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/)

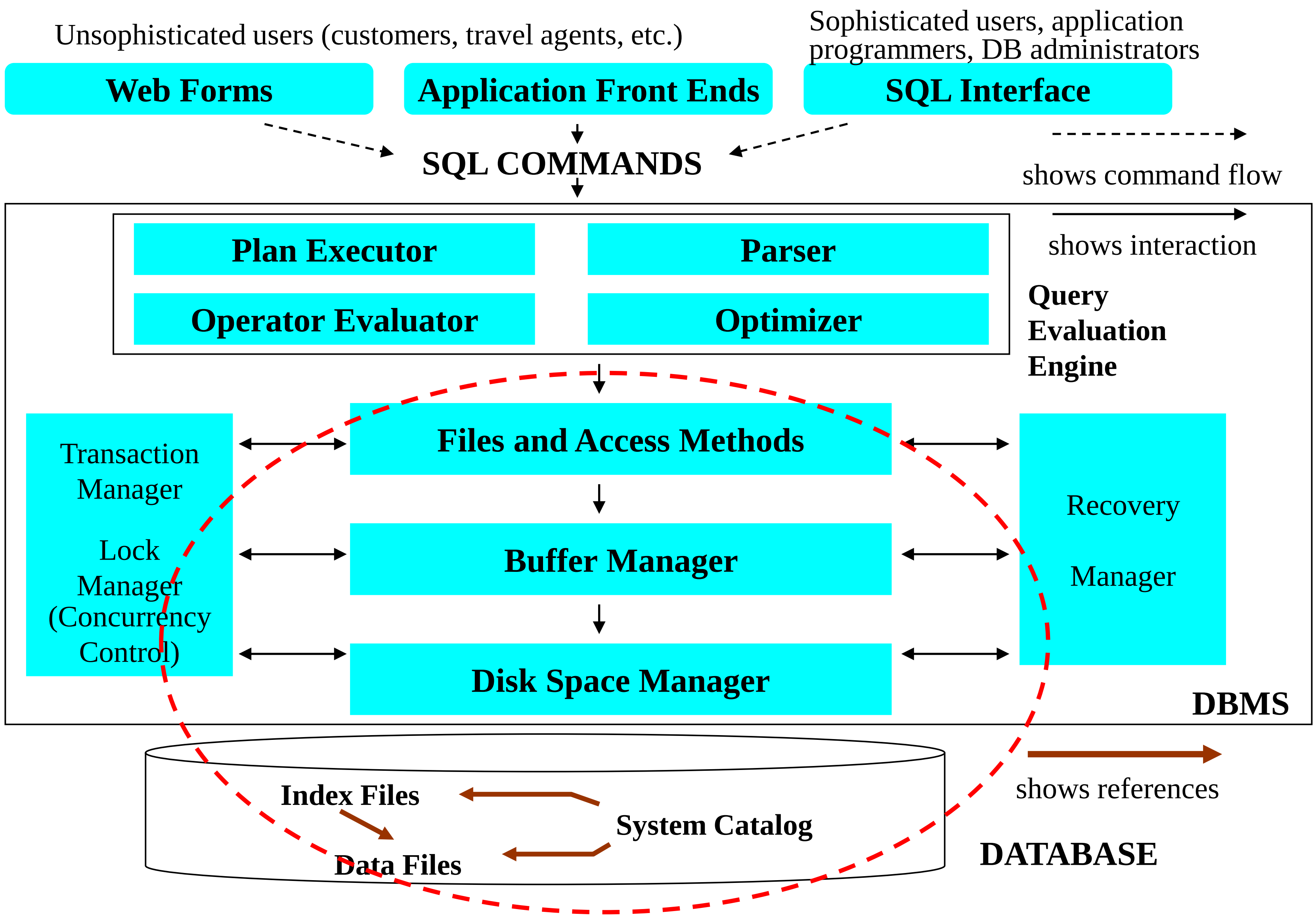
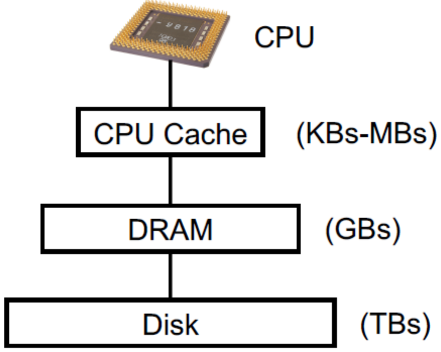


Figure 1.3 Anatomy of an RDBMS

Contents

9.0 Disks and Files



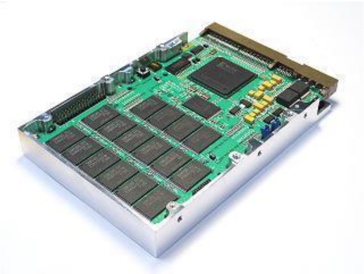
- DBMS stores information on harddisks or flash SSDs.
  - Electronic (CPU, DRAM) vs. Mechanical (harddisk) vs. Electronic (SSD)
- This has major implications for DBMS design!
  - READ: transfer data from disk to main memory (RAM).
  - WRITE: transfer data from RAM to disk.
  - Both are expensive operations, 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

A new challenger!

VS

60 years champion

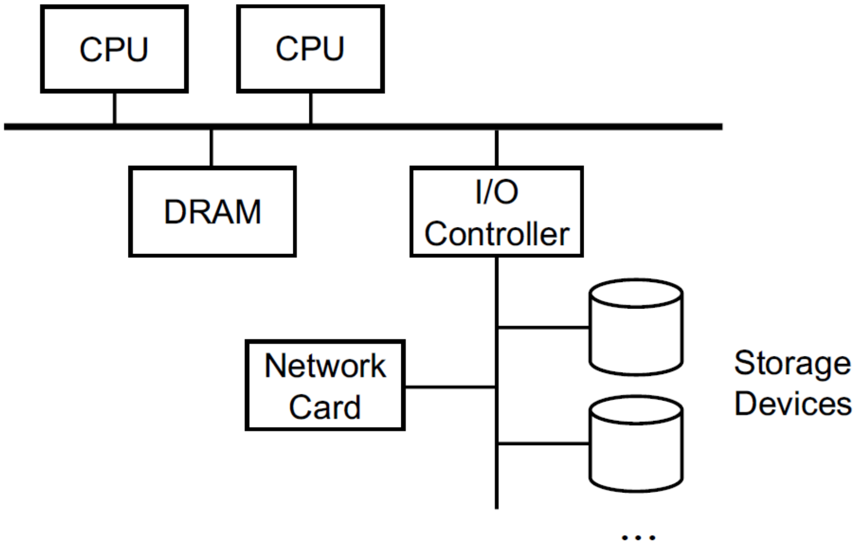


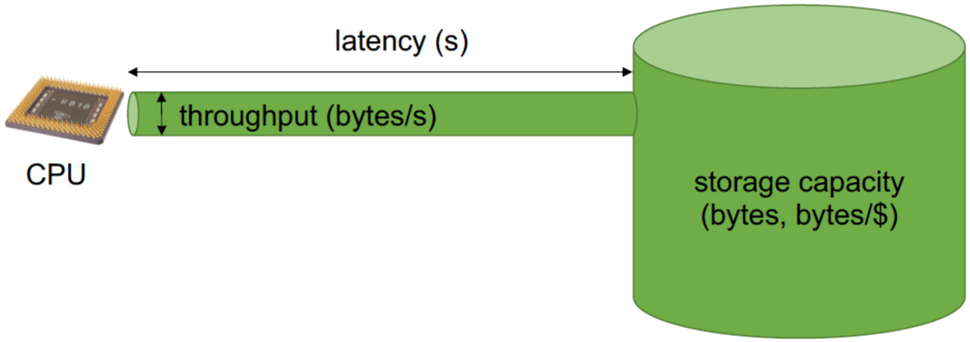
Identical Interface

$\equiv$

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

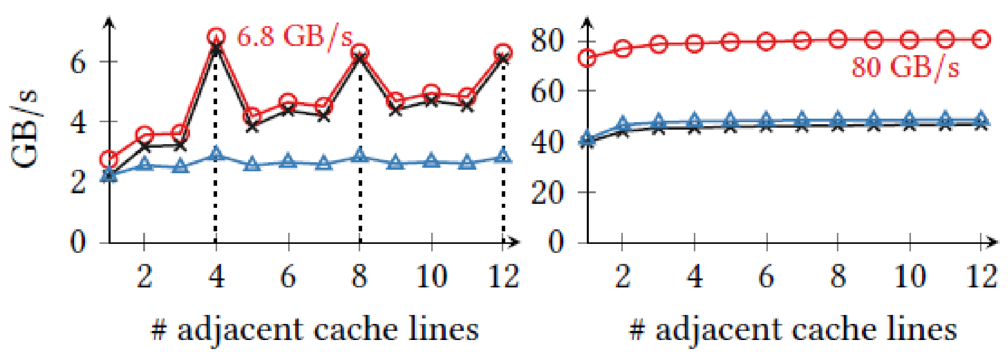
[Source: Rethinking Flash In the Data Center, IEEE Micro 2010]

Very Large Data Bases

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

(a) PMem write bandwidth (b) DRAM write bandwidth



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?

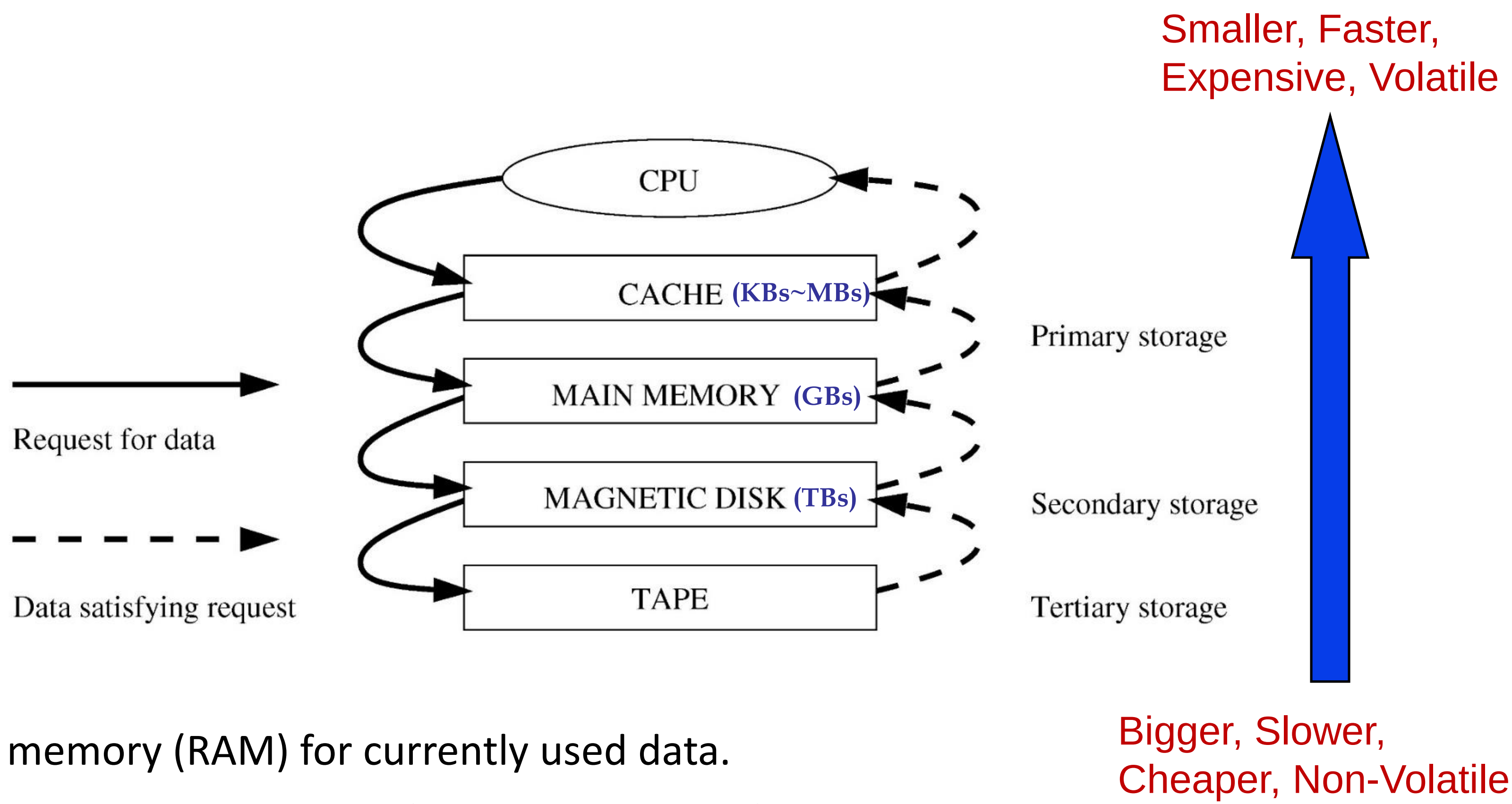
- Data-intensive ML algorithms. (source: Compressed Linear Algebra for Declarative Large-Scale Machine Learning)
  - Many ML algorithms are iterative, with repeated read-only data access. These algorithms often rely on matrix-vector multiplications, which require one complete scan of the matrix with 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

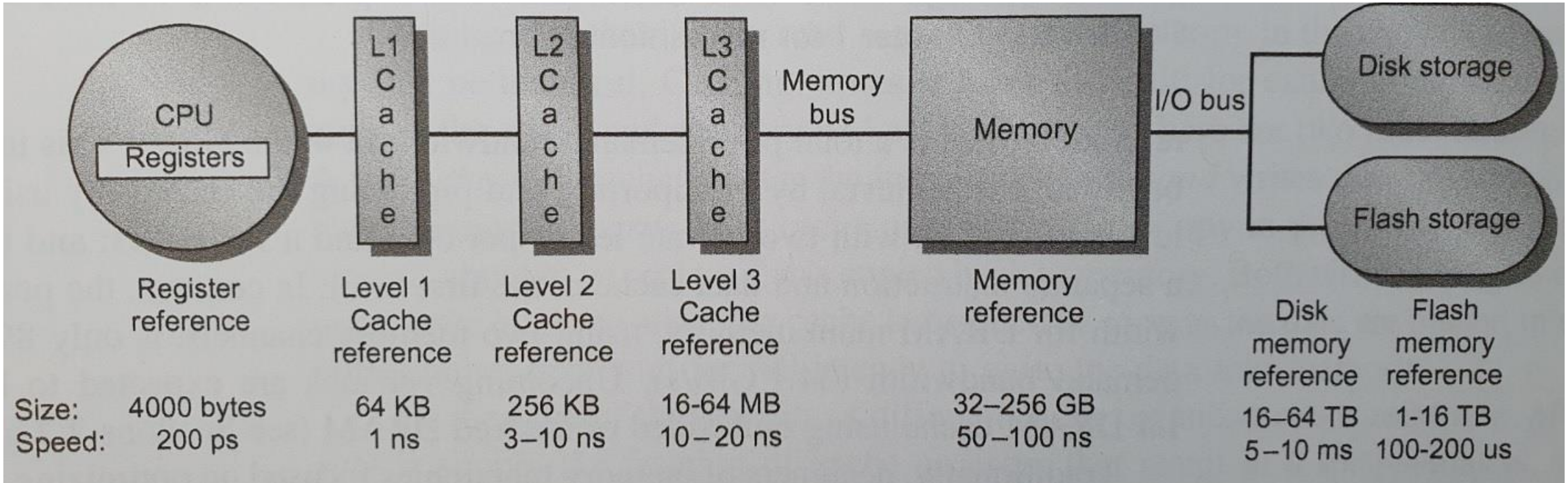
- Object storage is good for scalability for big data (https://blog.storagecraft.com/object-storage-systems/)
  - Scalability is where object-based storage does its most impressive work. Scaling out an object architecture is as simple as adding additional nodes to the storage cluster. Every server has 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

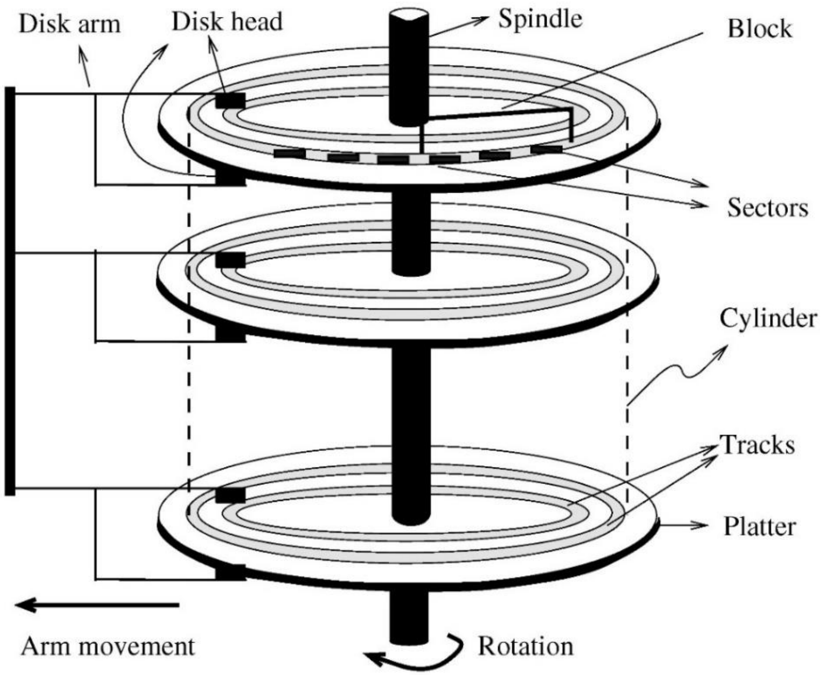
Source: Figure 1.2 in "Computer Architecture: A Quantitative Approach (6th Ed.)"

Disks

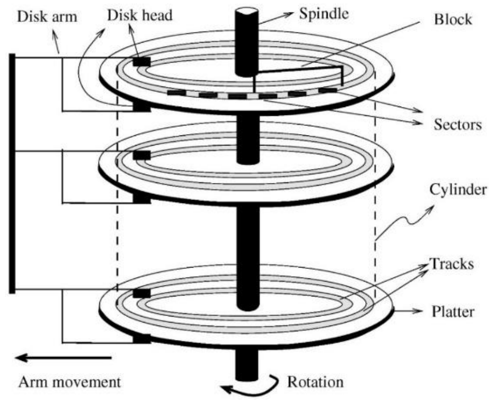

- Secondary storage device of choice. (non-volatile, durable)
- Main advantage over tapes: random access vs. sequential.
  - E.g. To find a record (with its address known) among 1 billion records:
- Data is stored and retrieved in disk blocks or pages unit.
- Unlike RAM, time to retrieve a disk page 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

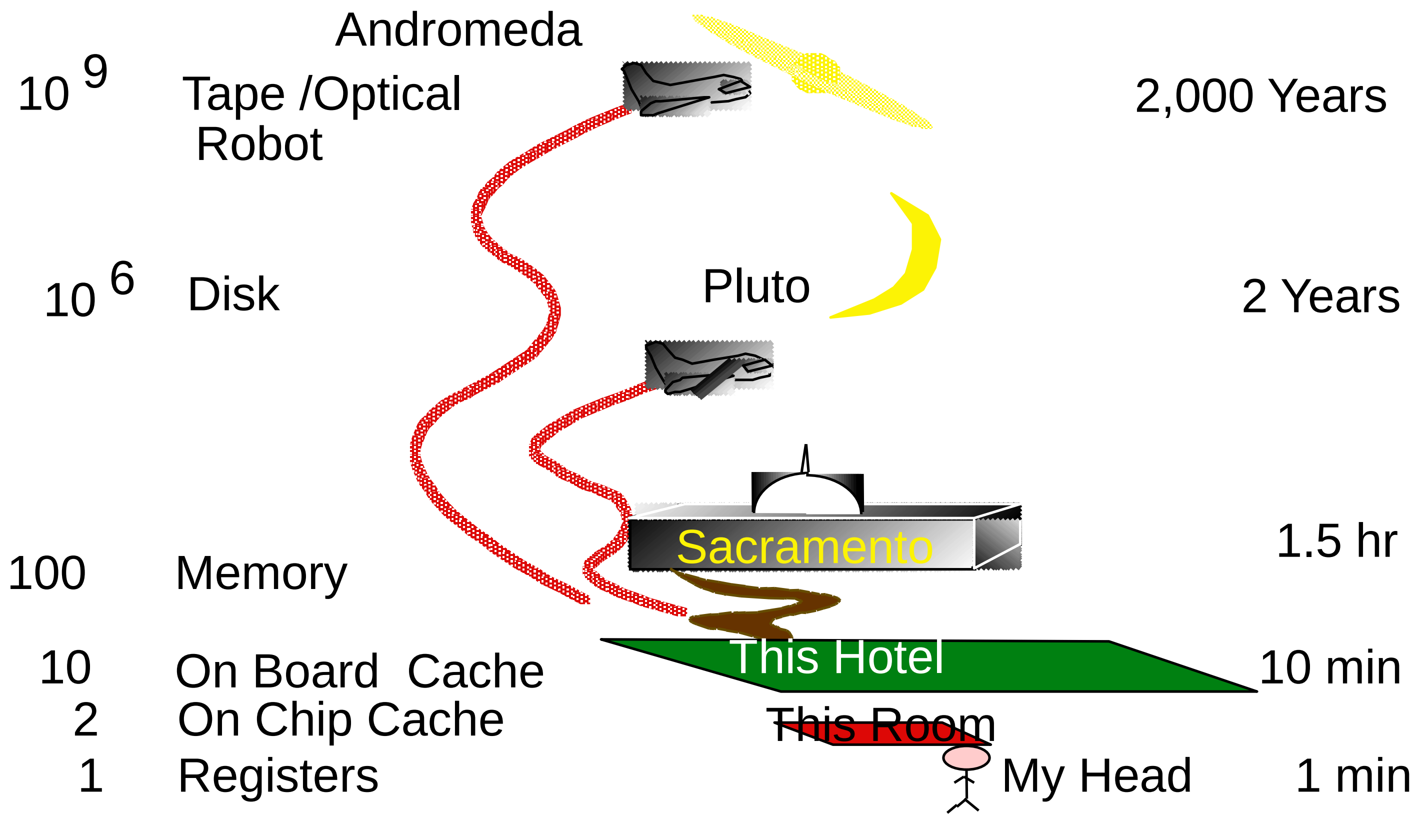
Arranging Pages on Disk

- `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

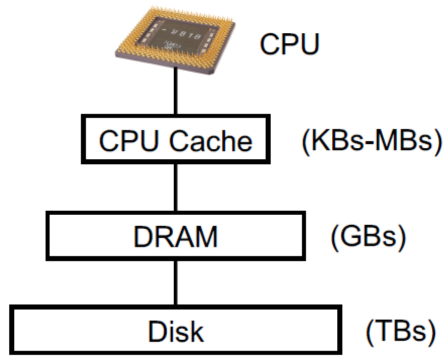
Assuming ~1GB/sec SSD

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

Very Large Data Bases

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**

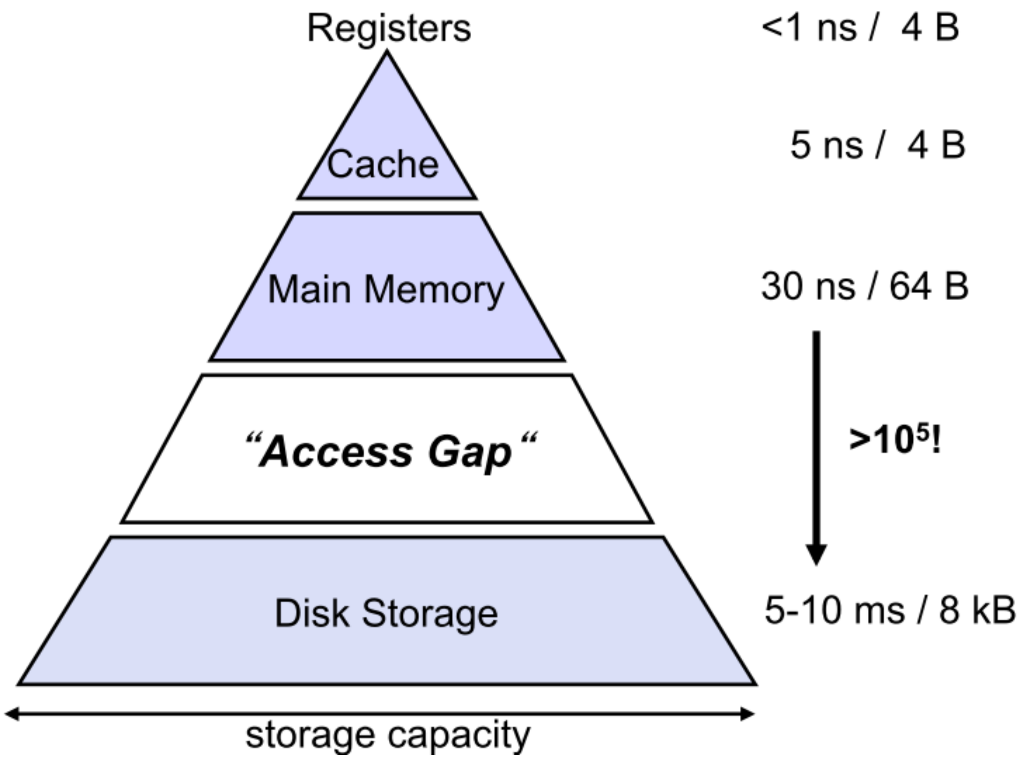
Metric	DRAM				HDD				SATA Flash SSD	
	1987	1997	2007	2017	1987	1997	2007	2017	2007	2017
Unit price($)	5k	15k	48	80	30k	2k	80	49	1k	560
Unit capacity	1MB	1GB	1GB	16GB	180MB	9GB	250GB	2TB	32GB	800GB
$/MB	5k	14.6	0.05	0.005	83.33	0.22	0.0003	0.00002	0.03	0.0007
Random IOPS	-	-	-	-	15	64	83	200	6.2k	67k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

Very Large Data Bases

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

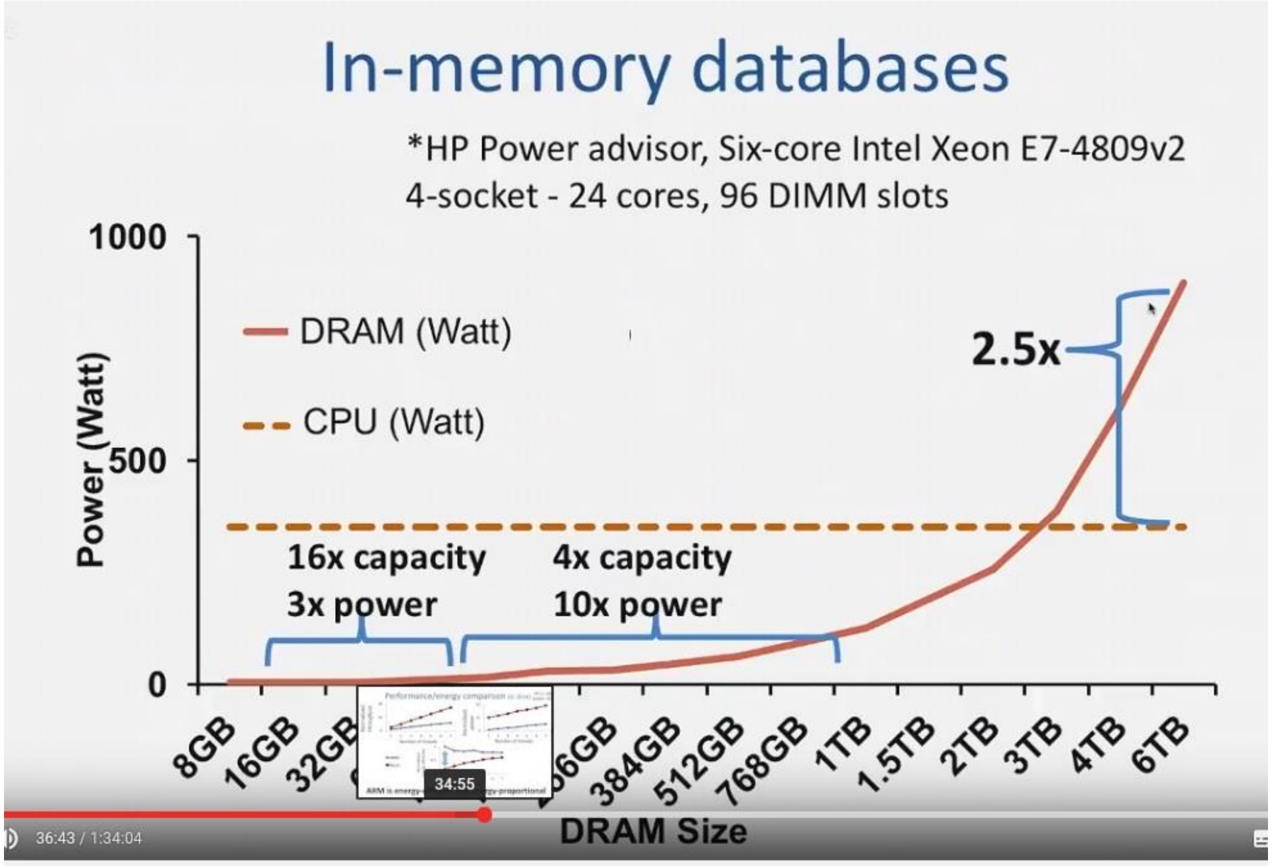
Why Not Store It All in Main Memory?

- 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

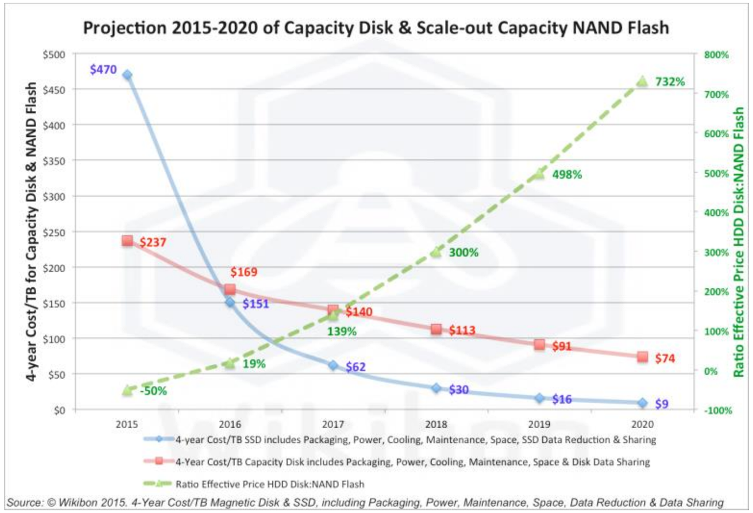
- Why MMDBMS has been recently popular since mid-2000s?
  - Sap Hana, MS Hekaton, Oracle In-memory, Altibase, ….
  - The price of DRAM had ever dropped for the last two decades
    - ✓ $/IOPS @ DISK >> $/GB @ DRAM
  - The overhead of disk-based DBMS is not negligible
  - Applications with extreme performance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

Power Consumption Issue in Big Memory



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

HDD vs. SSD [Patterson 2016]



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

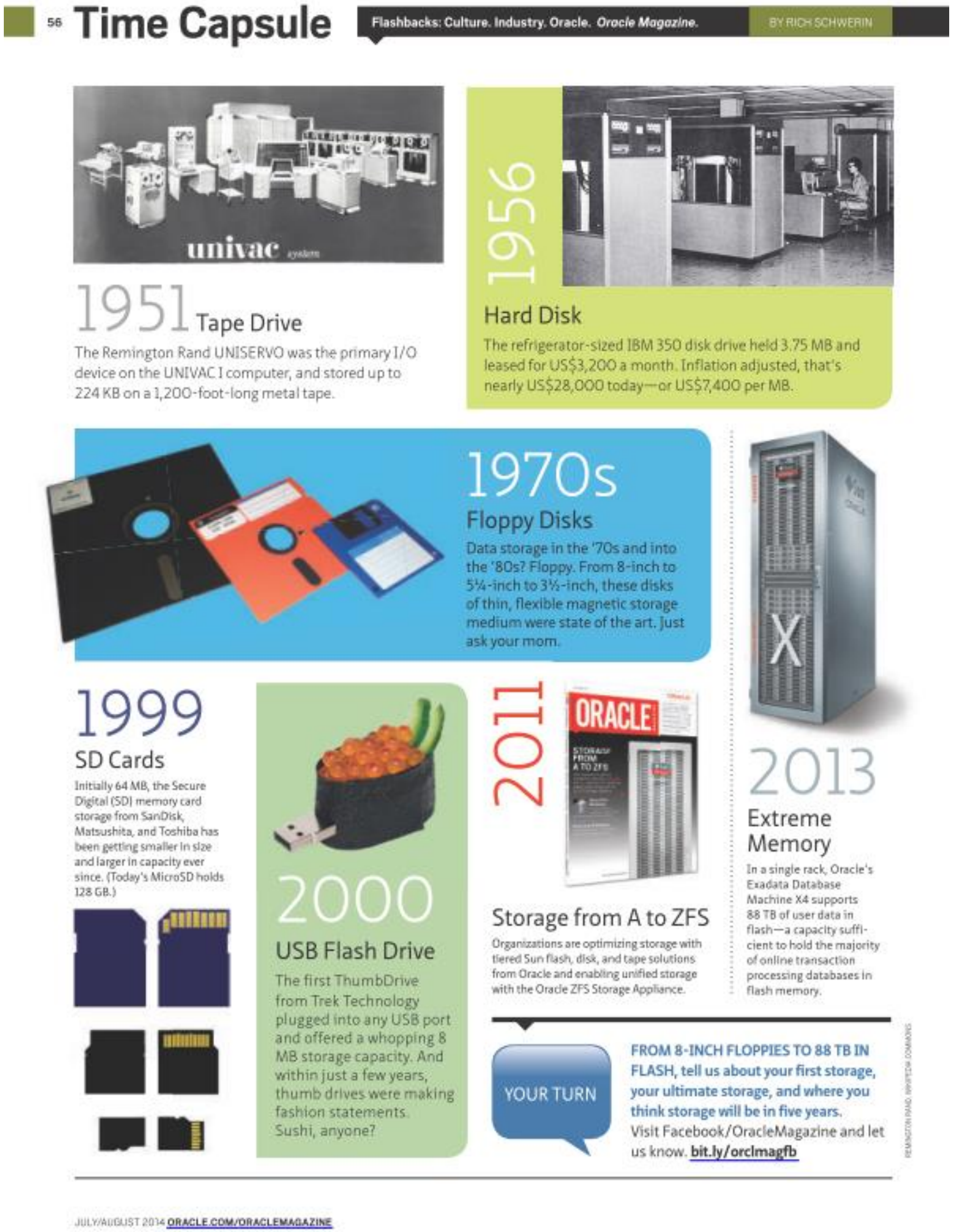
SSDs vs. HDDs

- 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

univac

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
ORACLE
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

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

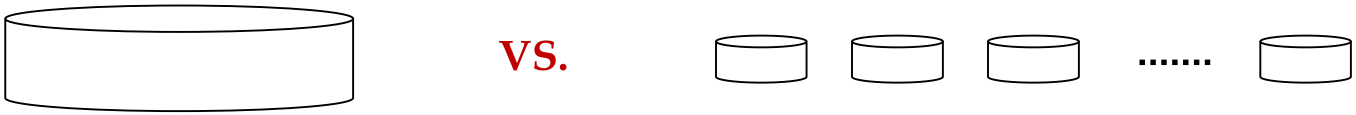
- CPU: Single-core → Multi-core (End of Moore's Law?)
- DRAM: small and expensive → large and become cheaper
- Storage: HDD → Flash SSD (→ NVRAM ?)
- Data center/Cloud: disaggregation, object storage

➔

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

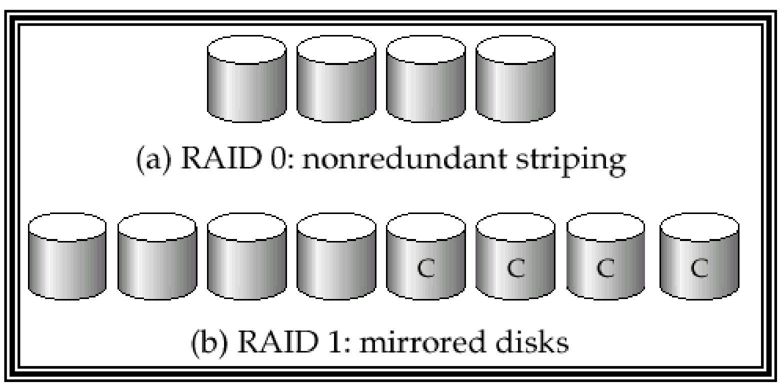
9.2 RAID

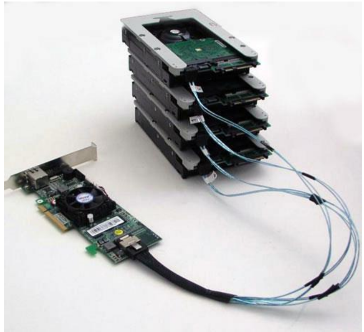
- SLED (Single Large Expensive Disk) approach till 1980s



- 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.

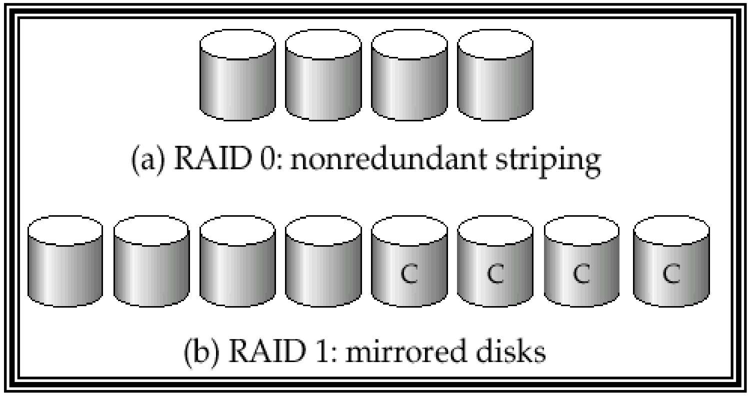




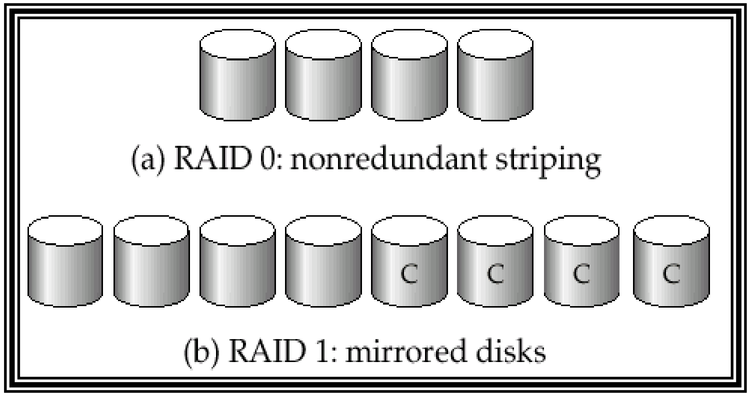
Cf. Tesla Battery and Rocket Tech.

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

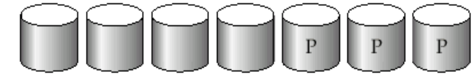


(a) RAID 0: nonredundant striping

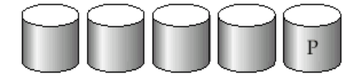
(b) RAID 1: mirrored disks

- 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.)

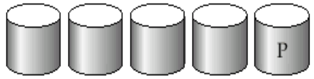


(c) RAID 2: memory-style error-correcting codes



(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.
- 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



(e) RAID 4: block-interleaved parity

(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