sap Hana, MS Hekaton, Oracle In-memory, Altibase, ….
  - The price of DRAM had ever dropped for the last two decades
    - ✓ $/IOPS @ DISK >> $/GB @ DRAM
  - The overhead of disk-based DBMS is not negligible
  - Applications with extreme performance requirements?

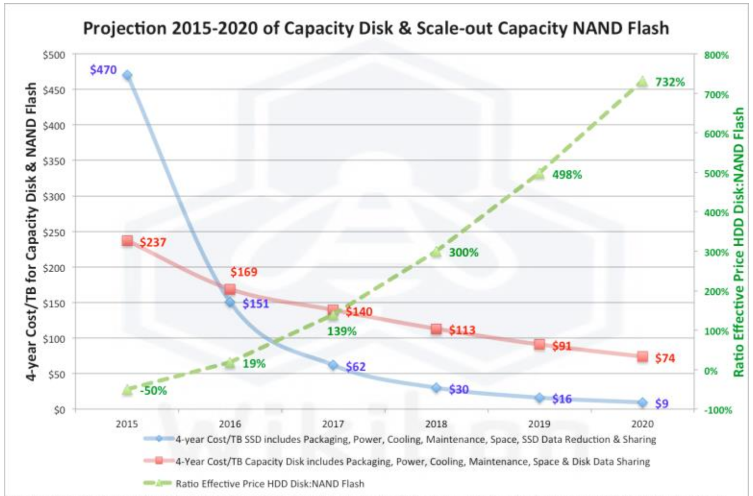
Power Consumption Issue in Big Memory



- Why exponential?
- 1KWh = 15 ~ 50 cents, 1 year = 1,752$

Very Large Data Bases

HDD vs. SSD [Patterson 2016]



Source: © Wikibon 2015. 4-Year Cost/TB Magnetic Disk & SSD, including Packaging, Power, Maintenance, Space, Data Reduction & Data Sharing

Future Memory Hierarchy Deeper

- Storage hierarchy gets more and more complex:
  - L1 cache
  - L2 cache
  - L3 cache
  - Fast DRAM (on interposer with CPU)
  - 3D XPoint based storage
  - SSD
  - (HDD)
- Need to design software to take advantage of this hierarchy

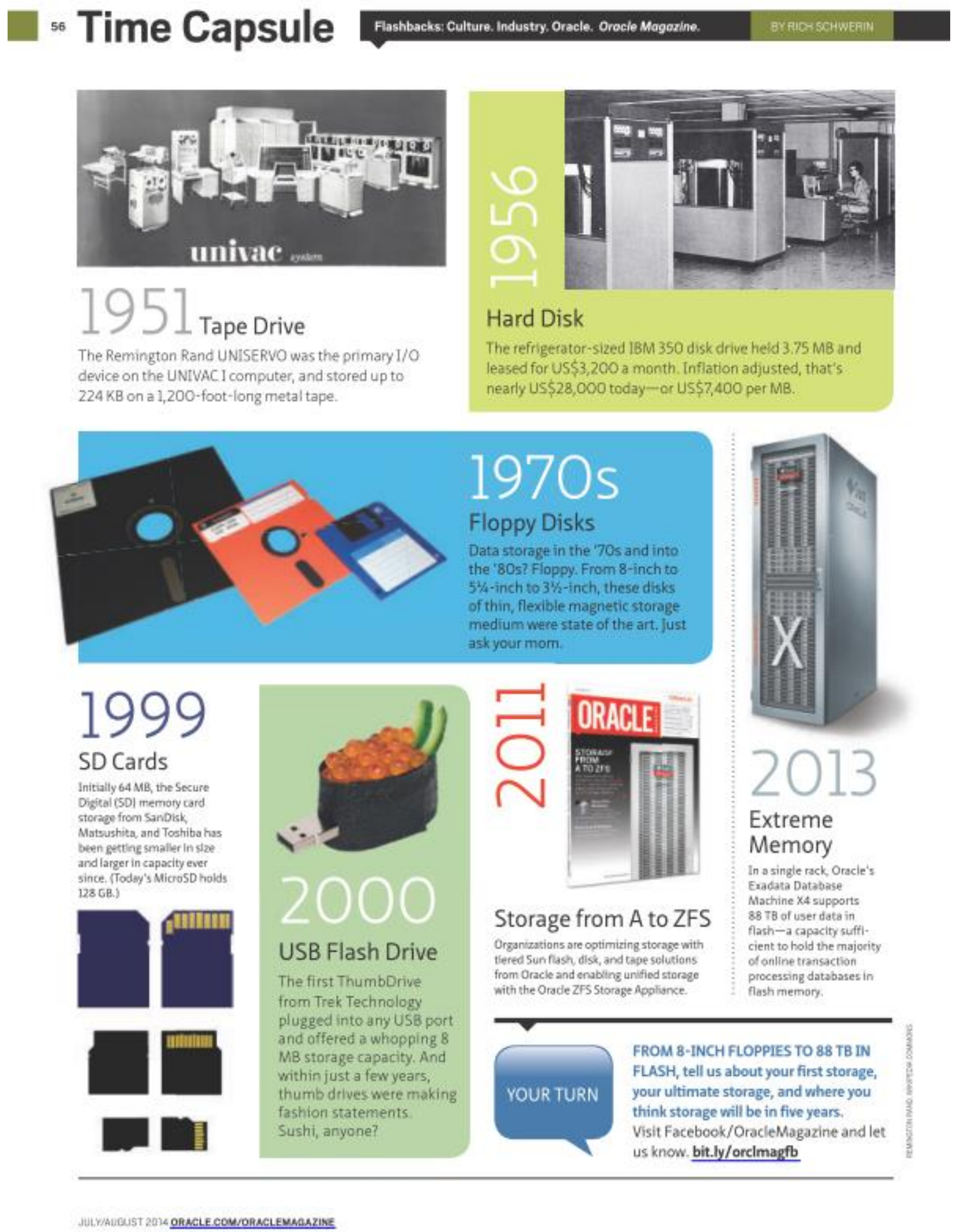
SSDs vs. HDDs

- SSDs will soon become cheaper than HDDs
- Transition from HDDs to SSDs will accelerate
  - Already most instances in Amazon Web Service have SSDs
- Going forward we can assume SSD-only clusters

"Tape is dead, Disk is tape, Flash is disk."
Jim Gray, 2007

Evolution of secondary storages

- Source: Oracle Magazine, July/August 2014
- The Life of a Data Byte (CACM, Dec. 2020)
  - A short history about modern storage medias


56 Time Capsule

Flashbacks: Culture. Industry. Oracle. *Oracle Magazine.* BY RICH SCHWERIN

1951 Tape Drive

The Remington Rand UNISERVO was the primary I/O device on the UNIVAC I computer, and stored up to 224 KB on a 1,200-foot-long metal tape.

1956 Hard Disk

The refrigerator-sized IBM 350 disk drive held 3.75 MB and leased for US$3,200 a month. Inflation adjusted, that's nearly US$28,000 today—or US$7,400 per MB.

1970s Floppy Disks

Data storage in the '70s and into the '80s? Floppy. From 8-inch to 5¼-inch to 3½-inch, these disks of thin, flexible magnetic storage medium were state of the art. Just ask your mom.

1999 SD Cards

Initially 64 MB, the Secure Digital (SD) memory card storage from SanDisk, Matsushita, and Toshiba has been getting smaller in size and larger in capacity ever since. (Today's MicroSD holds 128 GB.)

2000 USB Flash Drive

The first ThumbDrive from Trek Technology plugged into any USB port and offered a whopping 8 MB storage capacity. And within just a few years, thumb drives were making fashion statements. Sushi, anyone?

2011 Storage from A to ZFS

Organizations are optimizing storage with tiered Sun flash, disk, and tape solutions from Oracle and enabling unified storage with the Oracle ZFS Storage Appliance.

2013 Extreme Memory

In a single rack, Oracle's Exadata Database Machine X4 supports 88 TB of user data in flash—a capacity sufficient to hold the majority of online transaction processing databases in flash memory.

YOUR TURN

FROM 8-INCH FLOPPIES TO 88 TB IN FLASH, tell us about your first storage, your ultimate storage, and where you think storage will be in five years. Visit Facebook/OracleMagazine and let us know. bit.ly/orclmagfb

JULY/AUGUST 2014 ORACLE.COM/ORACLEMAGAZINE

Very Large Data Bases

Implications for DBMS Design

- The access characteristics of storage devices (e.g. hard disks and flash SSDs) neccessitate that database systems have the ability to control *where*, *how* and *when* data is physically accessed.

- **Disk Space Management**: 'Spatial control'
  - Where on the secondary storage is the data stored?
- **Buffer Management**: 'Temporal control'
  - When is data physically read from or written to disk?
- **Query Optimization** and **Execution**: 'Access pattern control'
  - How is data accessed? Sequentially or Random Access?

Mega Changes in Computer Architectures and Implications on Database Technology

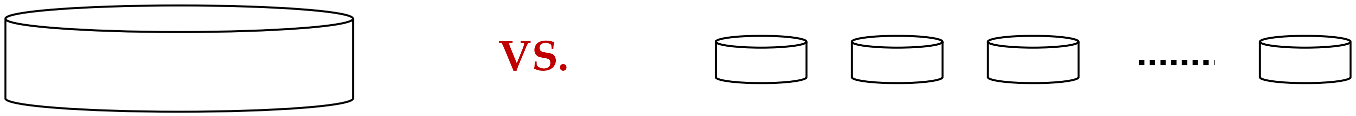
- CPU: Single-core → Multi-core (End of Moore's Law?)
- DRAM: small and expensive → large and become cheaper
- Storage: HDD → Flash SSD (→ NVRAM ?)
- Data center/Cloud: disaggregation, object storage

➔

- MMDBMS
- New concurrency control: 2PL → OCC
- Buffer replacement algorithm
- Cloud-native DBMS (Amazon Aurora, Snowflake)
- And, many others

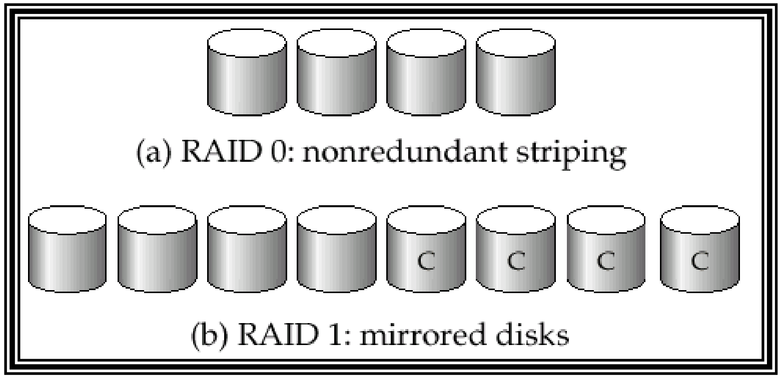
9.2 RAID

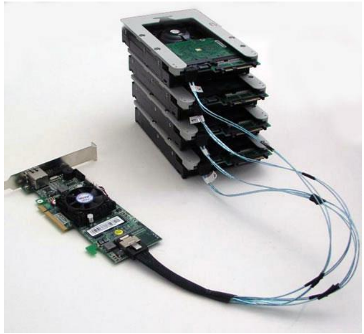
- SLED (Single Large Expensive Disk) approach till 1980s



- Redundant Arrays of Independent(or Inexpensive) Disks
  - Disk array: arrangement of several disks that gives abstraction of a single, large disk.

- Goals: Increase performance and reliability.

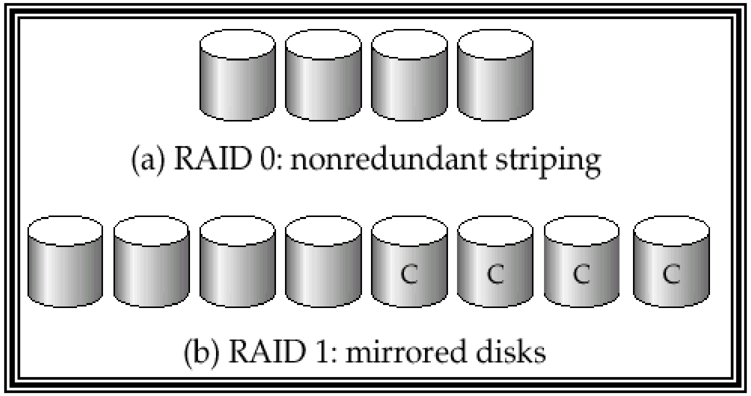




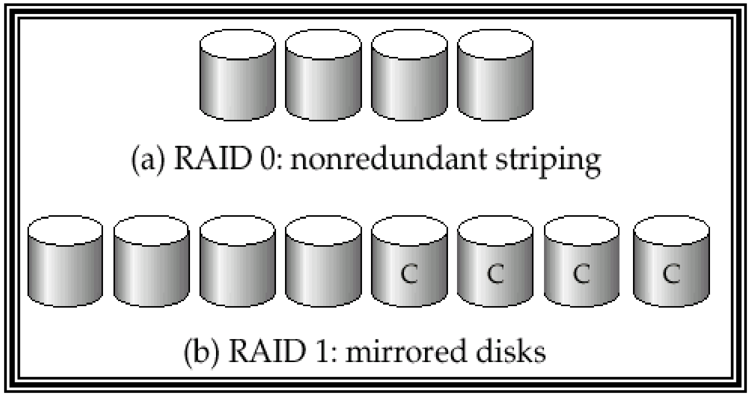
Cf. Tesla Battery and Rocket Tech.

RAID

- Two main techniques:
  - Data striping: Data is partitioned; size of a partition is called the striping unit. Partitions are distributed over several disks.
    - ✓ For large data, larger bandwidth (i.e. transfer rate)
    - ✓ For small random data, higher IOPS
  - Mirroring for redundancy: More disks => more failures. Redundant information allows reconstruction of data if a disk fails.
- Benefits of RAID
  - Bandwidth for sequential IOs
  - IOPS for random IOs
  - Reliability by redundancy
- Another beauty in computer science
  - Simple and powerful!!



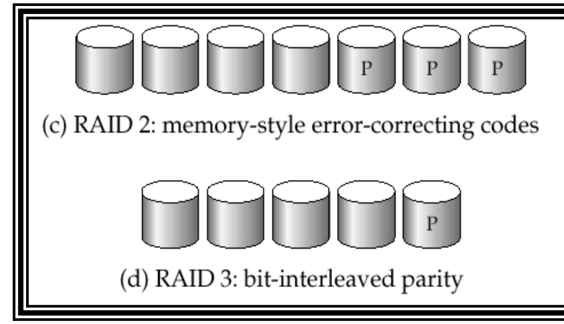
RAID Levels



(a) RAID 0: nonredundant striping
(b) RAID 1: mirrored disks

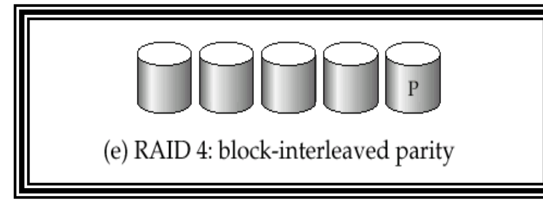
- Level 0: No redundancy
- Level 1: Mirrored (two identical copies)
  - Each disk has a mirror image (check disk)
  - Parallel reads, a write involves two disks.
  - Maximum transfer rate = transfer rate of one disk
- Level 0+1: Striping and Mirroring
  - Parallel reads, a write involves two disks.
  - Maximum transfer rate = aggregate bandwidth

RAID Levels (Contd.)

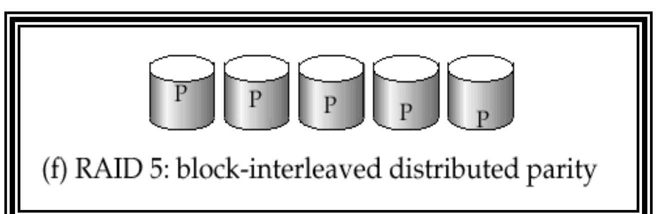

(c) RAID 2: memory-style error-correcting codes

(d) RAID 3: bit-interleaved parity

- Level 3: Bit-Interleaved Parity
  - Striping Unit: One bit. One check disk.
  - Each read and write request involves all disks; disk array can process one request at a time.


(e) RAID 4: block-interleaved parity

- Level 4: Block-Interleaved Parity
  - Striping Unit: One disk block. One check disk.
  - Parallel reads possible for small requests, large requests can utilize full bandwidth
  - Writes involve modified block and check disk
- Level 5: Block-Interleaved Distributed Parity
  - Similar to RAID Level 4, but parity blocks are distributed over all disks


(f) RAID 5: block-interleaved distributed parity

P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

9.3 Disk Space Management

- Lowest layer of DBMS software manages space on disk.
- Higher levels call upon this layer to:
  - allocate/de-allocate a page
  - read/write a page
- Request for a sequence of pages must be satisfied by allocating the pages sequentially on disk
- Higher levels don't need to know how this is done, or how free space is managed.

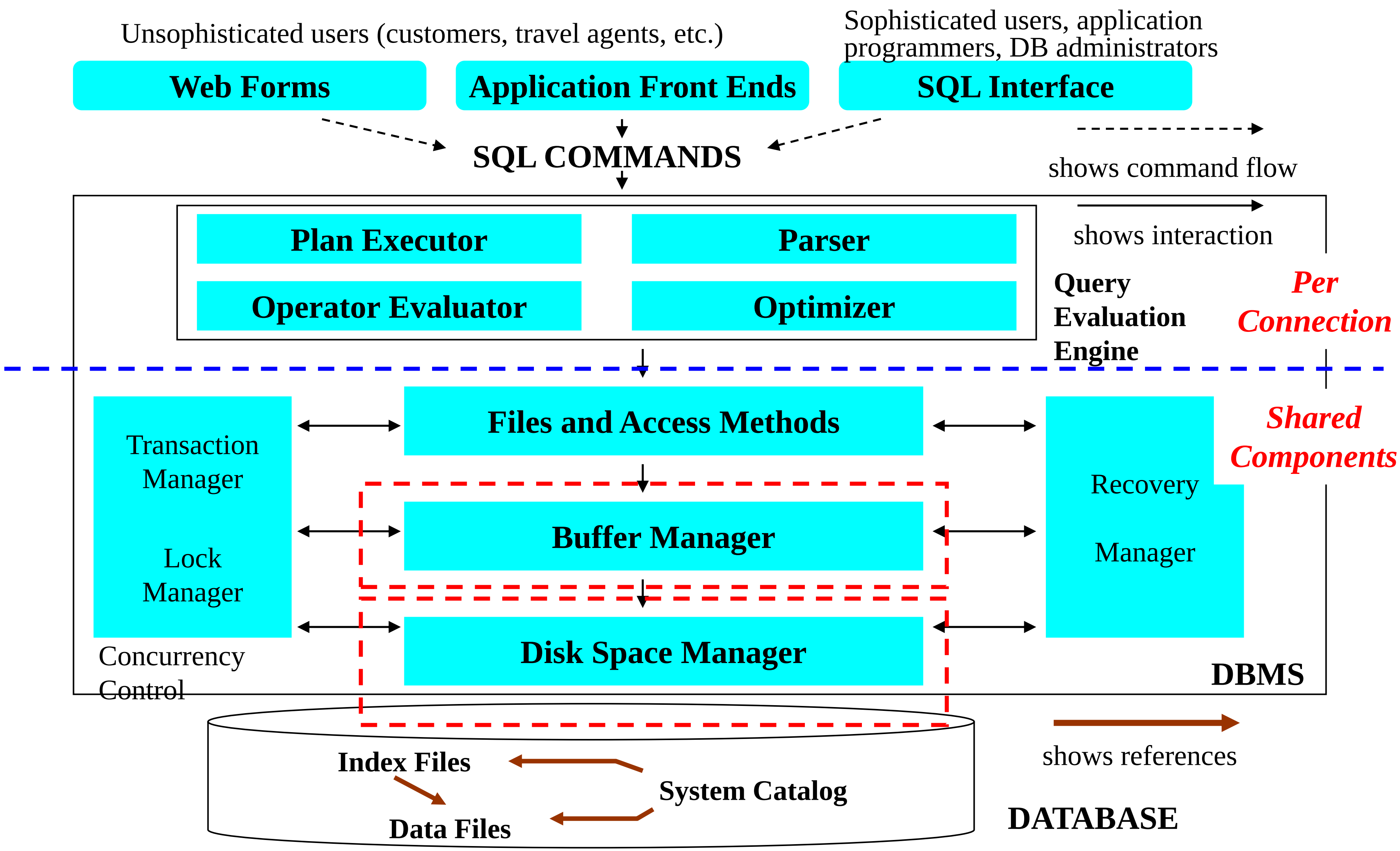
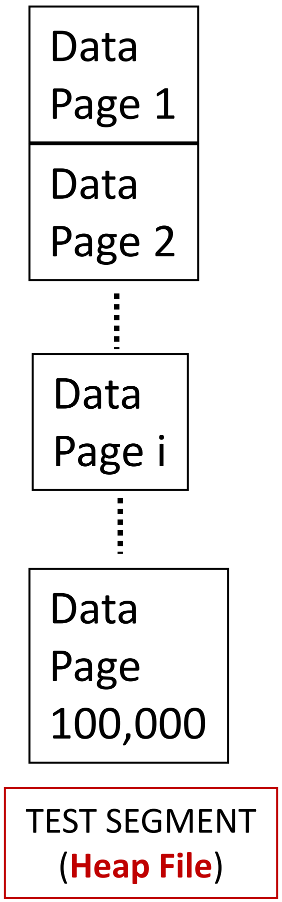


Figure 1.3 Architecture of a DBMS

9.4 Buffer Management in DBMS

Table, Insertions, Heap Files

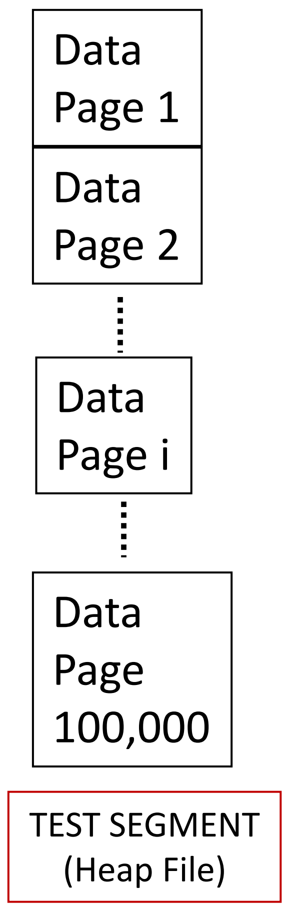
```
CREATE TABLE TEST (a int, b int, c varchar2(650));


/* Insert 1M tuples into TEST table (approximately 664 bytes per
tuple) */
BEGIN
   FOR i IN 1..1000000 LOOP
       INSERT INTO TEST (a, b, c) values (i, i, rpad('X', 650, 'X'));
   END LOOP;
END;

/*
Page = 8KB
10 tuples / page
100,000 pages in total
TEST table = 800MB
*/
```



OLAP vs. OLTP

- On-Line Analytical vs. Transactional Processing

```
SQL> SELECT SUM(b) FROM TEST;


SUM(B)
----------
5.0000E+11
Execution Plan
---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |     5 | 22053   (1)| 00:04:25 |
|   1 |  SORT AGGREGATE    |      |     1 |     5 |            |          |
|   2 |   TABLE ACCESS FULL| TEST |  996K|  4865K| 22053   (1)| 00:04:25 |
---------------------------------------------------------------------------
Statistics
----------------------------------------------------------
      179  recursive calls
        0  db block gets
   100152  consistent gets
   100112  physical reads
…..
        1  rows processed
```



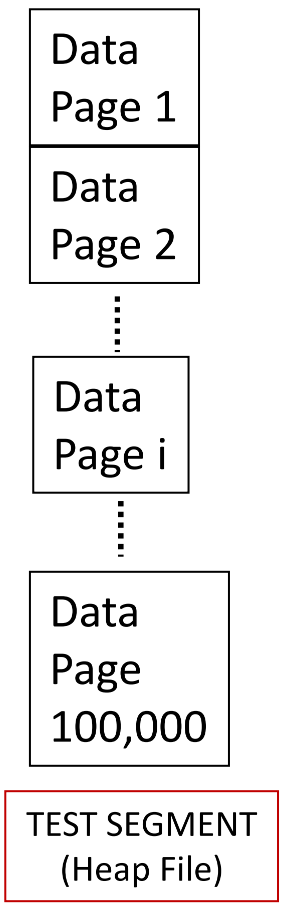
OLAP vs. OLTP

- On-Line Transactional Processing: Point or Range Query

SQL> **SELECT B FROM TEST WHERE A = 500000;**

```
B
----------
 500000
Execution Plan
--------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |     5 | 22053   (1)| 00:04:25 |
|   1 |  SORT AGGREGATE    |      |     1 |     5 |            |          |
|   2 |   TABLE ACCESS FULL| TEST |   996K|  4865K| 22053   (1)| 00:04:25 |
--------------------------------------------------------------------------
Statistics
----------------------------------------------------------
       179  recursive calls
         0  db block gets
    100152  consistent gets
    100112  physical reads
…..
         1  rows processed
```

No Index on A column
Thus, Full Table Scan

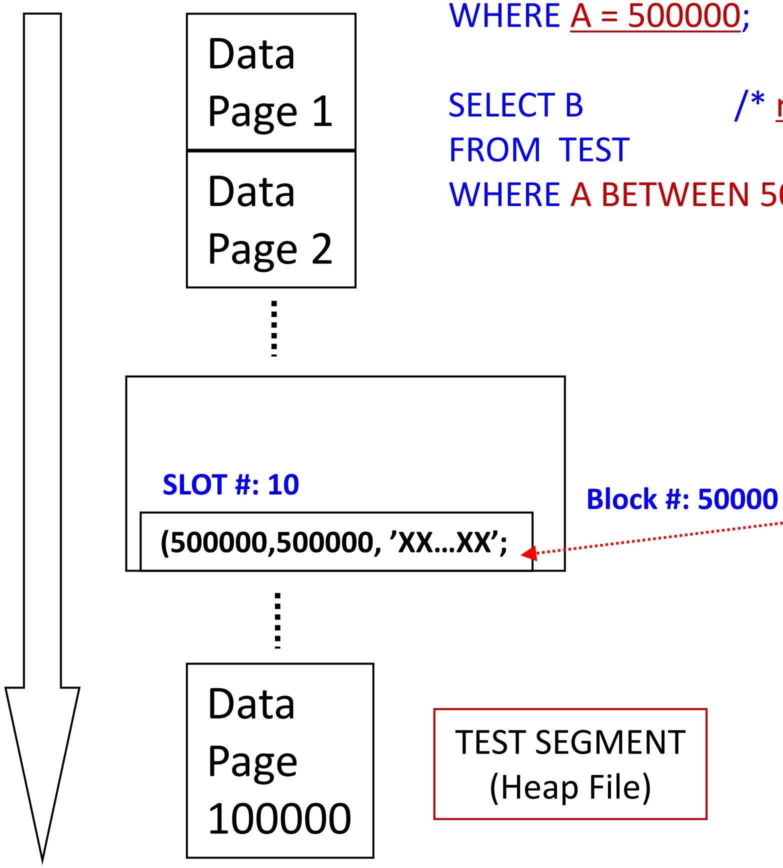


OLAP vs. OLTP

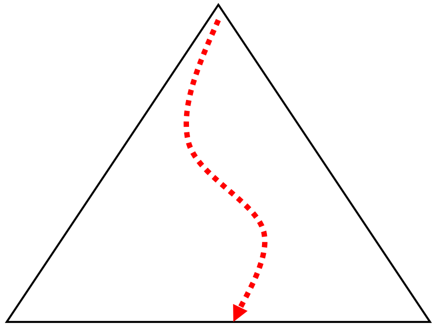
CREATE INDEX TEST_A ON TEST(A);

B-tree Index on TEST(A)

```
SELECT B            /* point query */
FROM  TEST
WHERE A = 500000;

SELECT B            /* range query */
FROM  TEST
WHERE A BETWEEN 50001 and 50101;
```



SEARCH KEY: 500000



(500000, (50000,10))

Cost:
- Full Table Scan: 100,000 Block Accesses
- Index: (3~4) + Data Block Access
  - Point query: 1
  - Range queries: depending on range

OLAP vs. OLTP

- Index-based Table Access

```
SQL> SELECT B FROM TEST WHERE A = 500000;
 B
----------
 500000
Execution Plan
--------------------------------------------------------------------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |        |     1 |    10 |     4   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TEST   |     1 |    10 |     4   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | TEST_A |     1 |       |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------
Statistics
----------------------------------------------------------
…..
          5  consistent gets
          4  physical reads
…..
          1  rows processed
```

Column A is now Indexed, Thus, Index-Scan!

Very Large Data Bases

OLTP: TPC-A/B/C Benchmark

From Gray's Presentation

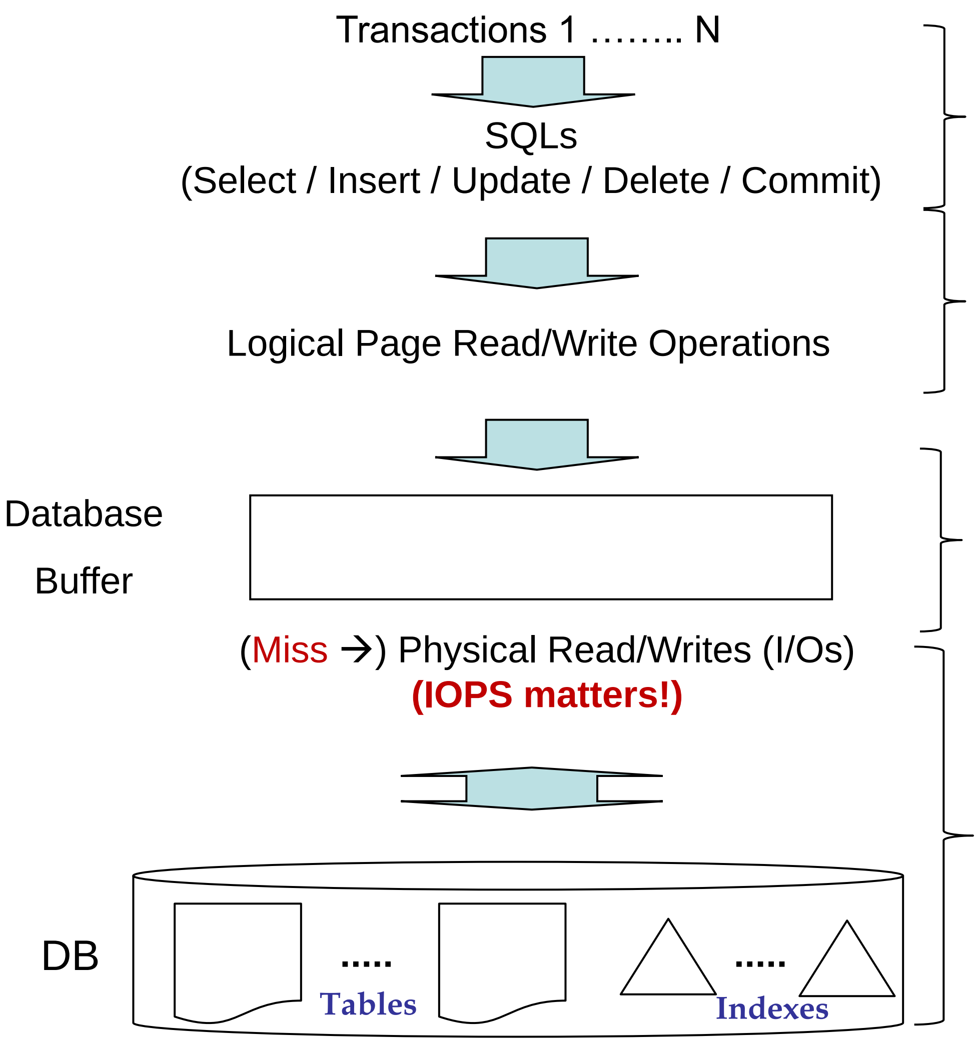
```
ACCOUNT(ACCOUNT_NUMBER, CUSTOMER_NUMBER, ACCOUNT_BALANCE, HISTORY)
CUSTOMER(CUSTOMER_NUMBER, CUSTOMER_NAME, ADDRESS,.....)
HISTORY(TIME, TELLER, CODE, ACCOUNT_NUMBER, CHANGE, PREV_HISTORY)
CASH_DRAWER(TELLER_NUMBER, BALANCE)
BRANCH_BALANCE(BRANCH, BALANCE)
TELLER(TELLER_NUMBER, TELLER_NAME,......)
```

```
exec sql begin declare section;
long Aid, Bid, Tid, delta, Abalance;
exec sql end declare section;
DCApplication()
{    read input msg;
     exec sql begin work;
    exec sql update accounts set Abalance = Abalance + :delta where Aid = :Aid;
    exec sql select Abalance into :Abalance from accounts where Aid = :Aid;
    exec sql update tellers  set Tbalance = Tbalance + :delta where Tid = :Tid;
    exec sql update branches set Bbalance = Bbalance + :delta  where Bid = :Bid;
    exec sql insert into history(Tid, Bid, Aid, delta, time)  values (:Tid, :Bid, :Aid, :delta, CURRENT);
    send output msg;
     exec sql commit work; }
```

A transaction
= A sequence of SQL statements
= A sequence of Reads and Writes

NOTE:
Most tables in OLTP are indexed!! Thus, index-based access!!

Database IO Architecture (OLTP vs. OLAP)



- A transaction consists of SQL statements executed in sequence
- Each SQL (Select/Insert/Delete/Update)
  - Select reads tuples from page(s) while Insert/Delete/Update changes records in page(s)
  - Thus, access one or more pages

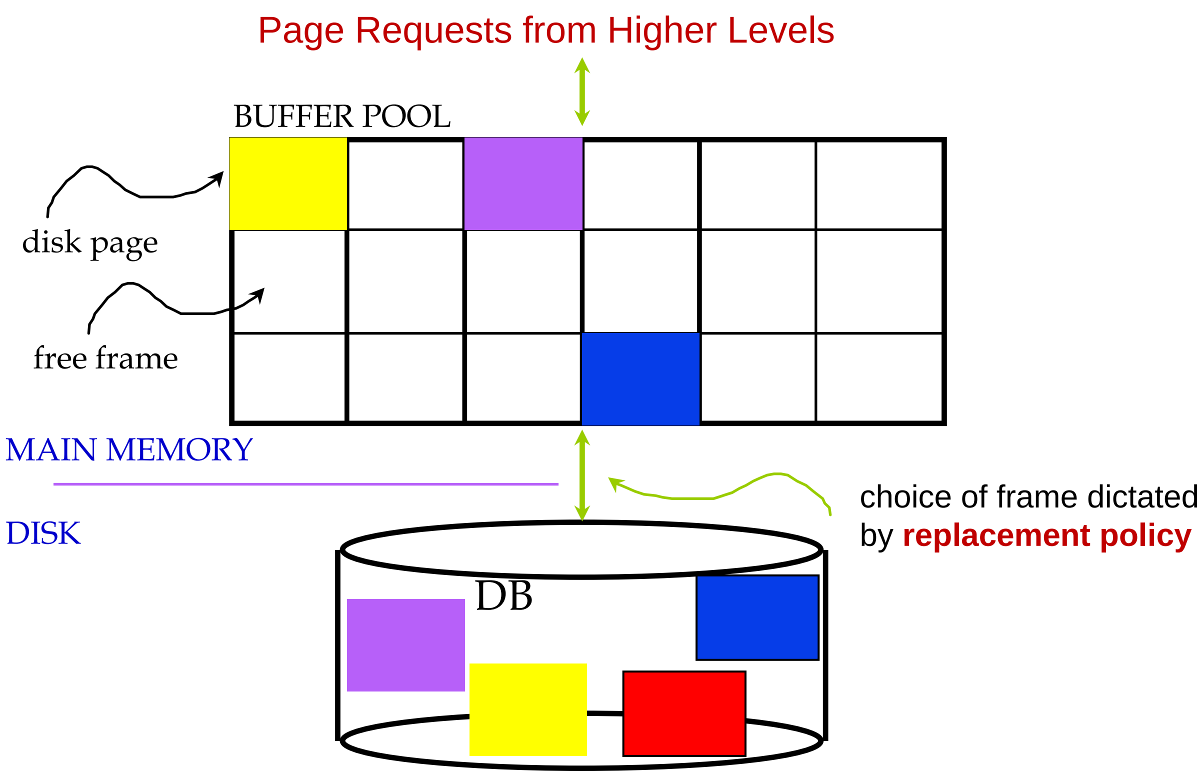
Query type and Index existence
-> Access method (FTS vs. IDX)

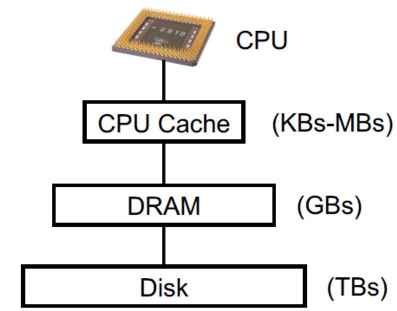
- When page(s) are in buffer (i.e., HIT): DRAM operation
- Otherwise (i.e., MISS), IOs are issued
  - read page(s) from the storage
  - In case of dirty victim, write page to the storage
  - Also, checkpoint writes (chap 18)

IO patterns
: Random vs. Sequential
: Hot vs. cold page writes

9.4 Buffer Management in a DBMS

- Data must be in RAM for DBMS to operate on it!





Typically want to cache **frequently accessed data** at a high level of the storage hierarchy to improve performance

Why Small Buffer Cache Works?
Access Skew and Temporal Locality Exist in the access pattern

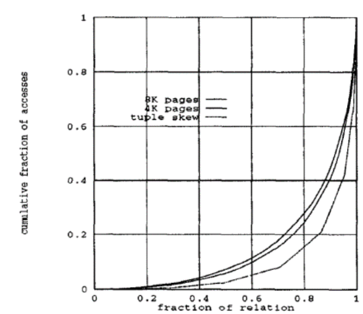


Figure 5: Stock Relation CDF

When a Page is Requested …

- Buffer pool information table: <frame#, pageid, pin_cnt, dirty>
  - In big systems, it is not trivial to just check whether a page is in pool

- If requested page is not in pool:
  - Choose a frame for replacement
  - If frame is dirty, write it to disk
  - Read requested page into chosen frame
- Pin the page and return its address.

- If requests can be predicted (e.g., sequential scans) pages can be pre-fetched several pages at a time!

More on Buffer Management

- Requestor of page must unpin it, and indicate whether page has been modified (using dirty bit)

- Page in pool may be requested many times,
  - a pin_count is used.
  - a page is a candidate for replacement iff pin_count = 0.

- CC & recovery may entail additional I/O when a frame is chosen for replacement. (e.g. Write-Ahead Log protocol)

Buffer Manager Pseudo Code

```
void* BufferManager.getPage( pid int ) {
        void * frame = search_buffer-for_page(pid);
        if ( !frame ) {
                // we get here if the requested page is not in the buffer
                frame = get_next_empty_frame();
                if ( !frame ) // and we ran out of space…
                        frame = replacement_policy.choose();
                if ( frame.dirty ) { // only if we allow to 'steal' a frame
                        write_page(frame);
                        frame.dirty = false;
                }
                load(pid, frame);
        }
        frame.pin ++;
        return frame;
}
```

[Source: Uwe Röhm's Slide]

Buffer Replacement Policy

- Hit vs. miss
- Hit ratio = # of hits / (# of page requests to buffer cache)
  - One miss incurs one (or two) physical IO. Hit saves IO.
  - Rule of thumb: at least 80 ~ 90%
- Problem: for the given (future) references, which victim should be chosen for highest hit ratio (i.e. least # of IOs)?
  - Numerous policies
  - Does one policy win over the others?
  - One policy does not fit all reference patterns!

Buffer Replacement Policy (2)

- Frame is chosen for replacement by a replacement policy*:*
  - Random, FIFO, LRU, MRU, LFU, Clock etc.
  - Replacement policy can have big impact on # of I/O's; depends on the access pattern

- For a given workload, one replacement policy, A, achieves 90% hit ratio and the other, B, does 91%.
  - How much improvement? 1% or 10%?
  - We need to interpret its impact in terms of miss ratio, not hit ratio

Buffer Replacement Policy (3)

- Least Recently Used (LRU)
  - For each page in buffer pool, keep track of time last *unpinned*
  - Replace the frame that has the oldest (earliest) time
  - Very common policy: intuitive and simple
  - Why does it work?
    - ✓ ``Principle of (temporal) locality" (of references) (https://en.wikipedia.org/wiki/Locality_of_reference)
    - ✓ Why temporal locality in database?: e.g. hot items, insertion to heap files
  - The correct implementation is not trivial
    - ✓ Especially in large scale systems: e.g. time stamp
- Variants
  - Linked list of buffer frames, LRU-K, 2Q (Linux cache), midpoint-insertion and touch count algorithm (Oracle, MySQL/InnoDB), Clock (MS-SQL), ARC, LIRS ...
  - Implication of big memory: "random" > "LRU"??


All About Oracle's Touch Count Algorithm

Buffer Replacement Policies (4): Taxonomy

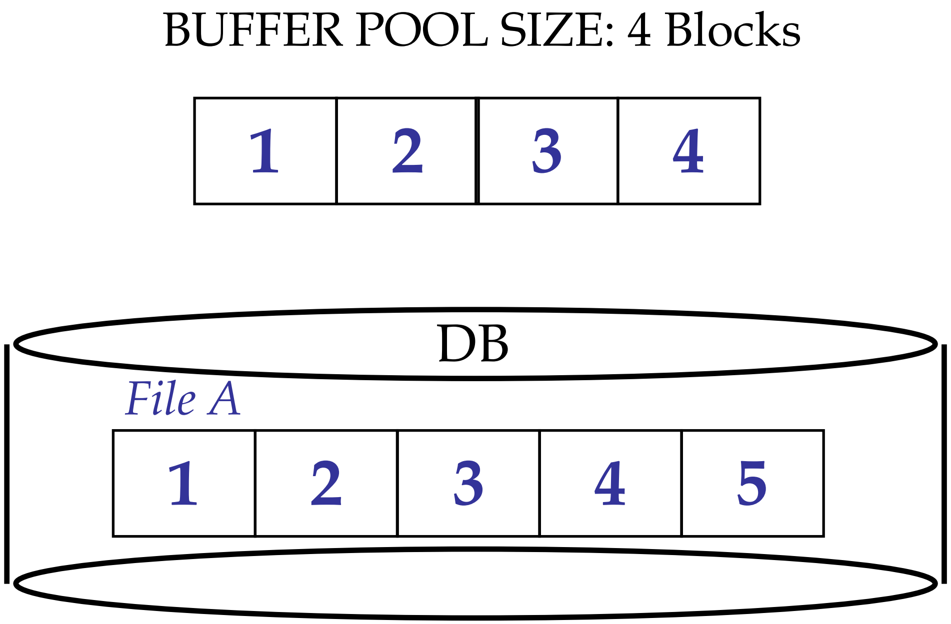
- Recency vs. Frequency

replacement decision based on		Age		
		no	**since last usage**	**since arrival**
References	**none**	RANDOM		FIFO
	last reference		LRU MRU CLOCK	
	all references	LFU	GCLOCK	

[Source: Uwe Röhm’s Slide]

Buffer Replacement Policy (5): LRU is NOT Scan-Resistant

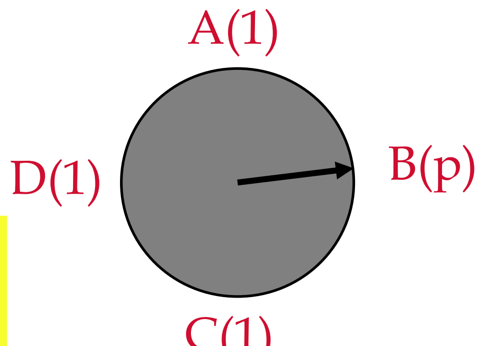
- Problem of LRU - sequential flooding
  - caused by LRU + repeated sequential scans.
  - # buffer frames < # pages in file means each page request causes an I/O. MRU much better in this situation (but not in all situations, of course).



- Assume repeated sequential scans of file A
- What happens when reading 5th blocks? and when reading 1st block again? ….

Buffer Replacement Policy (6): “Clock” Algorithm

- An approximation of LRU
- Arrange frames into a cycle, store one reference bit *per frame*
  - Can think of this as the 2nd chance bit
- When pin count reduces to 0, turn on reference bit
- When replacement necessary




```
do for each page in cycle {
        if (pincount == 0 && ref bit is on)
                turn off ref bit;
        else if (pincount == 0 && ref bit is off)
                choose this page for replacement;
    } until a page is chosen;
```

- ...ence counter

Buffer Replacement: Hit Ratio vs. Buffer Size

Why Small Buffer Cache Works?
Access Skew and Temporal Locality
Exist in the access pattern

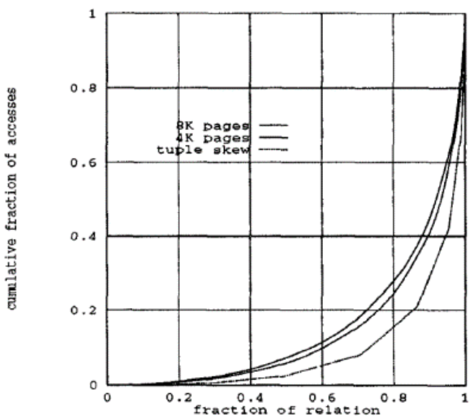
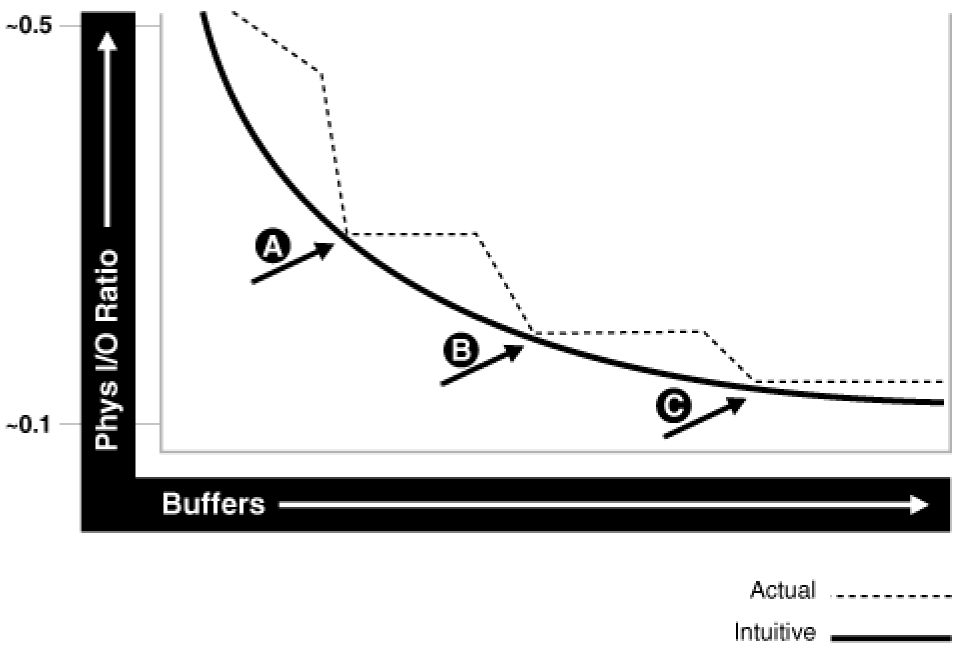


Figure 5: Stock Relation CDF



Very Large Data Bases

OLTP Through the Looking Glass [sigmod 08]

New Order Transaction

1. Select(whouse-id) from Warehouse
2. Select(dist-id, whouse-id) from District
3. Update(dist-id, whouse-id) in District
4. Select(customer-id, dist-id, whouse-id) f
5. Insert into Order
6. Insert into New-Order
7. For each item (10 items):
   (a) Select(item-id) from Item
   (b) Select(item-id,whouse-id) from Stock
   (c) Update(item-id,whouse-id) in Stock
   (d) Insert into Order-Line
8. Commit

New Order

begin
for loop(10)
.....Btree lookup(I), pin
Btree lookup(D), pin
Btree lookup (W), pin
Btree lookup (C), pin
update rec (D)
for loop (10)
.....Btree lookup(S), pin
.....update rec (S)
.....create rec (O-L)
.....insert Btree (O-L)
create rec (O)
insert Btree (O)
create rec (N-O)
insert Btree (N-O)
insert Btree 2ndary(N-O)
commit

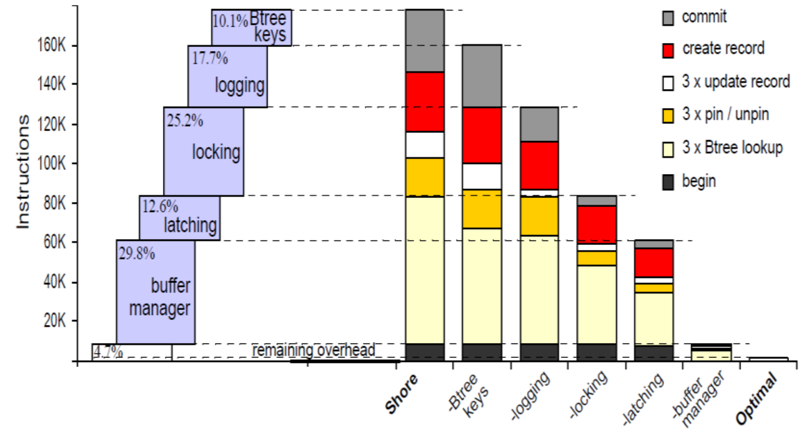


Figure 5. Detailed instruction count breakdown for Payment transaction.

Relation	New Order	Payment	Order Status	Delivery	Stock Level	Average
warehouse	U(1)	U(1)				0.87
district	U(1)	U(1)			P(1)	0.93
customer	NU(1)	NU(2.2)	NU(2.2)	P(10)		1.524
stock	NU(10)				P(200)	12.4
item	NU(10)					4.4
order	A(1)		P(1)	P(10)		0.53
new-order	A(1)			P(10)		0.49
order-line	A(10)		P(10)	P(100)	P(200)	13.3
history		A(1)				0.43

Belady's MIN Algorithm

- Theoretical optimal buffer replacement algorithm
  - "The *most* efficient caching algorithm would be to always discard the information that will not be needed for the longest time in the future. … Since it is generally impossible to predict how far in the future information will be needed, this is generally not implementable in practice. The practical minimum can be calculated only after experimentation, and one can compare the effectiveness of the actually chosen cache algorithm." (https://en.wikipedia.org/wiki/Cache_algorithms#B.C3.A9l.C3.A1dy.27s_Algorithm)
- Offline algorithm (vs. online algorithm)
  - *"All practical solutions are attempts to approximate the optimal Belady's MIN policy"* (from "Principles of Operating Systems: Design and Applications", Brian L. Stuart)
- Belady's anomaly
  - If the number of page frames is increased, would always the hit ratio will be higher or at least same?

Belady's Anomaly

- FIFO Algorithm: 3 vs. 4 page frames

All pages frames initially empty

		0	1	2	3	0	1	4	0	1	2	3	4	
Youngest page		0	1	2	3	0	1	4	4	4	2	3	3	
			0	1	2	3	0	1	1	1	4	2	2	
Oldest page				0	1	2	3	0	0	0	1	4	4	
		P	P	P	P	P	P	P			P	P		9 Page faults

(a)

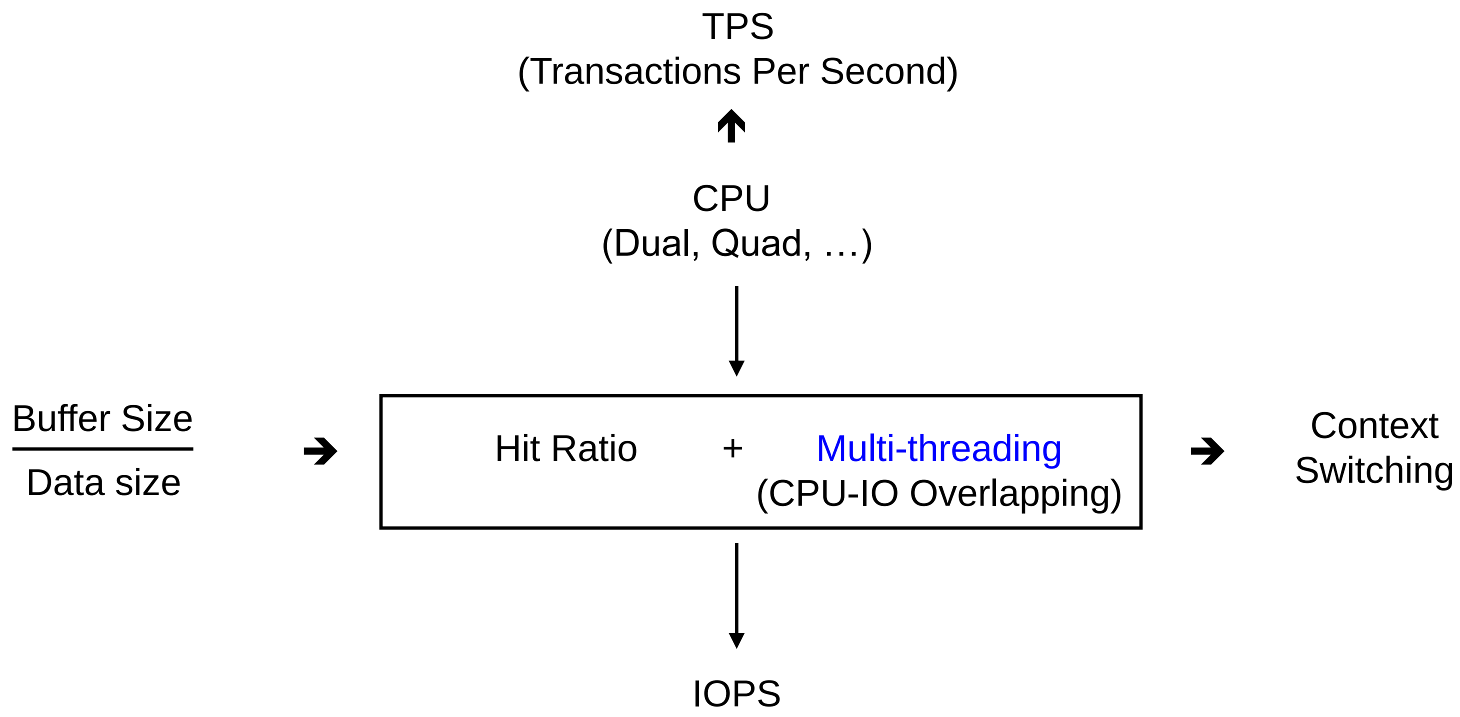
		0	1	2	3	0	1	4	0	1	2	3	4	
Youngest page		0	1	2	3	3	3	4	0	1	2	3	4	
			0	1	2	2	2	3	4	0	1	2	3	
Oldest page				0	1	1	1	2	3	4	0	1	2	
					0	0	0	1	2	3	4	0	1	
		P	P	P	P			P	P	P	P	P	P	10 Page faults

(b)

Source: https://www.cs.ucsb.edu/~chris

CPU-IO Overlapping

- 3 States: CPU Bound, IO Bound, Balanced



**For perfect CPU-IO overlapping,
IOPS Matters!**

IOPS Crisis in OLTP

<u>IBM for TPC-C (2008 Dec.)</u>

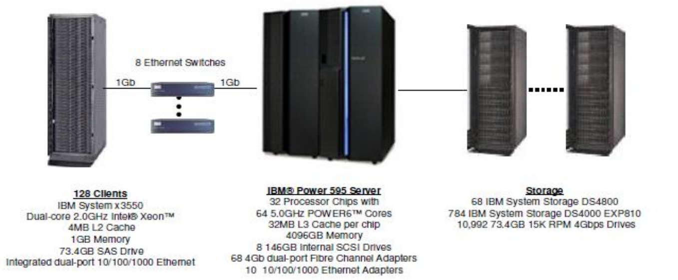
- 6M tpmC
- Total cost: 35M $
  - Server HW: 12M $
  - Server SW: 2M $
  - Storage: 20M $
  - Client HW/SW: 1M $

- They are buying IOPS, not capacity

IBM	IBM Power 595 Model 9119-FHA	TPC-C Rev. 5.9 / Report Date: June 10, 2008

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$17,111,788 USD	6,085,166	$2.81 USD	December 10, 2008

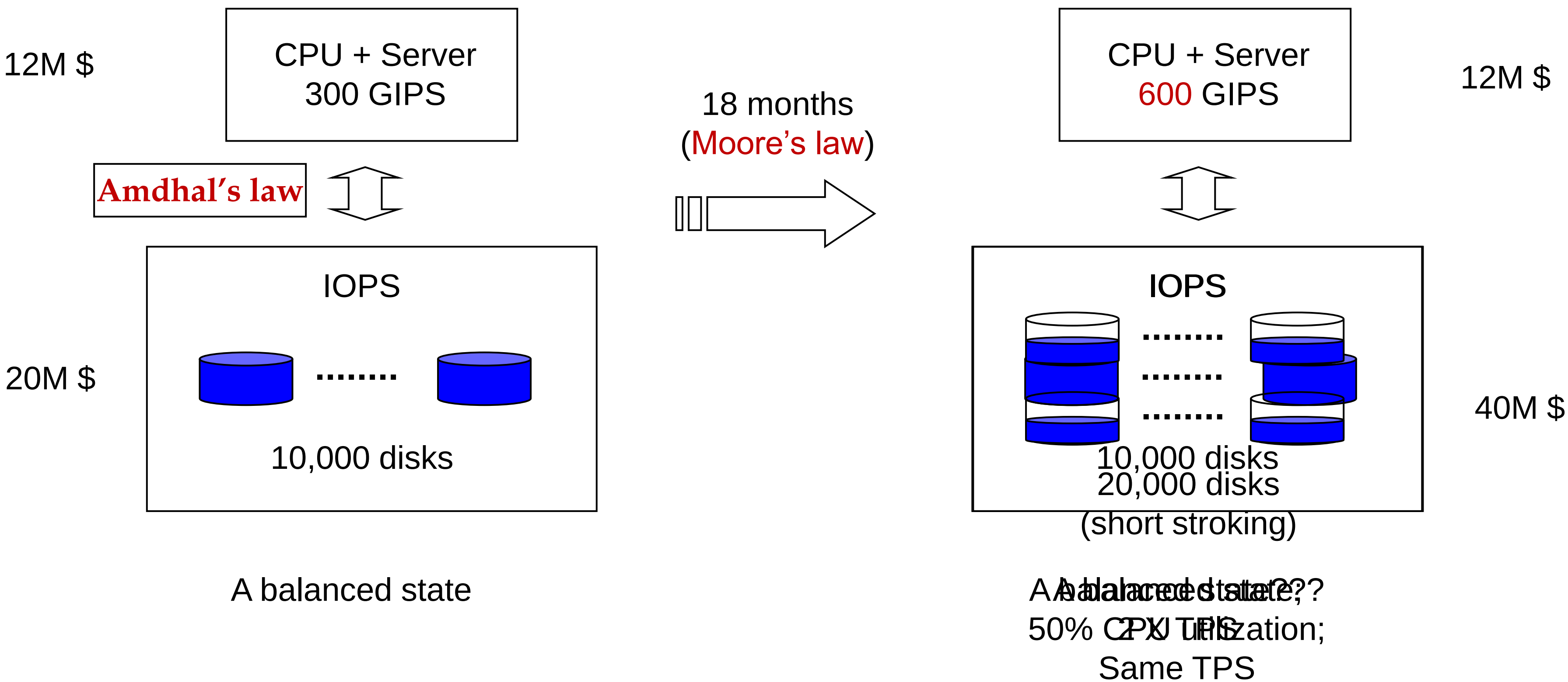
Database Server Processor Chip/Core/Thread	Database Manager	Operating System	Other Software	No. Users
32/64/128 POWER6 5.0GHz	DB2 9.5	AIX 5L V5.3	Microsoft Visual C++ Microsoft COM+	5,184,000



System Components	Server		Each of the 128 Clients	
	Quantity	Description	Quantity	Description
Processors Chips /Cores/Threads	32/64/128	5.0GHz POWER6	1/2/2	2.0GHz 4MB L2 Xeon Processor
Memory	64	64GB	2	512 MB
Disk Controllers	1 68 68	SAS Controller 4Gb FC Adapters IBM System Storage DS4800	1	SAS Controller
Disk Drives	8 10,992	146.8GB 15K RPM SCSI 73.4GB 15K RPM 4Gb FC	1	73.4GB 15K RPM SAS
Total Storage		746,467GB		67.86GB
Terminals	1	System Console	1	System Console

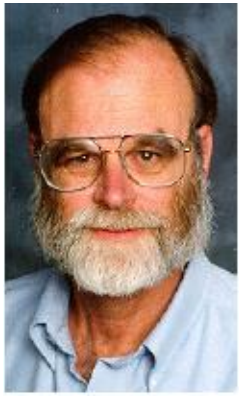
IOPS Crisis in OLTP(2)

- For **balanced systems**, OLTP systems pay $$$ on disks for high IOPS
  - vs. IO-bound, CPU-bound



Flash Disk Opportunity for Server Applications
(MS-TR 2007)

What If FLASH Disks Delivered Thousands of IO/s and Were "Big"?



Jim Gray

My tests and those of several others suggest that FLASH disks can deliver about 3K random 8KB reads/second and with some re-engineering about 1,100 random 8KB writes per second. Indeed, it appears that a single FLASH chip could deliver nearly that performance and there are many chips inside the "box" – so the actual limit could be 4x or more. But, even the current performance would be VERY attractive for many enterprise applications. For example, in the TPC-C benchmark, has approximately equal reads and writes. Using the graphs above, and doing a weighted average of the 4-deep 8 KB random read rate (2,804 IOps), and 4-deep 8 KB sequential write rate (1233 IOps) gives *harmonic average* of 1713 (1-deep gives 1,624 IOps). TPC-C systems are configured with ~50 disks per cpu. For example the most recent Dell TPC-C system has ninety 15Krpm 36GB SCSI disks costing 45k$ (with 10k$ extra for maintenance that gets "discounted"). Those disks are 68% of the system cost. They deliver about 18,000 IO/s. That is comparable to the requests/second of ten FLASH disks. So we could replace those 90 disks with ten NSSD if the data would fit on 320GB (it does not). That would save a lot of money and a lot of power (1.3Kw of power and 1.3Kw of cooling).

The current flash disks are built with 16 Gb NAND FLASH. When, in 2012, they are built with a 1 terabit part, the device will have 2TB of capacity and will indeed be able to store the TPC-C database. So we could replace a 44k$ disk array with a few (say 10) 400$ flash disks (maybe).

If one looks at the system diagram of the Samsung NSSD there are many opportunities for innovation. It suggests interesting RAID options for fault tolerance (combining the MSR-TR-2006-176 ideas with non-volatile storage map and a block-buffer, and with writing raid-5 stripes of data across the chip array) adding a battery, adding logic for copy-on-write snapshots, and so on. These devices enable whole new approaches to file systems. They are potential gap fillers between disks and RAM and they are interesting "hot data" storage devices in their own right.

"My tests and those of several others suggest that FLASH disks can deliver about 3K random 8KB reads/second and with some re-engineering about 1,100 random 8KB writes per second. Indeed, it appears that a single FLASH chip could deliver nearly that performance and there are many chips inside the "box" – so the actual limit could be 4x or more. But, even the current performance would be VERY attractive for many enterprise applications. For example, in the TPC-C benchmark, has approximately equal reads and writes. Using the graphs above, and doing a weighted average of the 4-deep 8 KB random read rate (2,804 IOps), and 4-deep 8 KB sequential write rate (1233 IOps) gives *harmonic average* of 1713 (1-deep gives 1,624 IOps). TPC-C systems are configured with ~50 disks per cpu. For example the most recent Dell TPC-C system has ninety 15Krpm 36GB SCSI disks costing 45k$ (with 10k$ extra for maintenance that gets "discounted"). Those disks are 68% of the system cost. They deliver about 18,000 IO/s. That is comparable to the requests/second of ten FLASH disks. So we could replace those 90 disks with ten NSSD if the data would fit on 320GB (it does not). That would save a lot of money and a lot of power (1.3Kw of power and 1.3Kw of cooling)." (excerpts from "Flash disk opportunity for server-applications")

Our Message in SIGMOD 2009

One FlashSSD can beat Ten Harddisks in OLTP
- Performance, Price, Capacity, Power – (in 2008)

- In 2015, one FlashSSD can beat several 10x or more HDDs in OLTP

- "My tests and those of several others suggest that FLASH disks can deliver about 3K random 8KB reads/second and with some re-engineering about 1,100 random 8KB writes per second. Indeed, it appears that a single FLASH chip could deliver nearly that performance and there are many chips inside the "box" – so the actual limit could be 4x or more. But, even the current performance would be VERY attractive for many enterprise applications. For example, in the TPC-C benchmark, has approximately equal reads and writes. Using the graphs above, and doing a weighted average of the 4-deep 8 KB random read rate (2,804 IOps), and 4-deep 8 KB sequential write rate (1233 IOps) gives *harmonic average* of 1713 (1-deep gives 1,624 IOps). TPC-C systems are configured with ~50 disks per cpu. For example the most recent Dell TPC-C system has ninety 15Krpm 36GB SCSI disks costing 45k$ (with 10k$ extra for maintenance that gets "discounted"). Those disks are 68% of the system cost. They deliver about 18,000 IO/s. That is comparable to the requests/second of ten FLASH disks. So we could replace those 90 disks with ten NSSD if the data would fit on 320GB (it does not). That would save a lot of money and a lot of power (1.3Kw of power and 1.3Kw of cooling)." (excerpts from Flash disk opportunity for server-applications (Jim Gray)

Flash-based TPC-C @ 2013 September

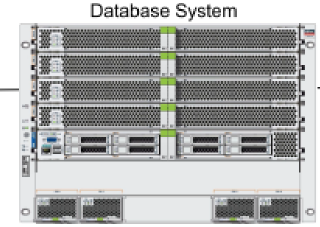
- Oracle + Sun Flash Storage
  - 8.5M tpmC
- Total cost: 4.7M $
  - Server HW: .6M $
  - Server SW: 1.9M $
  - Storage: 1.8M $
    - ✓ 216 400GB Flash Module: 1.1M $
    - ✓ 86 3TB 7.2K HDD : 0.07M
  - Client HW/SW: 0.1M $
  - Others: 0.1M$
- Implications
  - More vertical stacks (by SW vendor)
  - Harddisk vendors (e.g. Seagate)

ORACLE	SPARC T5-8 Server			TPC-C 5.11.0 TPC-Pricing 1.7.0 Report Date March 26, 2013
Total System Cost	TPC-C Throughput	Price/Performance		Availability Date
$4,663,073USD	8,552,523 tpmC	$0.55USD/tpmC		September 25, 2013
Database Server Processors/Cores/Threads	Database Manager	Operating System	Other Software	Number of Users
SPARC T5 3.6GHz 8 / 128 / 1,024	Oracle Database 11g Release 2 Enterprise Edition With Oracle Partitioning	Oracle Solaris 11.1	Oracle Tuxedo CFSR Oracle Web Tier 1	6,800,000

Clients

16 Sun Server X3-2
2 Intel® Xeon®
E5-2690 2.93GHz
64GB Memory
2 600GB SAS disk

Database System



SPARC T5-8 Server
8 SPARC T5 3.6GHz
4TB Memory
2 600GB 10K RPM SAS
12 8Gb/s FC HBA, 2 port
10GbE SFP+

Storage

54 DATA COMSTAR
Sun Server X3-2L w/
2 Intel® Xeon® E5-2609 2.4GHz
16GB DDR3
46 w/ 1 3TB 7.2K RPM SAS
8 w/ 2 3TB 7.2K RPM SAS
4 F40 PCI-E 400GB SSD

2 REDO COMSTAR
Sun Server X3-2L w/
2 Intel® Xeon® E5-2609 2.4GHz
16GB DDR3
12 3TB 7.2K RPM SAS

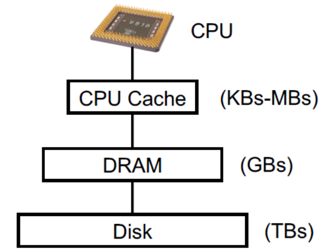
System Component	Each Server Node		Each Client	
Processors/Cores/Threads and cache	8/128/1024	SPARC T5 3.6GHz 8MB L3 Cache	2/16/32	Intel® Xeon® E5-2690 12MB Smart Cache
Memory		4TB		64GB
Disk Controllers	12	8Gb/s FC HBA 2 Port	1	8 port Internal SAS
OS Disks (each system)	2	600GB 10K RPM SAS	2	600GB 10K RPM SAS
External Storage	216 86	Flash Accelerator F40 PCI-E 400GB eMLC 3TB 7.2K RPM SAS		
Total Storage		344.4TB		

Very Large Data Bases

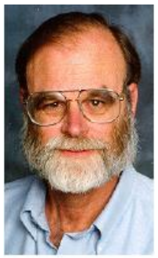
MMDBMS vs. All-Flash DBMS: Personal Thoughts

- Why MMDBMS has been recently popular?
  - Sap Hana, MS Hekaton, Oracle In-memory, Altibase, ….
  - The price of DRAM had ever dropped for the last two decades
    - ✓ $/IOPS @ DISK >> $/GB @ DRAM
  - The overhead of disk-based DBMS is not negligible
  - Applications with extreme performance requirements?
- Flash storage
  - Lowered $/IOPS
  - $/IOPS @ SSD << $/GB @ DRAM
- MMDBMS vs. ***All-Flash*** DBMS (with some optimizations)
  - Winner? Time will tell: read **"Umbra: A Disk-Based System with In-Memory Performance@CIDR2020"**
  - But, in 2017, the average DRAM DDR4 price has increased by 2.3.
    - ✓ ``Reducing DRAM footprint with NVM in Facebook" (Eurosys 2018)

Buffer Management vs. 5 Min Rule & Pareto's Law



- Five minute rule



  - "The 5 Minute Rule for Trading Memory Accesses for Disc Accesses", SIGMOD record 1985;
  - See also "The Five-minute Rule Thirty Years Later and its Impact on the Storage Hierarchy", ADMS, 2017.
  - "How much high-tier storage size should we have?"
    - ✓ Can determine based on data's access frequency and devices' cost
  - @1987: If a data object is accessed in every 5 minutes or more frequently, it should be memory-resident, not in harddisk @ 1987
  - @2017: 3 hour rule with harddisk; this justifies MMDBMS
  - @2017: 10 second rule with low latency flash SSD
- Pareto's law: 80/20 rules
  - E.g. A Modeling Study of the TPC-C Benchmark, SIGMOD 1993
    - ✓ **84 % of accesses go to 20 % of the tuples**.

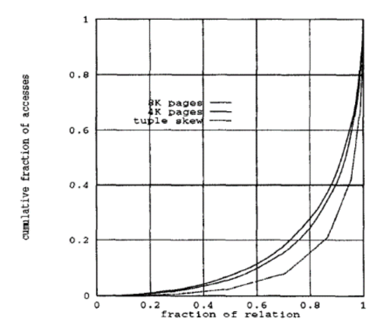


Figure 5: Stock Relation CDF

DBMS vs. OS File System

OS does disk space & buffer mgmt: why not let OS manage these tasks? (See appendix slides for more issues)

- Differences in OS support: portability issues
- Some limitations, e.g., files can't span disks.
  - note, this is changing --- OS File systems are getting smarter (i.e., more like databases!)
- Buffer management in DBMS requires ability to:
  - pin a page in buffer pool, force a page to disk (important for implementing CC & recovery),
  - adjust replacement policy, and pre-fetch pages based on access patterns in typical DB operations.

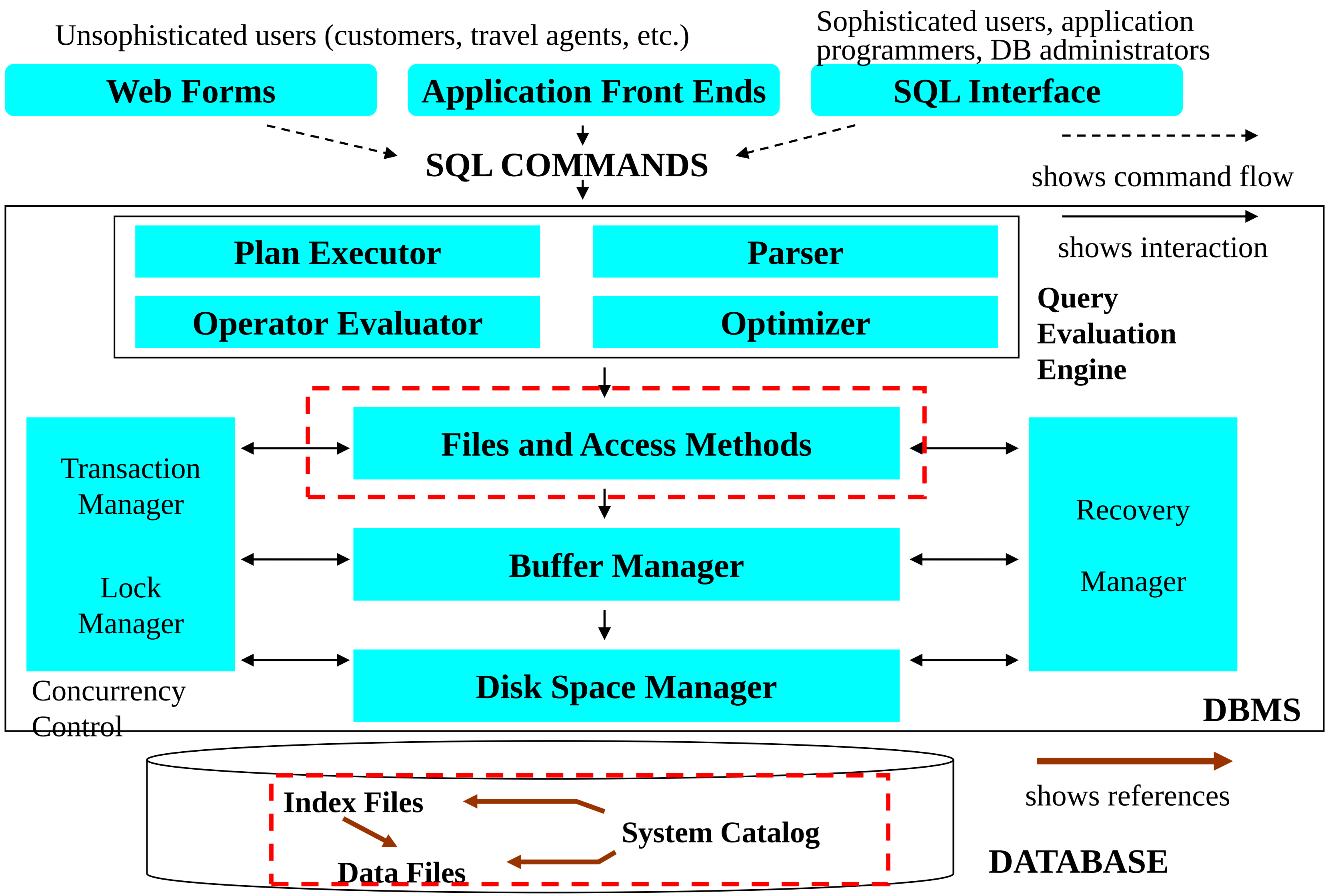


Figure 1.3 Architecture of a DBMS

How to Physically Store a Logical Table in Disk?

- Logical database level
  - DB: a set of relations, Relation: a set of records(or tuples), Tuple: a sequence of attributes
  - e.g. CREATE TABLE EMP (

```
empno NUMBER PRIMARY KEY,
ename VARCHAR(30),
 …
);
```

- Physical database level
  - How to represent SQL data types, a (variable-length) tuple with several attributes, a collection of tuples in a page? A set of pages in a relation / index?
  - How to handle insert, update, delete?

9.5 Files of Records (or Tuples/Rows)

- Blocks interface for I/O between buffer manager and disk space manager, but...

- Higher levels of DBMS operate on records, and files of records.

- FILE (per table): a collection of pages, each containing a collection of records (which in general belong to same table). Must support the following operations:
  1. Insert/delete/modify record
  2. Fetch a particular record (specified using record id)
  3. Scan all records (possibly with some conditions on the records to be retrieved)

Unordered (Heap) Files

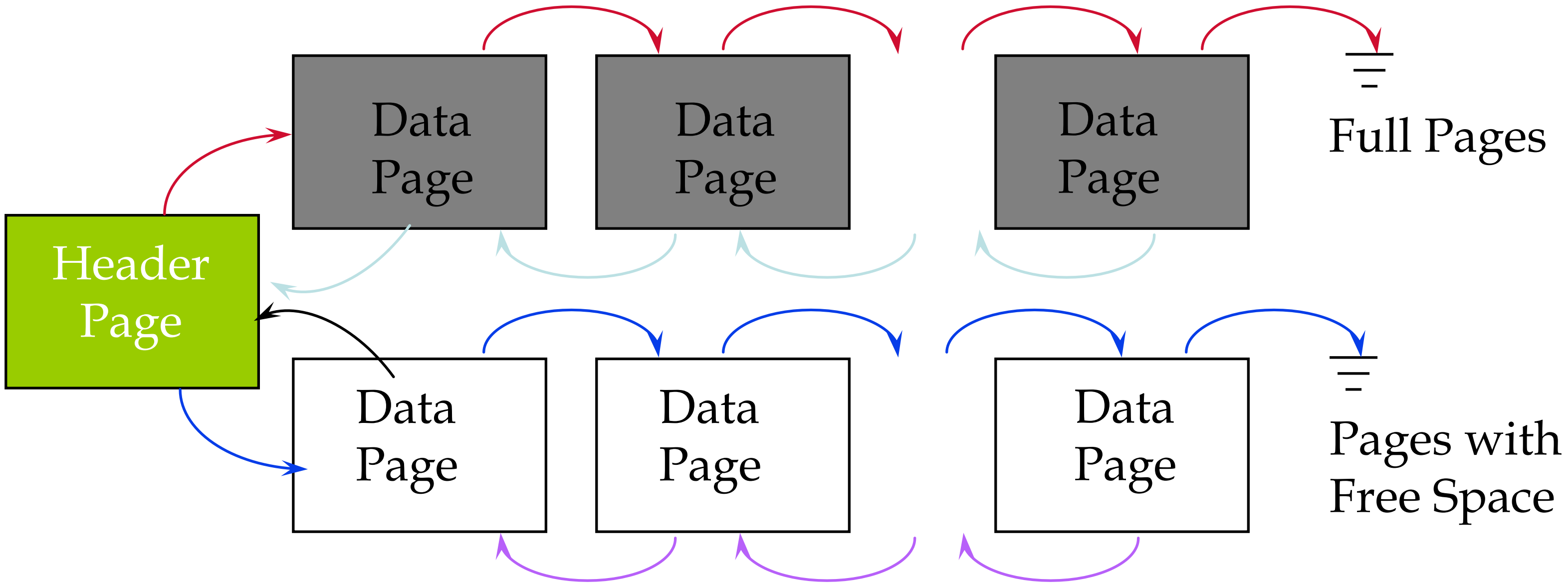
- Simplest file structure contains records in no particular order.
  - As a file grows and shrinks, disk pages are allocated and de-allocated.

- To support record level operations, we must:
  - keep track of the pages in a file
  - keep track of free space on pages
  - keep track of the records on a page

- There are many alternatives for keeping track of this

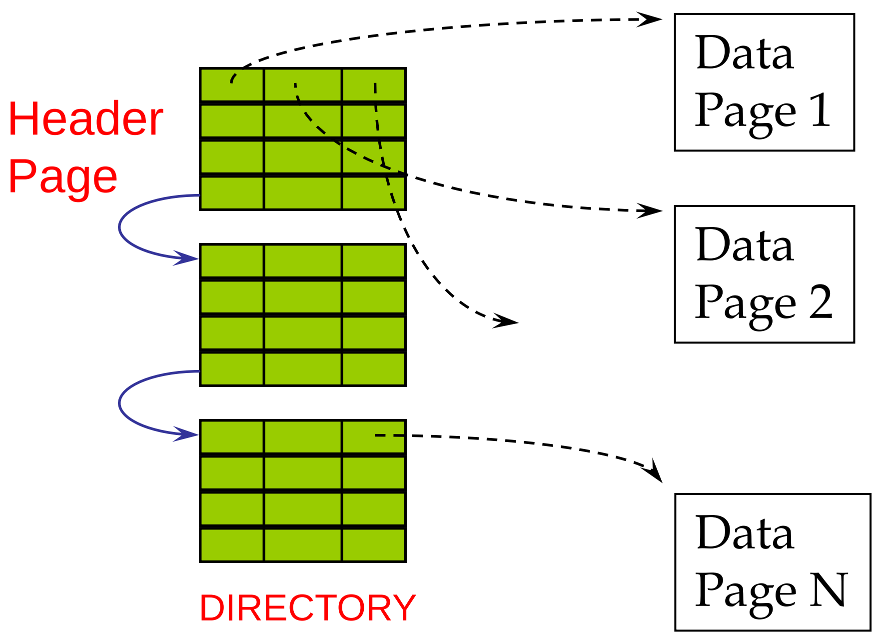
Heap File: Implemented as a List

- The header page id and heap file name must be stored someplace: database catalog
- Each page contains 2 'pointers' plus data.
- A critical problem: inefficient to find a page for record insertion.



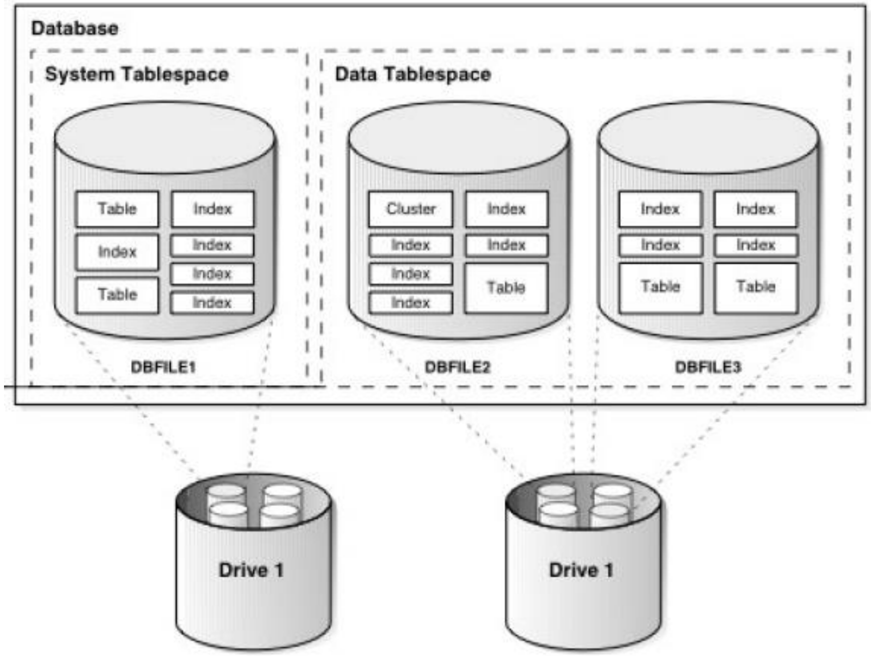
Heap File: Using a Page Directory

- The entry for a page can include # of free bytes on the page.
- The directory is a collection of pages; linked list implementation is just one alternative.
  - Much smaller than linked list of all HF pages!



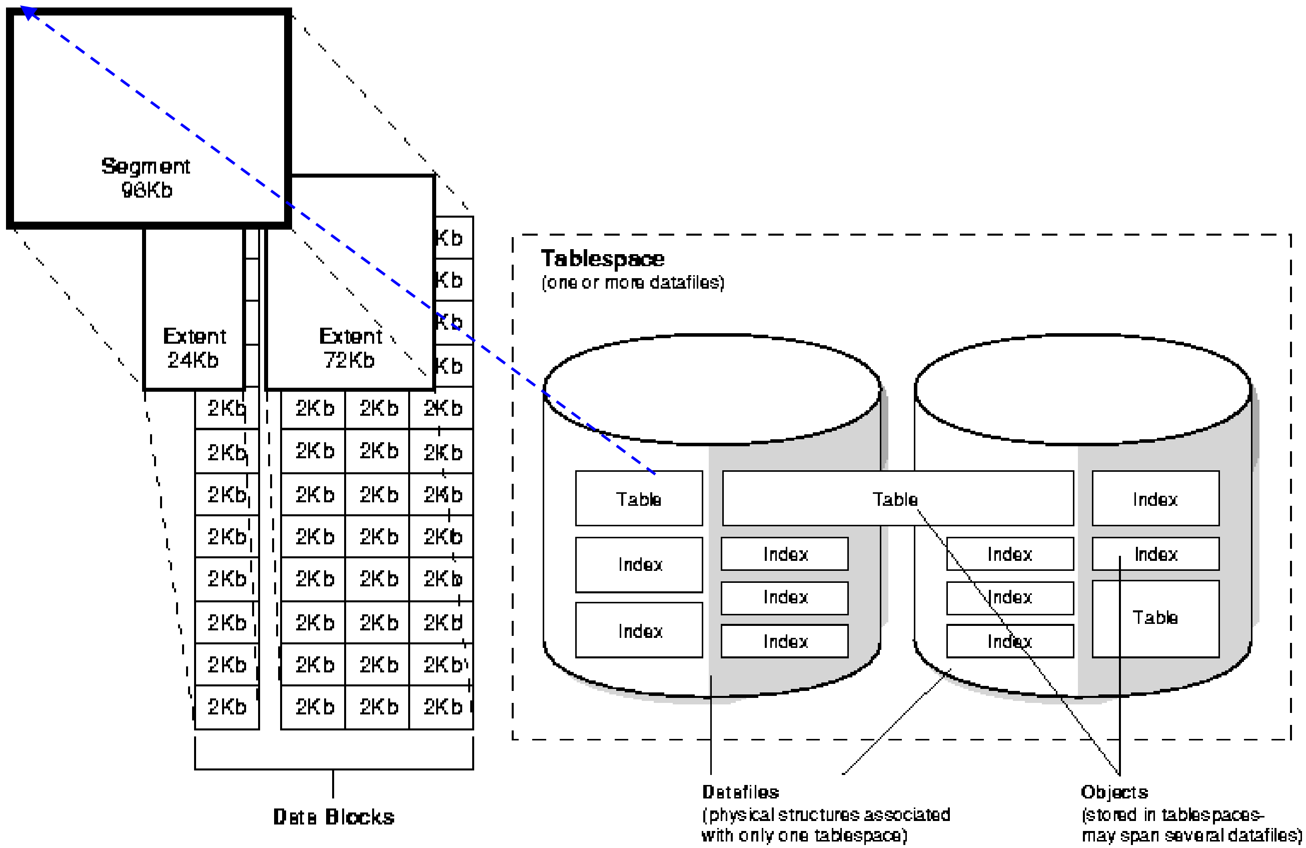
Oracle:
Tablespace, Segments, Extents, and Blocks

- Tablespace as logical DBMS 'file'
  - Consist of several physical files in the file system; may span several disks
- Separate data segments for
  - each table, index,...
- Data segments and hence Tablespaces can grow (by extents))
- An extent is the unit of disk space allocation: a sequence of disk blocks
- Rows/tuples are stored on disk blocks (or pages)



Very Large Data Bases

Oracle: Segments, Extents, and Blocks (2)



Oracle: Segments, Extents, and Blocks (3)

```
COLUMN table_name FORMAT a10
COLUMN tablespace_name FORMAT a10

select table_name, tablespace_name, blocks, pct_free, avg_row_len, avg_space
from user_tables
where table_name = 'TEST'; /* TEST table in Ch9-Script.sql */
```

```
TABLE_NAME      TABLESPACE_NAME     BLOCKS   PCT_FREE AVG_ROW_LEN  AVG_SPACE
--------------- --------------- ---------- ---------- ----------- ----------
TEST            USERS                 1000         10         664       1410
```


Very Large Data Bases

Oracle: Segments, Extents, and Blocks (4)

```
conn scott/tiger as sysdba

COLUMN segment_name FORMAT a10
COLUMN segment_type FORMAT a10

SELECT segment_name, segment_type, header_file, header_block,
          blocks,extents, max_extents
FROM dba_segments
WHERE segment_name = 'TEST';
```

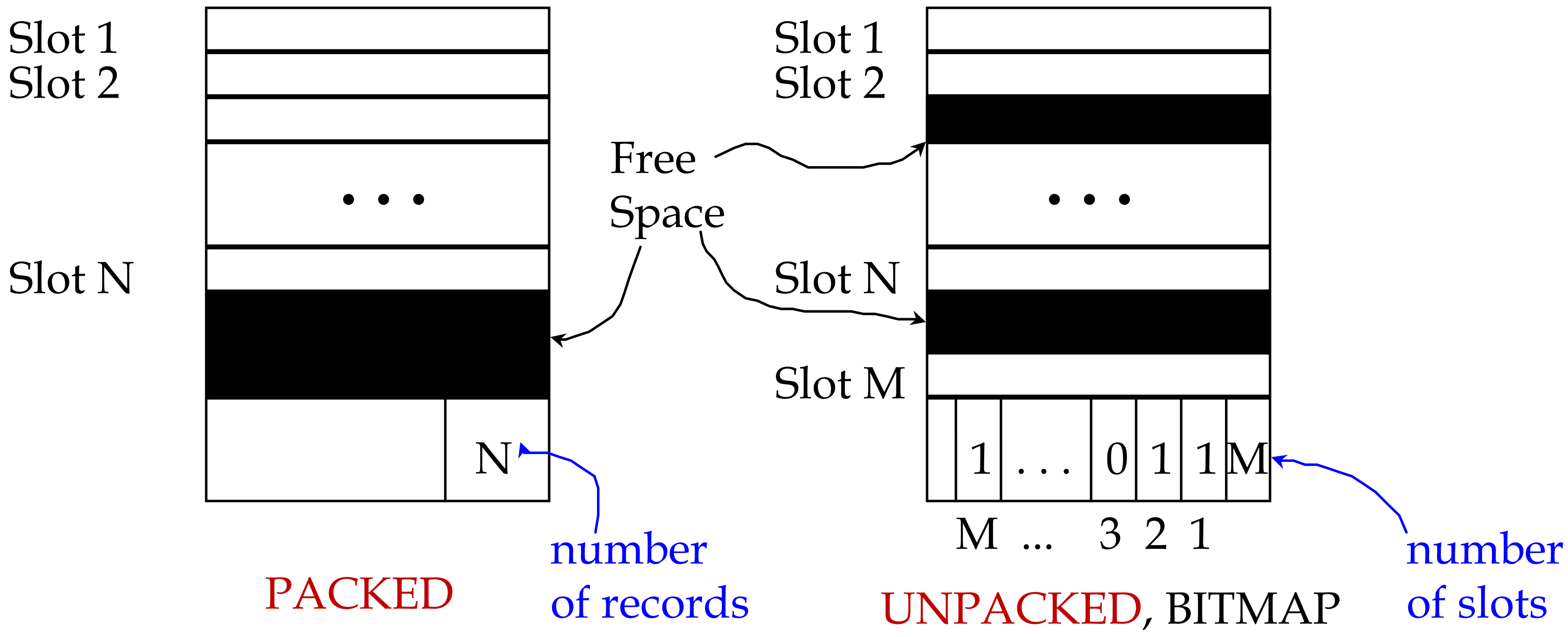
SEGMENT_NAME	SEGMENT_TYPE	HEADER_FILE	HEADER_BLOCK	BLOCKS	EXTENTS	MAX_EXTENTS
TEST	TABLE	4	363	1024	23	2147483645

Oracle: Segments, Extents, and Blocks (5)

```
COLUMN segment_name FORMAT a10
COLUMN tablespace_name FORMAT a10

SELECT  segment_name, tablespace_name, extent_id, file_id, block_id, blocks
FROM     dba_extents
WHERE  segment_name = 'TEST';
```

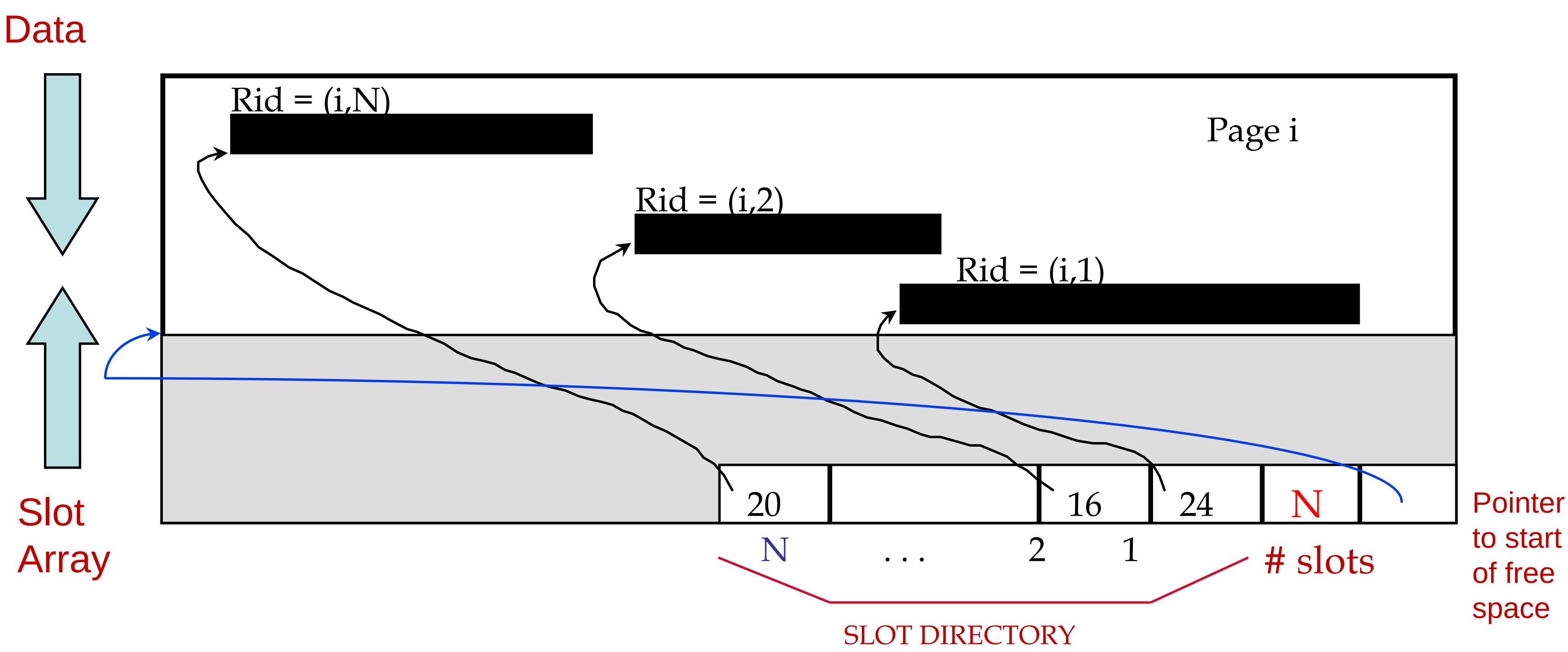
SEGMENT_NA	TABLESPACE	EXTENT_ID	FILE_ID	BLOCK_ID	BLOCKS
TEST	USERS	0	4	361	8
TEST	USERS	1	4	369	8
TEST	USERS	2	4	377	8
		…………………………………			
TEST	USERS	14	4	473	8
TEST	USERS	15	4	489	8
TEST	USERS	16	4	521	128
TEST	USERS	17	4	649	128
		…………………………………			
TEST	USERS	21	4	1161	128
TEST	USERS	22	4	1289	128

9.6 Page Formats: Fixed Length Records



- record id = <page id, slot #> /* can fetch a record using its record_id */
  - In first alternative (that is, left figure), moving records for free space management changes rid; may not be acceptable
    - ✓ why? external references

Page Formats: Variable Length Records
- Tuple-Oriented, Slotted Page Structure-



- Can move records on page without changing rid; so, attractive for fixed-length records too.
  - each slot entry = <record offset, length>
- Page is full when data space and slot array meet.

Block Dump in Oracle

```
conn scott/tiger as sysdba

select header_file, header_block, bytes, blocks
from dba_segments
where segment_name =  ‘EMP';
```

```
HEADER_FILE HEADER_BLOCK      BYTES     BLOCKS
----------- ------------ ---------- ----------
          4           27      65536          8
```

```
alter system dump datafile 4 block min 27 block max 30;
        --- Check the trace file in admin/udump/xxx.trc file.
```

Oracle Formatted Block Dumps

```
data_block_dump,data header at 0x3041074
===============
tsiz: 0x1f88
hsiz: 0x2e
pbl: 0x03041074
bdba: 0x0040c652
     76543210
flag=--------
ntab=1
nrow=14
frre=-1
fsbo=0x2e
fseo=0x1d51
avsp=0x1d23
tosp=0x1d23
0xe:pti[0]          nrow=14   offs=0
0x12:pri[0]         offs=0x1f62
0x14:pri[1]         offs=0x1f37
0x16:pri[2]         offs=0x1f0c
0x18:pri[3]         offs=0x1ee3
. . . . .
0x22:pri[8]         offs=0x1e16
0x24:pri[9]         offs=0x1deb
0x26:pri[10]        offs=0x1dc5
0x28:pri[11]        offs=0x1d9f
0x2a:pri[12]        offs=0x1d78
0x2c:pri[13]        offs=0x1d51
```

```
block_row_dump:
tab 0, row 0, @0x1f62
tl: 38 fb: --H-FL-- lb: 0x0  cc: 8
col  0: [ 3]  c2 4a 46
col  1: [ 5]  53 4d 49 54 48
col  2: [ 5]  43 4c 45 52 4b
col  3: [ 3]  c2 50 03
col  4: [ 7]  77 b4 0c 11 01 01 01
col  5: [ 2]  c2 09
col  6: *NULL*
col  7: [ 2]  c1 15
. . . . .
tab 0, row 13, @0x1d51
tl: 39 fb: --H-FL-- lb: 0x0  cc: 8
col  0: [ 3]  c2 50 23
col  1: [ 6]  4d 49 4c 4c 45 52
col  2: [ 5]  43 4c 45 52 4b
col  3: [ 3]  c2 4e 53
col  4: [ 7]  77 b6 01 17 01 01 01
col  5: [ 2]  c2 0e
col  6: *NULL*
col  7: [ 2]  c1 0b

end_of_block_dump
End dump data blocks tsn: 0 file#: 1
minblk 50769 maxblk 50770
```

Page Size

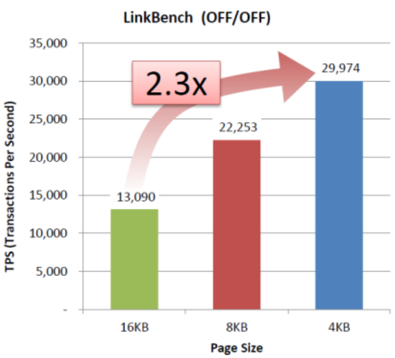
- Oracle
  - “Oracle recommends smaller Oracle Database block sizes (2 KB or 4 KB) for online transaction processing or mixed workload environments and larger block sizes (8 KB,16 KB, or 32 KB) for decision support system workload environments.” (see chapter “IO Config. And Design” in “Database Performance Tuning Guide” book)

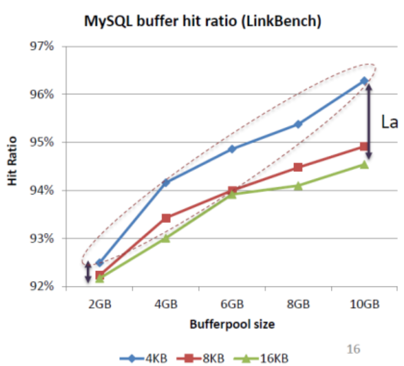
Table 17-3 Block Size Advantages and Disadvantages

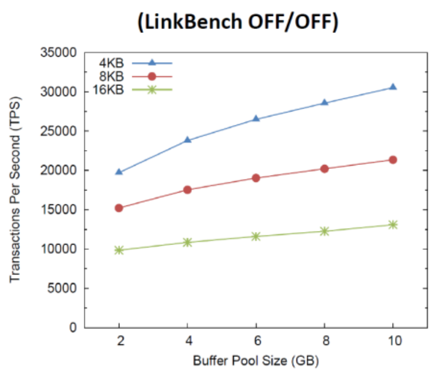
Block Size	Advantages	Disadvantages
Smaller	Good for small rows with lots of random access. Reduces block contention.	Has relatively large space overhead due to metadata (that is, block header). Not recommended for large rows. There might only be a few rows stored for each block, or worse, row chaining if a single row does not fit into a block,
Larger	Has lower overhead, so there is more room to store data. Permits reading several rows into the buffer cache with a single I/O (depending on row size and block size). Good for sequential access or very large rows (such as LOB data).	Wastes space in the buffer cache, if you are doing random access to small rows and have a large block size. For example, with an 8 KB block size and 50 byte row size, you waste 7,950 bytes in the buffer cache when doing random access. Not good for index blocks used in an OLTP environment, because they increase block contention on the index leaf blocks.

Page Size (2): e.g. InnoDB Page Tuning on SSD: 16KB -> 4KB

- Why better with smaller 4KB on SSD?
  - Higher hit ratio; Higher IOPS
  - Better throughput inside SSD
    - ✓ Better parallelism and less interference among requests
  - Less latch contention for hot pages (Contention Split @ CIDR '21)
  - Less space amplification (Bohyun's TPC-C exp.)







Random IOPS	Page Size 16KB	8KB	4KB
Read-only (128 threads)	29,870	57,847	89,083
Write-only (1-fsync)	196	206	225
Write-only (256-fsync)	4,563	7,978	12,647
Write-only (128 no-barrier)	13,446	25,546	49,009

(a) *DuraSSD*

Random IOPS	Page Size 16KB	8KB	4KB
Read-only (128 threads)	516	528	538
Write-only (128 threads)	428	439	444

(b) Harddisk (Seagate Cheetah 15K.6 146.8GB)

Table 2: Effect of page size on IOPS

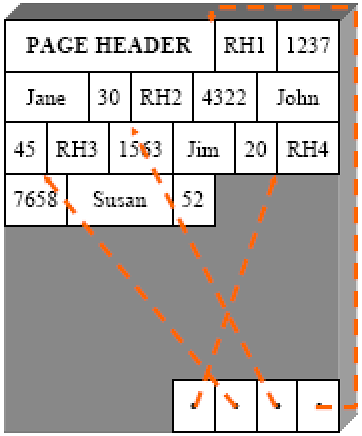
Source: Durable Write Cache in Flash Memory SSD for Relational and NoSQL Databases (SIGMOD 2014))

“Classical” Data Layout on Disk Page: Row-Store

- NSM (n-ary Storage Model, or Slotted Pages)

R

RID	SSN	Name	Age
1	1237	Jane	30
2	4322	John	45
3	1563	Jim	20
4	7658	Susan	52
5	2534	Leon	43
6	8791	Dan	37



Records are stored sequentially
Attributes of a record are stored together

Other Layouts: C-Store and Pax

- C-Store(DSM) and Pax(Hybrid)

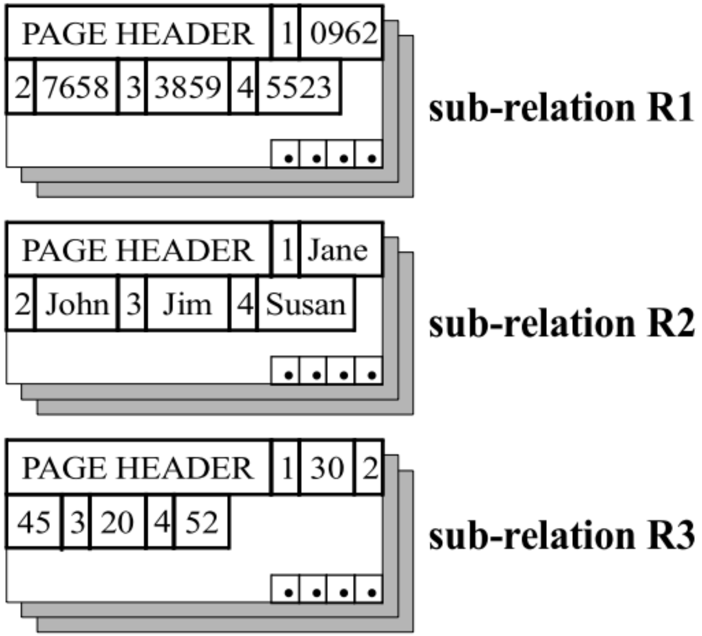


FIGURE 2: *The Decomposition Storage Model (DSM).*

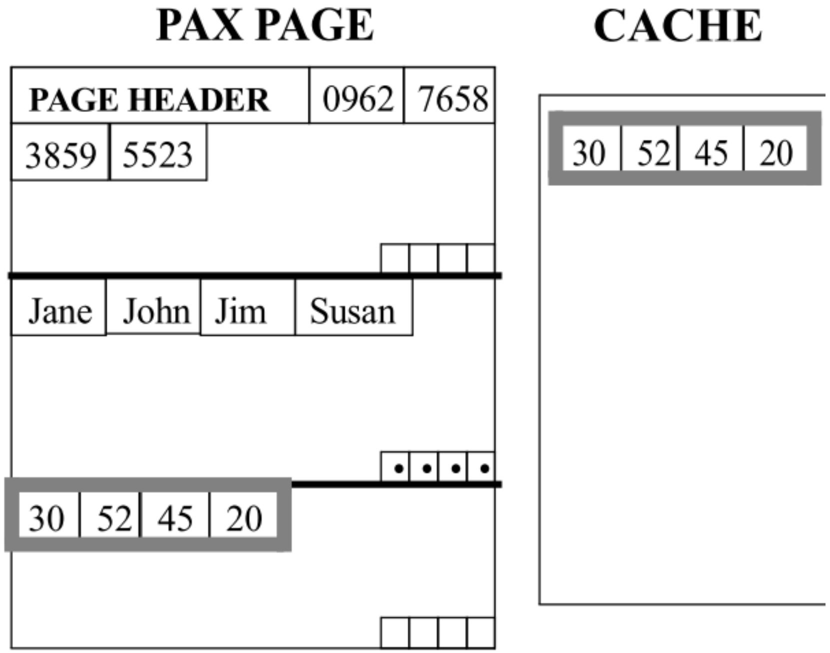
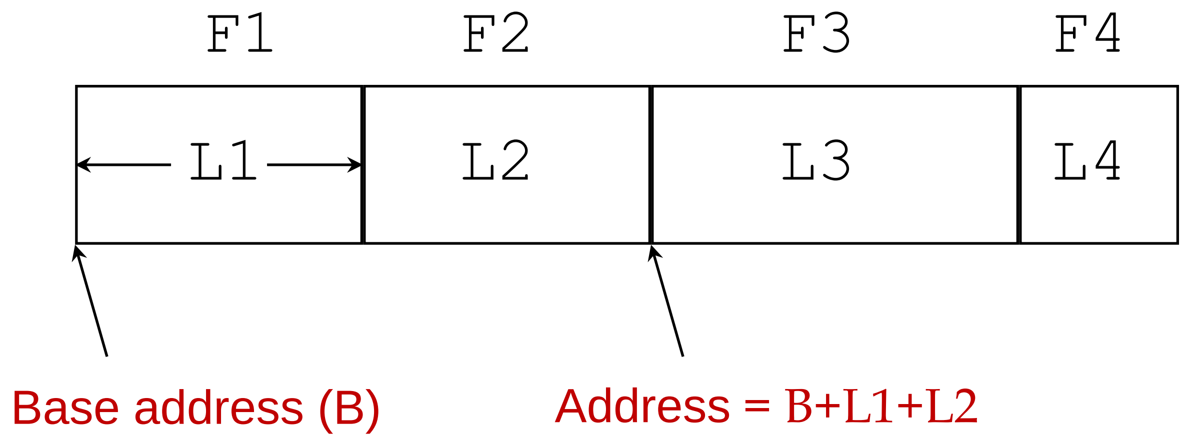


FIGURE 3: *The cache behavior of PAX.*

[Source: Weaving Relations for Cache Performance, Ailamaki et. al., VLDB 2001]

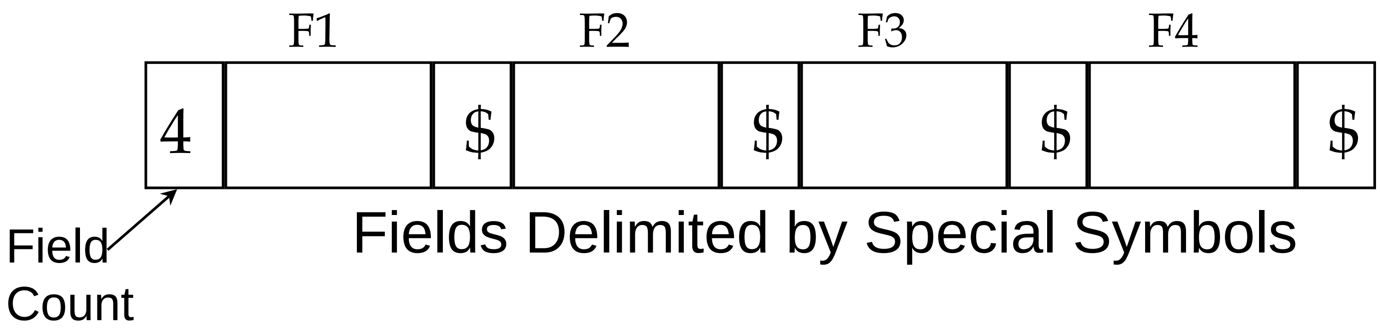
9.7 Record Formats: Fixed Length

- Information about field types same for all records in a file; stored in system catalogs.

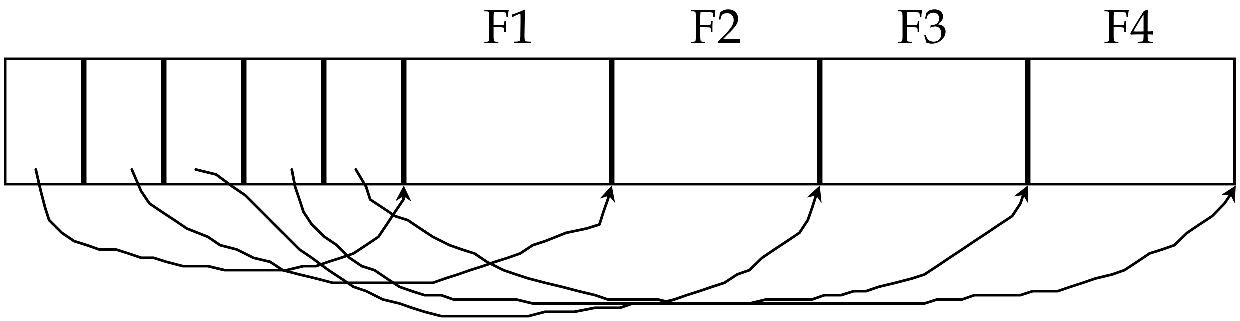


Record Formats: Variable Length

- Two alternative formats (# fields is fixed):



Fields Delimited by Special Symbols

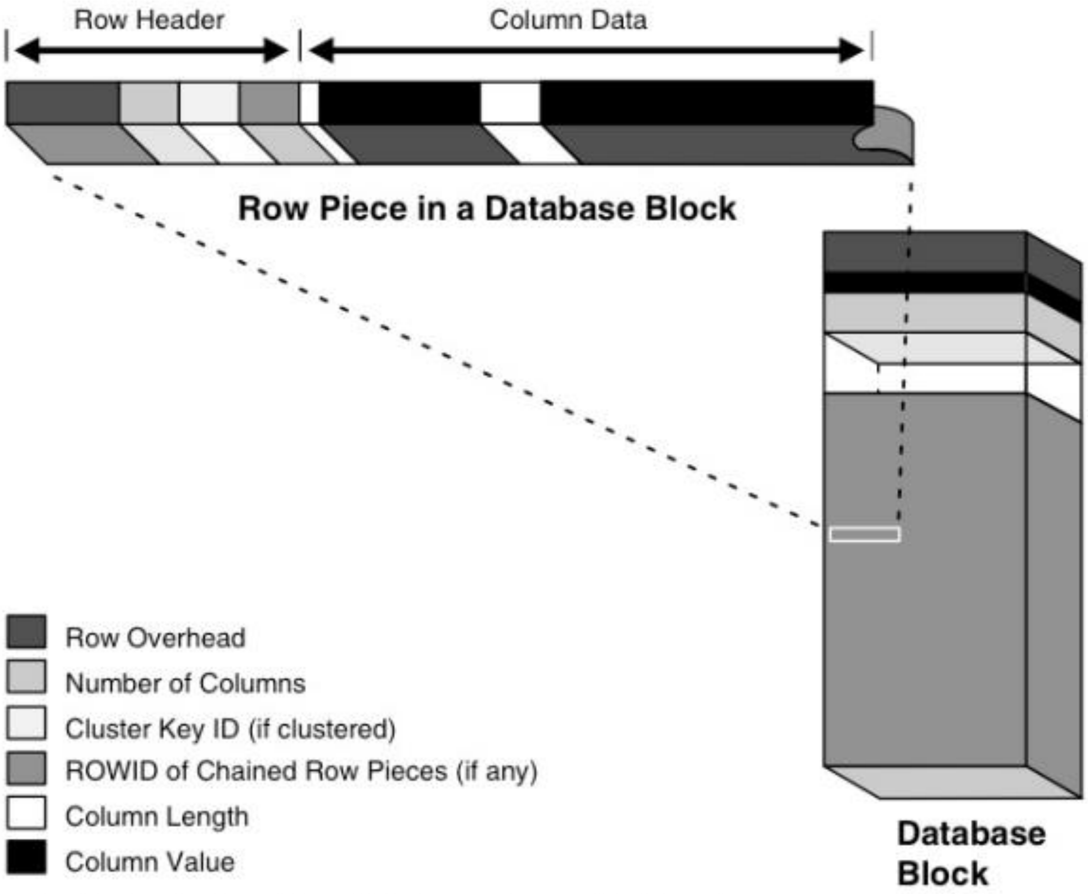


Array of Field Offsets

- Second offers direct access to i'th field, efficient storage of nulls;
  - small directory overhead.

Very Large Data Bases

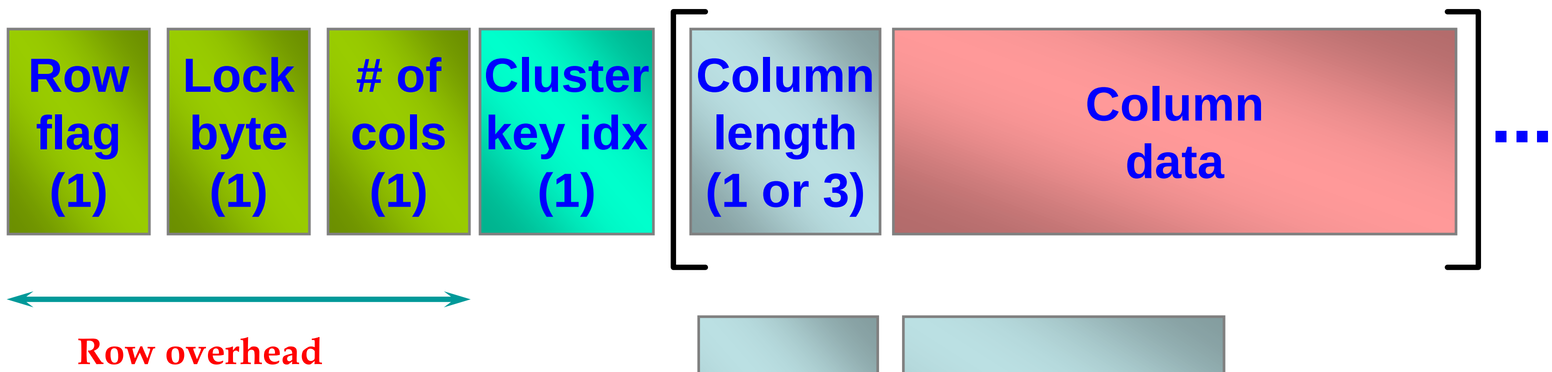
Row Layout in Oracle

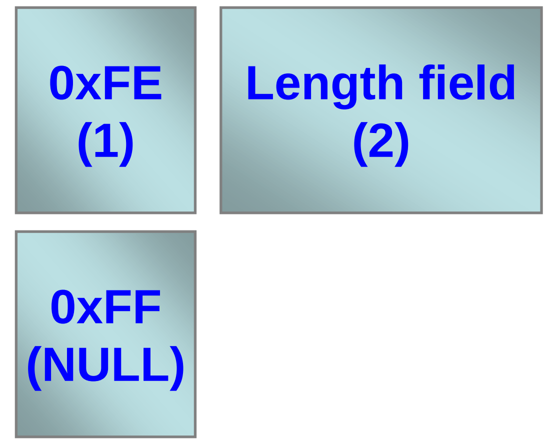


"Nulls are stored in the database if they fall between columns with data values.
In these cases they require 1 byte to store the length of the column (zero).
...
To conserve space, a null in a clumn only stores the column length (zero).
Oracle does not store data for the null column. Also, for trailing null columns, Oracle does not even store the column length."

[Oracle 10g Database Concepts, Chap. 5.4]

Row Layout in Oracle (2)





<u>tab 0, row 0, @0x1f62</u>
tl: 38 fb: --H-FL-- lb: 0x0 cc: 8
col 0: [3] c2 4a 46
col 1: [5] 53 4d 49 54 48
col 2: [5] 43 4c 45 52 4b
col 3: [3] c2 50 03
col 4: [7] 77 b4 0c 11 01 01 01
col 5: [2] c2 09
col 6: *NULL*
col 7: [2] c1 15

<u>fb: row migration</u>
-normal case: H FL
-Migration case
 - original row: H
 - migration row: FL

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80/12/17	800		20

Column Value in Oracle

```
=============== SQL DUMP function ===========
// dump() show s how a data value is stored in disk
select dump(a)
from test
where a = 1000;


DUMP(A)
------------------------------
Typ=2 Len=2: 194,11        ←Integer value 1000: type 2: int, column lengh 2 byte
                           ← data value 1000 is represented as (194, 11)
```

System Catalogs(or Data Dictionary)

- For each relation:
  - name, file name, file structure (e.g., Heap file)
  - attribute name and type, for each attribute
  - index name, for each index
  - integrity constraints
- For each index:
  - structure (e.g., B+ tree) and search key fields
- For each view:
  - view name and definition
- + statistics, authorization, buffer pool size, etc.
- See here for Oracle's Catalog (https://docs.oracle.com/cd/B13789_01/server.101/b10755/toc.htm)

Catalogs are themselves stored as relations!!

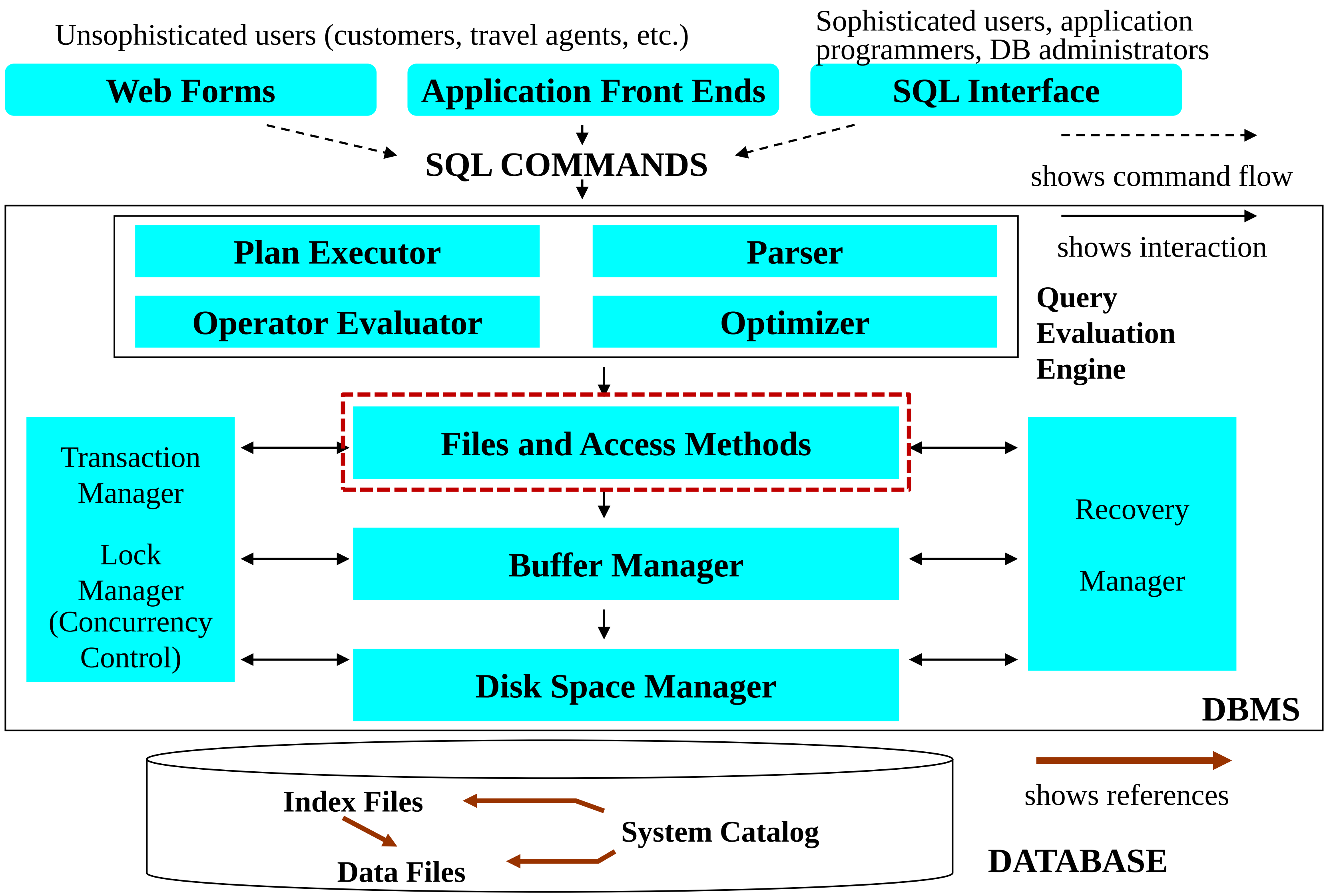


Figure 1.3 Anatomy of an RDBMS

Full Table Scan: An Access Method

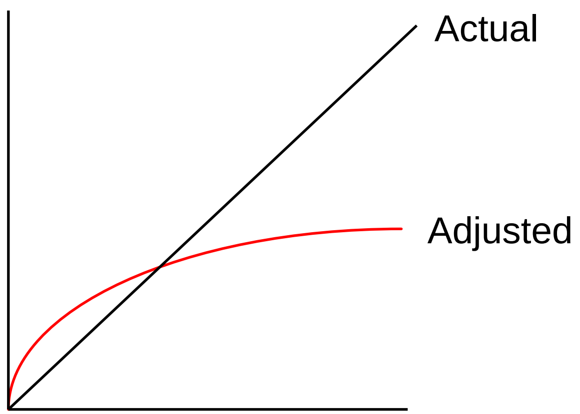
```
SELECT SUM(a)
FROM test;
```

- To process the above query, we should scan all the blocks of the 'TEST' table
  - FULL TABLE SCAN: the only access method for the query
  - Access method: data structure and algorithms for organizing and accessing data (e.g. heap file, index)
- # of blocks = 1000
- How many disk I/Os do you guess are necessary?
  - Cost model!!

Cost Model of Full Table Scan

- Full table scan
  - Cost = # of Blocks / Adjusted *db_file_multiblock_read_count* + 1 (in Oracle 9.2)
  - db_file_multiblock_read_count
  - Why adjust? memory caching!

Actual	Adjusted
4	**4.175**
8	**6.589**
16	10.398
32	**16.409**
64	**25.895**
128	**40.865**



(How the CBO works by Lewis, Jonathan, 2003 http://www.nocoug.org/download/2003-08/how_cbo_works.ppt)

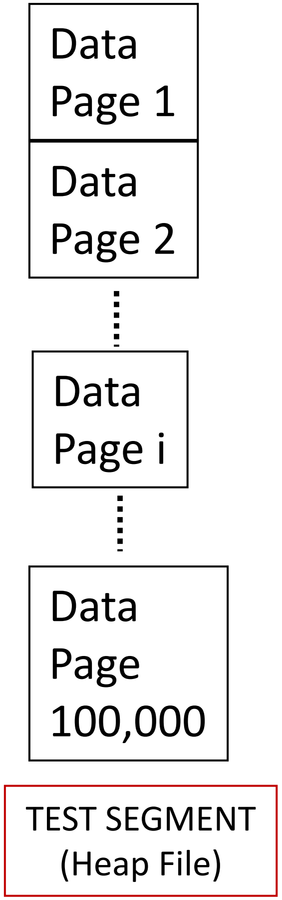
Estimated Cost of the Sample Query

- Cost = 1000 / 10.4 + 1 = 98
- Query optimizer estimates that the given query will incurs 98 disk I/Os
- What about the real disk I/O #?
- NOTE: estimated vs. real cost

OLAP vs. OLTP

- On-Line { Analytical vs. Transactional} Processing

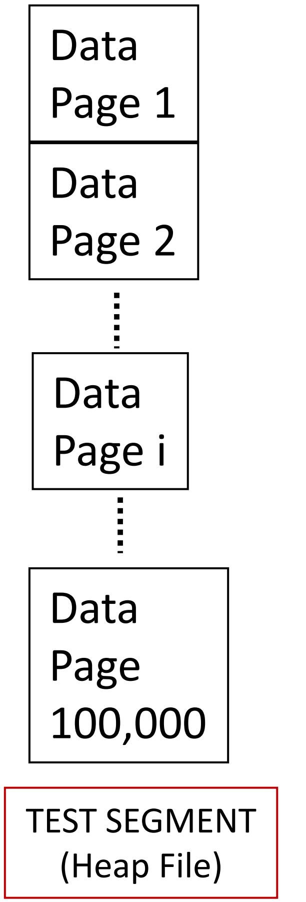
```
SQL> SELECT SUM(b) FROM TEST;
SUM(B)
----------
5.0000E+11
Execution Plan
--------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |     5 | 22053   (1)| 00:04:25 |
|   1 |  SORT AGGREGATE    |      |     1 |     5 |            |          |
|   2 |   TABLE ACCESS FULL| TEST |  996K|  4865K| 22053   (1)| 00:04:25 |
--------------------------------------------------------------------------
Statistics
----------------------------------------------------------
      179  recursive calls
        0  db block gets
   100152  consistent gets
   100112  physical reads
…..
        1  rows processed
```



OLAP vs. OLTP

- Point or Range Query

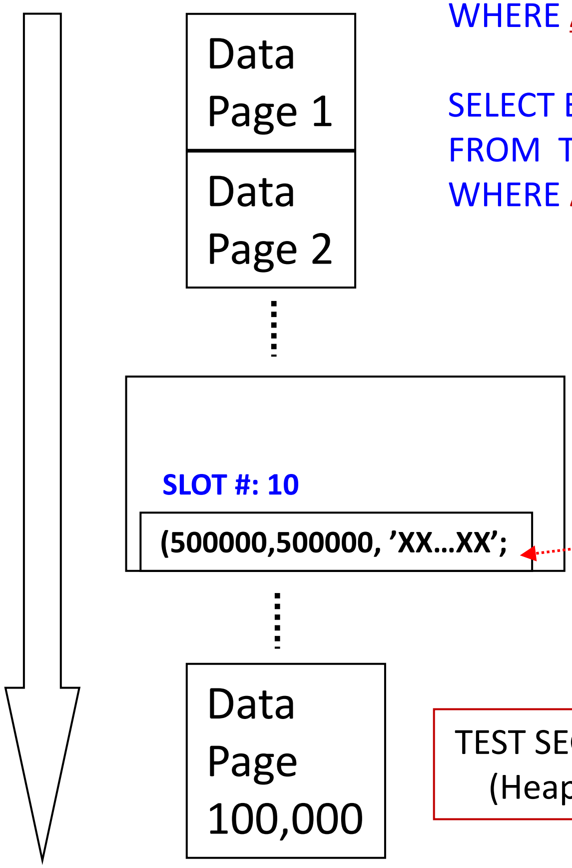
```
SQL> SELECT B FROM TEST WHERE A = 500000;
B
----------
 500000
Execution Plan
---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |     5 | 22053   (1)| 00:04:25 |
|   1 |  SORT AGGREGATE    |      |     1 |     5 |            |          |
|   2 |   TABLE ACCESS FULL| TEST |  996K|  4865K| 22053   (1)| 00:04:25 |
---------------------------------------------------------------------------
Statistics
----------------------------------------------------------
        179  recursive calls
          0  db block gets
     100152  consistent gets
     100112  physical reads
…..
          1  rows processed
```



OLAP vs. OLTP

CREATE INDEX TEST_A ON TEST(A);

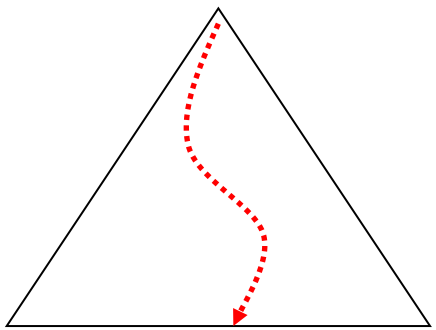
```
SELECT B            /* point query */
FROM  TEST
WHERE A = 500000;

SELECT B            /* range query */
FROM  TEST
WHERE A BETWEEN 50001 and 50101;
```



TEST SEGMENT
(Heap File)

B-tree Index on TEST(A)
(chapter 10.)

SEARCH KEY: 500000



(500000, (50000,10))

Cost:

- Full Table Scan: 100,000 Block Accesses
- Index: (3~4) + Data Block Access
  - Point query: 1
  - Range queries: depending on range

OLAP vs. OLTP

- Index-based Table Access

```
SQL> SELECT B FROM TEST WHERE A = 500000;
 B
----------
 500000
Execution Plan
-----------------------------------------------------------------------------------
| Id  | Operation                   | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |        |     1 |    10 |     4   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TEST   |     1 |    10 |     4   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | TEST_A |     1 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------
Statistics
----------------------------------------------------------
…..
          5  consistent gets
          4  physical reads
…..
          1  rows processed
```

Summary

- Disks provide cheap, non-volatile storage.
  - Random access, but cost depends on location of page on disk; important to arrange data sequentially to minimize *seek* and *rotation* delays.

- Buffer manager brings pages into RAM.
  - Page stays in RAM until released by requestor.
  - Written to disk when frame chosen for replacement (which is sometime after requestor releases the page).
  - Choice of frame to replace based on *replacement policy.*
  - Tries to *pre-fetch* several pages at a time.

- DBMS vs. OS File Support
  - DBMS needs features not found in many OS's, e.g., forcing a page to disk, controlling the order of page writes to disk, files spanning disks, ability to control pre-fetching and page replacement policy based on predictable access patterns, etc.

Summary (Contd.)

- Variable length record format with field offset directory offers support for direct access to i'th field and null values.
- Slotted page format supports variable length records and allows records to move on page
- File layer keeps track of pages in a file, and supports abstraction of a collection of records.
  - Pages with free space identified using linked list or directory structure (similar to how pages in file are kept track of).
- Catalog relations store information about the various database objects including relations, indexes, and views.

"If you want truly to understand a system, try to change it"

Kurt Lewin (A Psychologist)