Database Architecture 2 & Storage

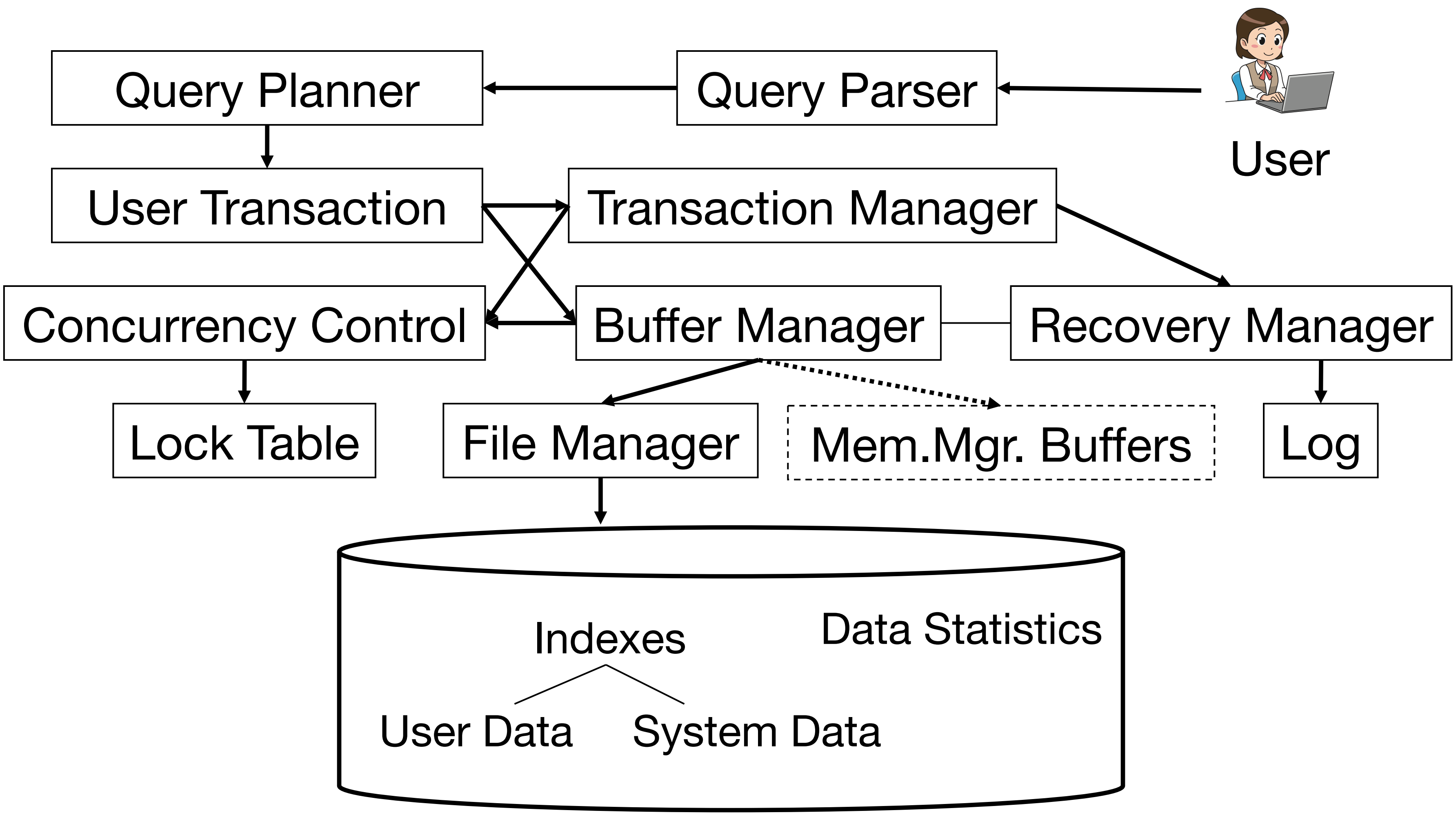
Instructor: Matei Zaharia

Outline

Alternative RDMBS architectures

Storage hardware

Typical RDBMS Architecture



Differentiating by Workload

2 big classes of commercial RDBMS today

Transactional DBMS: focus on concurrent, small, low-latency transactions (e.g. MySQL, Postgres, Oracle, DB2) → real-time apps

Analytical DBMS: focus on large, parallel but mostly read-only analytics (e.g. Teradata, Redshift, Vertica) → "data warehouses"

How To Design Components for Transactional vs Analytical DBMS?

Component	Transactional DBMS	Analytical DBMS
Data storage	B-trees, row oriented storage	Column-oriented storage
Locking	Fine-grained, very optimized	Coarse-grained (few writes)
Recovery	Log data writes, minimize latency	Log queries

How Can We Change the DBMS Architecture?

Decouple Query Processing from Storage Management

Example: data lake systems (Hadoop, GFS, Athena)

MapReduce
Apache Giraph
Apache Spark
Apache Storm
presto
TensorFlow

Processing engines

Parquet
{JSON} JavaScript Object Notation
DELTA LAKE
HIVE

Open storage and metadata formats

GFS
hadoop HDFS
Amazon S3

Large-scale file systems or blob stores

Decouple Query Processing from Storage Management

Pros:

- » Can scale compute independently of storage (e.g. in public cloud)
- » Let different orgs develop different engines
- » Your data is open by default to new engines

Cons:

- » Harder to guarantee isolation, reliability, etc
- » Harder to co-optimize compute and storage
- » Can't optimize across multiple engines
- » Hard to manage if too many engines!

Change the Data Model

Key-value stores: data is just key-value pairs, don't worry about record internals

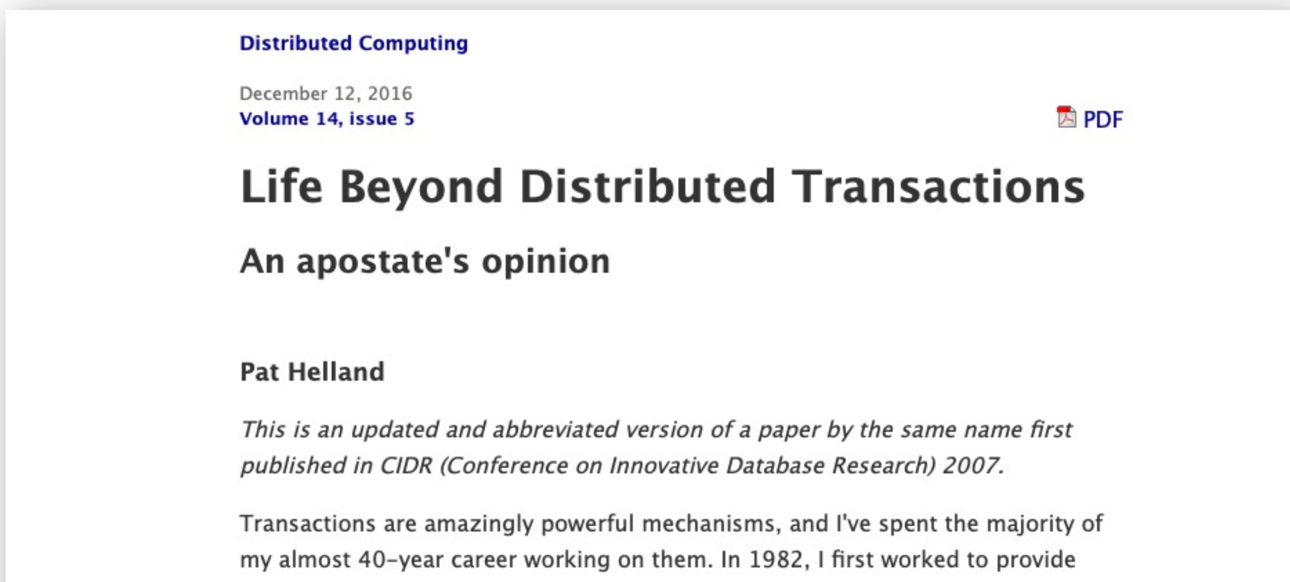
Message queues: data is only accessed in a specific FIFO order; limited operations

ML frameworks: data is tensors, models, etc

Change the Compute Model

Streaming: Apps run continuously; system manages updates, scaleup, recovery, etc

Eventual consistency: handle it at app level



Distributed Computing

December 12, 2016
Volume 14, issue 5

PDF

Life Beyond Distributed Transactions

An apostate's opinion

Pat Helland

This is an updated and abbreviated version of a paper by the same name first published in CIDR (Conference on Innovative Database Research) 2007.

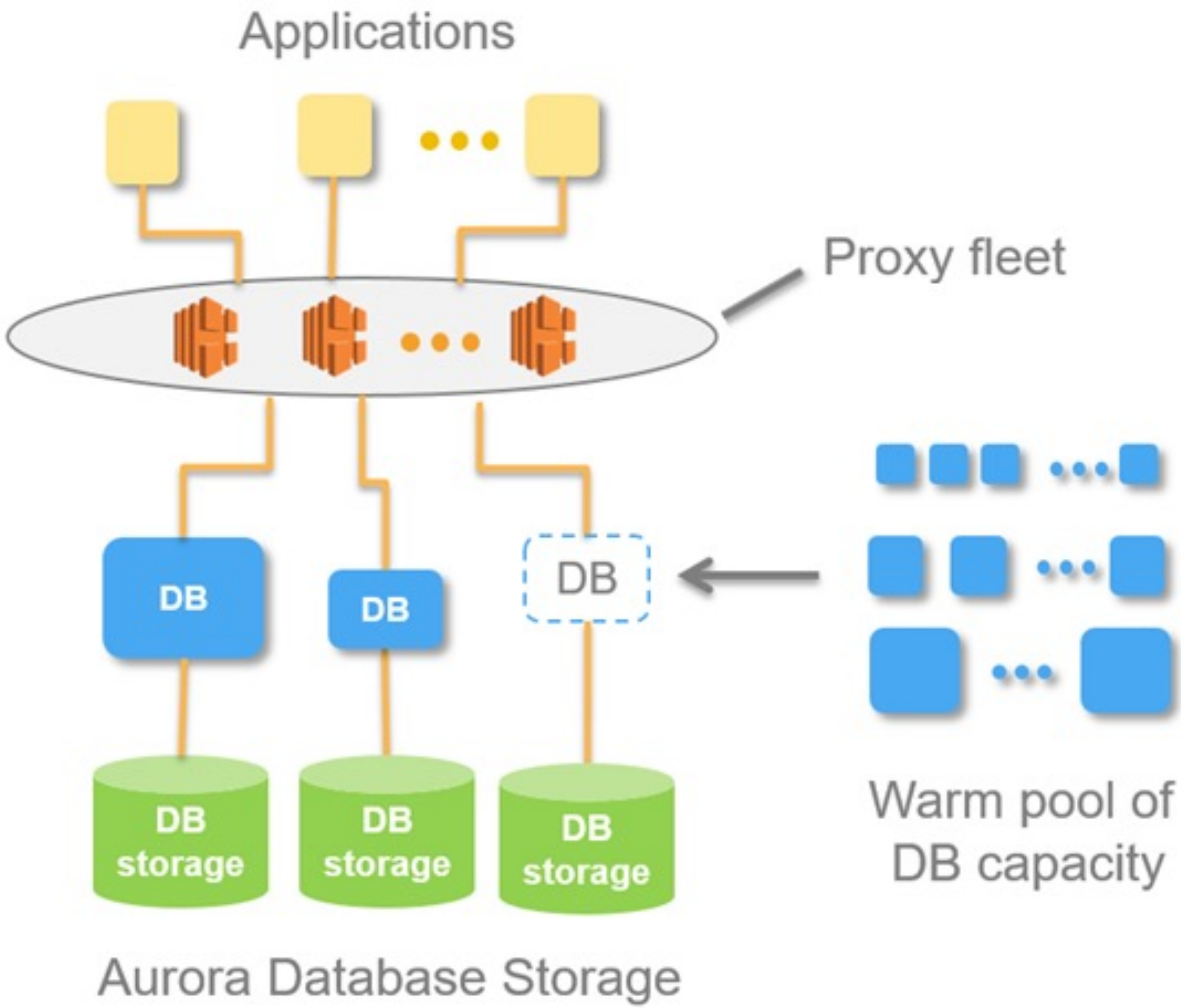
Transactions are amazingly powerful mechanisms, and I've spent the majority of my almost 40-year career working on them. In 1982, I first worked to provide

Different Hardware Setting

Distributed databases: need to distribute your lock manager, storage manager, etc

Public cloud: "serverless" databases that can scale compute independently of storage (e.g. AWS Aurora, Google BigQuery)

Example: AWS Aurora Serverless



Summary

All data systems face similar issues: API, performance, reliability, concurrency, etc

Relational DBMS offer one architecture that tackles many of these concerns together

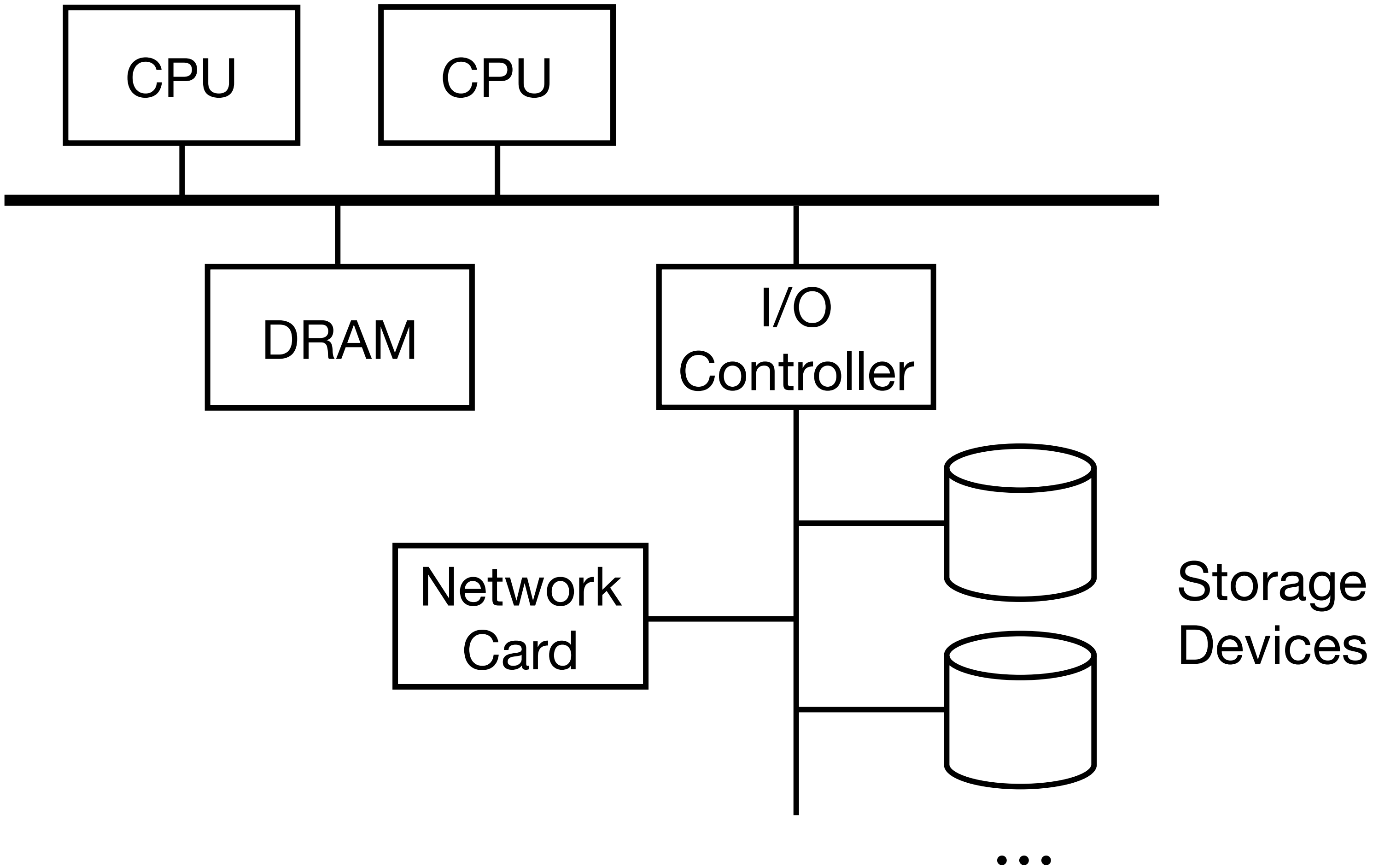
One trend is to **break apart** this monolithic architecture into specialized components

Outline

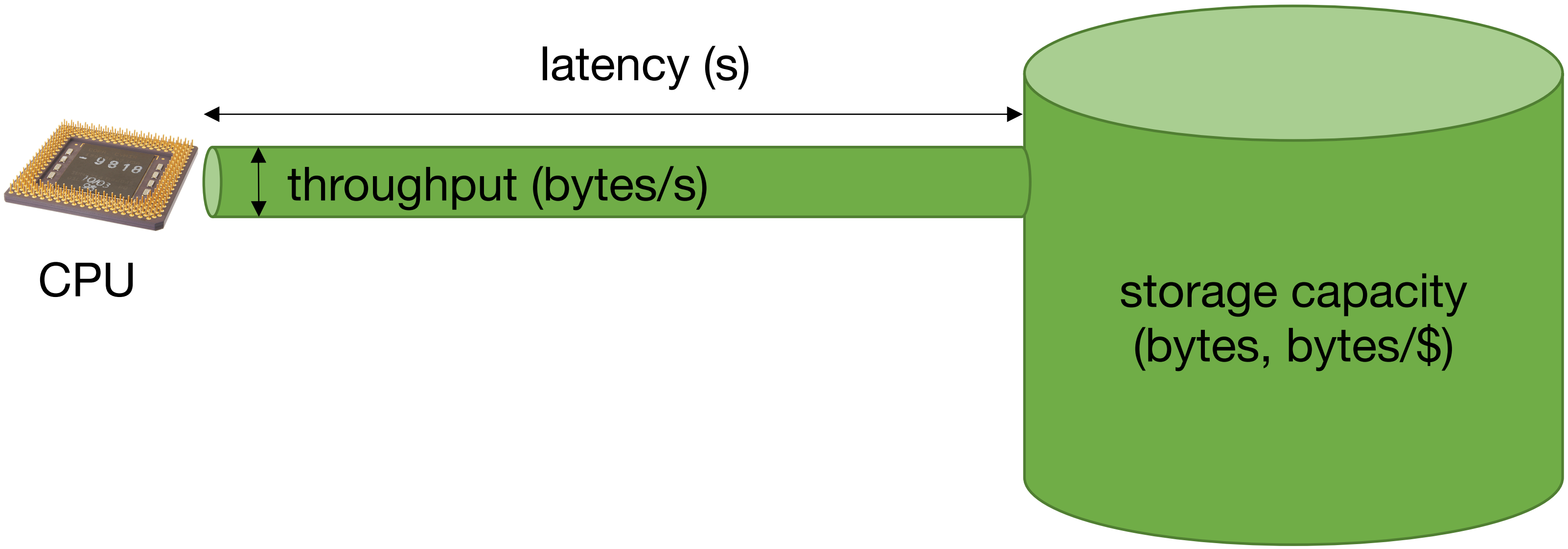
Alternative RDMBS architectures

Storage hardware

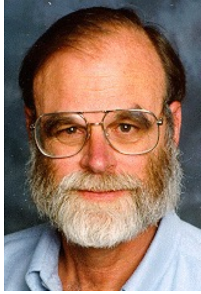
Typical Server



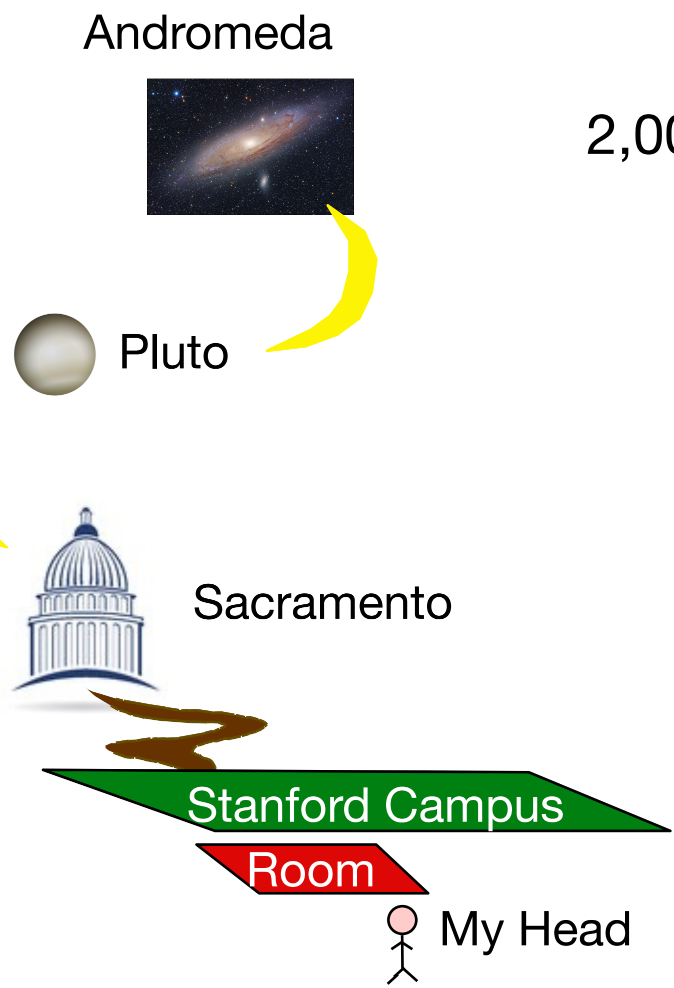
Storage Performance Metrics



Storage Latency



10^9	Tape /Optical Robot	2,000 Years
10^6	Disk	2 Years
150	Memory	2 hr
10	L2 Cache	10 min
2	L1 Cache	
1	Registers	1 min
ms		



Max Attainable Throughput

Varies significantly by device

» 100 GB/s for RAM

» 2 GB/s for NVMe SSD

» 130 MB/s for hard disk

Assumes large reads (≫1 block)!

Storage Cost

$1000 at NewEgg today buys:

» 0.3 TB of RAM

» 10 TB of NVMe SSD

» 50 TB of magnetic disk

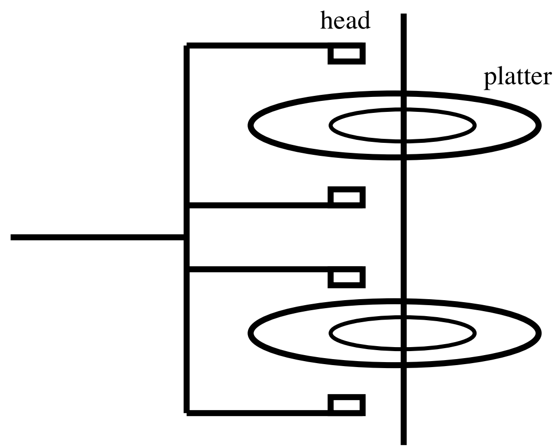
Hardware Trends over Time

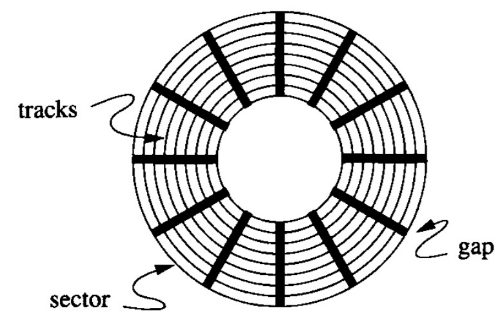
Capacity/$ grows exponentially at a fast rate (e.g. double every 2 years)

Throughput grows at a slower rate (e.g. 5% per year), but new interconnects help

Latency does not improve much over time

Most Common Permanent Storage: Hard Disks

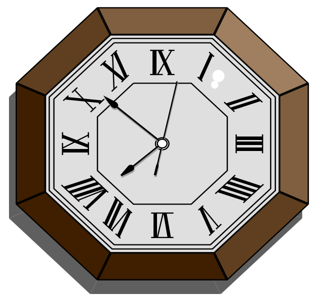




Terms: Platter, Head, Cylinder, Track, Sector (physical), Block (logical), Gap

Disk Access Time

I want
block X



?

block x
in memory

Disk Access Time

Time = Seek Time +
Rotational Delay +
Transfer Time +
Other

Typical Seek Time

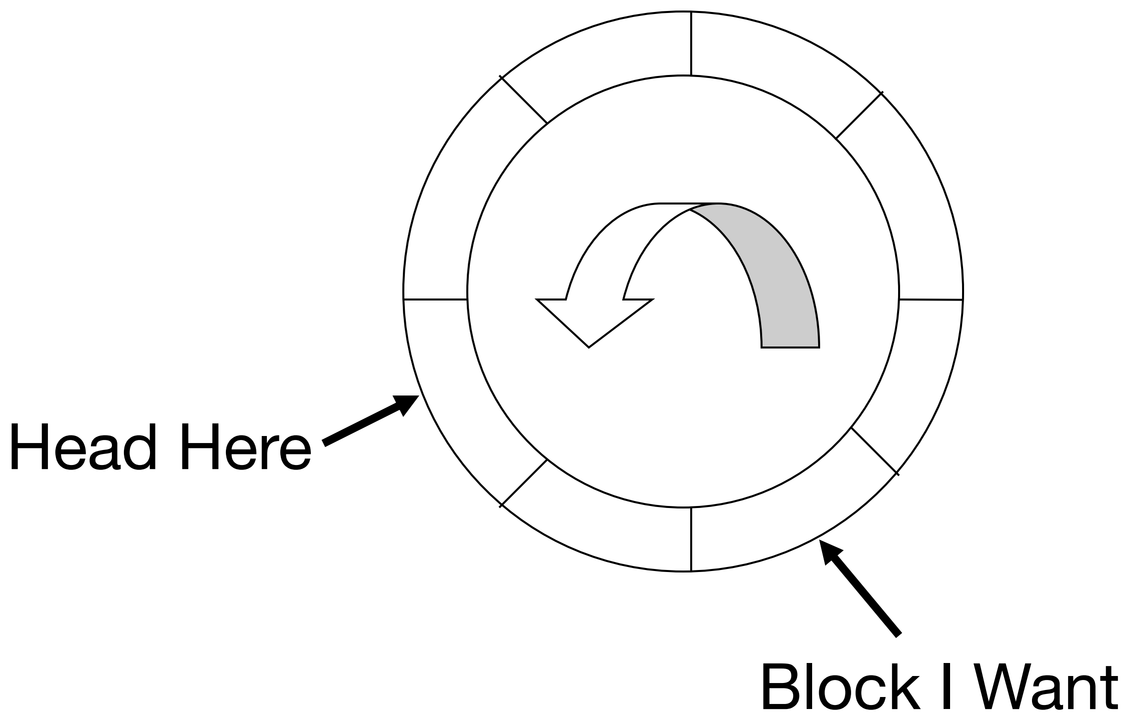
Ranges from

- » 4 ms for high end drives
- » 15 ms for mobile devices

In contrast, SSD access time ranges from

- » 0.02 ms: NVMe
- » 0.16 ms: SATA

Rotational Delay



Average Rotational Delay

R = 1/2 revolution

R=0 for SSDs

Typical HDD figures:

Rotation speed (rpm)	Average rotational latency (ms)
4,200	7.14
5,400	5.56
7,200	4.17
10,000	3.00
15,000	2.00

Transfer Rate

Transfer rate T is around 50-130 MB/s

Transfer time: size / T for contiguous read

Block size: usually 512-4096 bytes

So Far: Random Block Access

What about reading the “next” block?

If We Do Things Right (Double Buffer, etc)

Time to get = block size / T + negligible

Potential slowdowns:

» Skip gap

» Next track

» Discontinuous block placement

Sequential access generally much faster than random access

Cost of Writing: Similar to Reading

…. unless we want to verify!

need to add (full) rotation + block size / t

Cost To Modify a Block?

(a) Read Block

(b) Modify in Memory

(c) Write Block

[(d) Verify?]

Performance of DRAM

The same basic issues with “lookup time” vs throughput apply to DRAM

Min read from DRAM is a **cache line** (64 bytes)

64-byte random reads still slower than seq ones due to prefetching, page table, controllers, etc

Place co-accessed data together!

Example

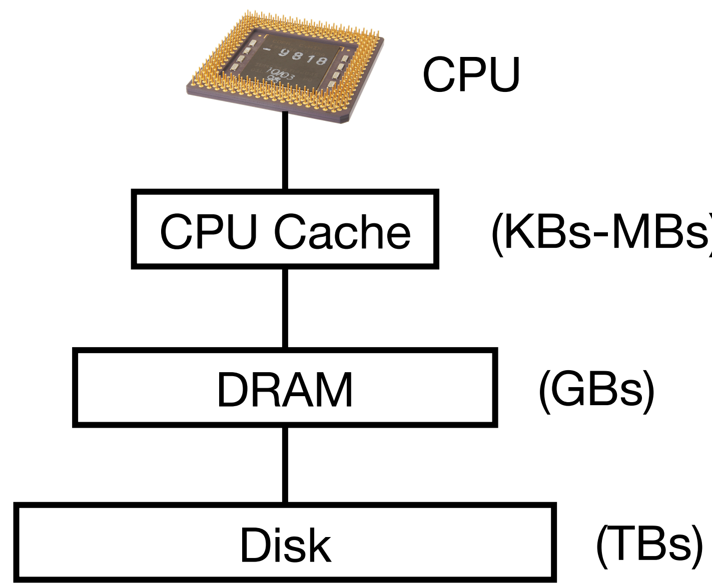
Suppose we're accessing 8-byte records in a DRAM with 64-byte cache line sizes

How much slower is random vs sequential?

In the random case, we are reading 64 bytes for every 8 bytes we need, so we expect to max out the throughput at least 8x sooner.

Storage Hierarchy

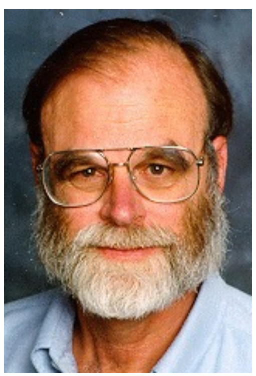
Typically want to **cache** frequently accessed data at a high level of the storage hierarchy to improve performance



Sizing Storage Tiers

How much high-tier storage should we have?

Can determine based on workload & cost



The 5 Minute Rule for Trading Memory Accesses for Disc Accesses
Jim Gray & Franco Putzolu
May 1985

The Five Minute Rule

Say a page is accessed every X seconds

Assume a disk costs D dollars and can do I operations/sec; cost of keeping this page on disk is

$$C_{disk} = C_{iop} / X = D / (I X)$$

Assume 1 MB of RAM costs M dollars and holds P pages; then the cost of keeping it in DRAM is:

$$C_{mem} = M / P$$

Five Minute Rule

This tells us that the page is worth caching when $C_{mem} < C_{disk}$, i.e.

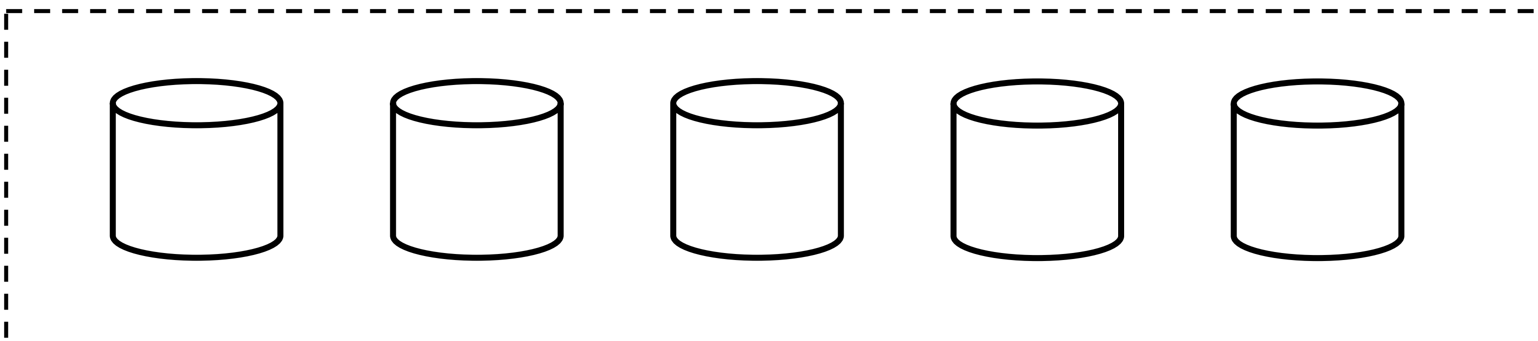
$$X < \frac{PagesPerMBofDRAM}{AccessesPerSecondPerDisk} \times \frac{PricePerDiskDrive}{PricePerMBofDRAM}$$

Tier	1987	1997	2007	2017
DRAM–HDD	5m	5m	1.5h	4h
DRAM–SSD	-	-	15m	7m (r) / 24m (w)
SSD–HDD	-	-	2.25h	1d

Source: The Five-minute Rule Thirty Years Later and its Impact on the Storage Hierarchy

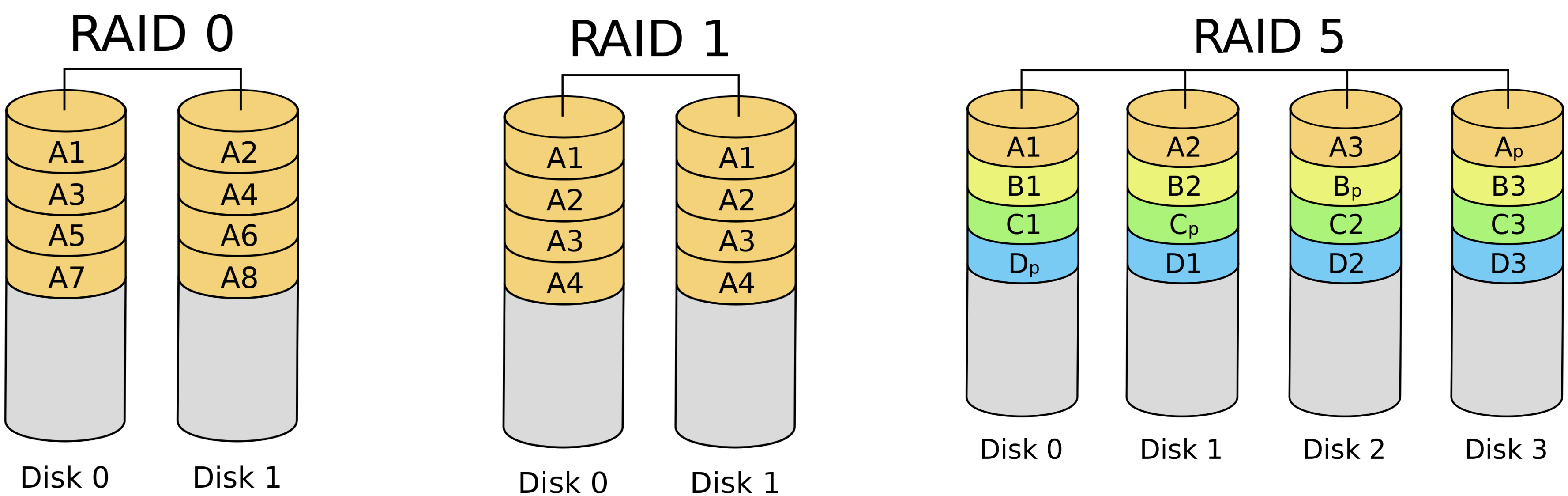
Combining Storage Devices

Many flavors of “RAID”: striping, mirroring, etc to increase **performance** and **reliability**



logically one disk

Common RAID Levels



Striping across 2 disks: adds performance but not reliability

Mirroring across 2 disks: adds reliability but not performance (except for reads)

Striping + 1 parity disk: adds performance and reliability at lower storage cost

Image source: Wikipedia

Handling Storage Failures

Detection: e.g., checksums

Correction: requires replicating data

Can be done at various levels:

» Single device (e.g., ECC RAM)

» Disk array

» OS

» Database system (e.g., logging)

Summary

Storage devices offer various tradeoffs in terms of latency, throughput and cost

In **all** cases, data layout and access pattern matter because random ≪ sequential access

Most systems will combine multiple devices

Assignment 1

Explores the effect of data layout for a simple in-memory database

» Fixed set of supported queries

» Implement a row store, column store, indexed store, and your own custom store!

Now posted on website!