COMP 421: Files & Databases

Lecture 0: Logistics, policies, scheduling, Project 0…
and maybe some actual material

WaitList

- I cannot make waitlist guarantees at this time
- To improve your chances of getting into the class (though not a guarantee), stay in the class and complete P0.
- Waitlist decisions will be made according to some fair metric, so please do not email us specific requests. We will not return these emails.

Course Overview

- This course is about the design/implementation of database management systems (DBMSs).
- This is **not** a course about how to use a DBMS to build applications or how to administer a DBMS.
  - See SILS, STOR, and more
  - You all have seen my memo at this point.

Course Pages

- Course Schedule, Syllabus, Policies, Projects: Course Web Page

Course Pages

- Project submission and autograding: Gradescope

Course Pages

- Mostly just links and grade/roster management: Canvas

Grading

- **Projects** (70%)
  - Project 0 (10%)
  - Project 1 (20%)
  - Project 2 (20%)
  - Project 3 (20%)

- **Midterm Exam** (10%)
- **Final Exam** (15%)

- **Participation** (5%)

Projects

- All projects will use the CMU DB Group BusTub academic DBMS.
  - Written in C++
  - Each project builds on the previous one.
- We will have at least two bootcamps for getting up to speed on C++
- More of these group sessions are possible if these first two are helpful

Late Policy

- You have a total of **four** late days to be used for the semester
  - Calendar days, not “business days”
- No late days for Project 0
- We will grant no-penalty extensions due to extreme circumstances (e.g., medical emergencies). These circumstances must be documented with the university. Do not email for an extension without attaching university documentation. We will not reply to emails without documentation.
  - If something comes up, please contact the instructors as soon as possible.

Office Hours

- Instructors and TAs will hold office hours on weekdays (Mon-Fri) at different times.
- Additional meetings can be made by appointment, or if there is overwhelming demand (e.g. project deadline)
- We need your help to schedule these, scheduling survey will arrive via email this week

<serious>

Office Hours: Do's and Don't's

- Do:
  - Come to office hours or make an appointment
  - Discuss material from the lectures/bootcamps/textbook
  - Discuss testing/debugging tools to help with projects
  - Discuss high-level issues or conceptual questions about the projects
- Don't:
  - Ask a TA or classmate to debug your code
  - Copy code that isn't yours (more on this…)

Reasonable Person Principle:

- Everyone will be reasonable.
- Everyone expects everyone else to be reasonable.
- No one is special.
- Do not be offended if someone suggests you are not being reasonable.

Reasonable people think about their needs, and the needs of others, and adjust their behavior to meet the goals of a common good for the community, i.e., expressing what you want to say, but accepting and accommodating the needs of others.

AI Policy

- Goal: A realistic, enforceable policy that acknowledges the reality of how humans program computers in 2025
- Caveat: still need to teach a course with gradable, right-sized assignments
- Policy: the limited use of AI is permitted, but students are wholly responsible for ensuring that their solutions adhere to the course **policy on plagiarism**
- Things that might be ok when working on project:
  - High level questions: “how does postgres store large objects?”
  - Limited coding assistance “include the library for `std::sort`”

Some Advice

- Don’t get in over your head with the AI generated code
  - Especially on Project 0

PLAGIARISM WARNING

- There is **no group collaboration on assignments or exams** in this course
- Exams are closed book (closed everything)
- You may **not** copy source code from other people or sources
- If you choose to use AI, you are responsible for ensuring you are submitting your own work
- From the syllabus: **"Students are responsible for ensuring that submitted work is not substantially similar to existing publicly or privately available materials"**
- Plagiarism is an honor code violation. This is your only warning.
  - Please ask me (not TAs!) if you are unsure.

One More Thing…

- We are using open source course materials, especially BusTub
- Make sure your Github fork of BusTub is **private**
  - (follow instructions on site/README)
- Making your solutions public (or distributing them in any way) is an honor code violation
- But also, its harmful to the CS Ed. Community
- And you'll have to answer to Andy Pavlo who wrote BusTub…

</serious>

Project 0

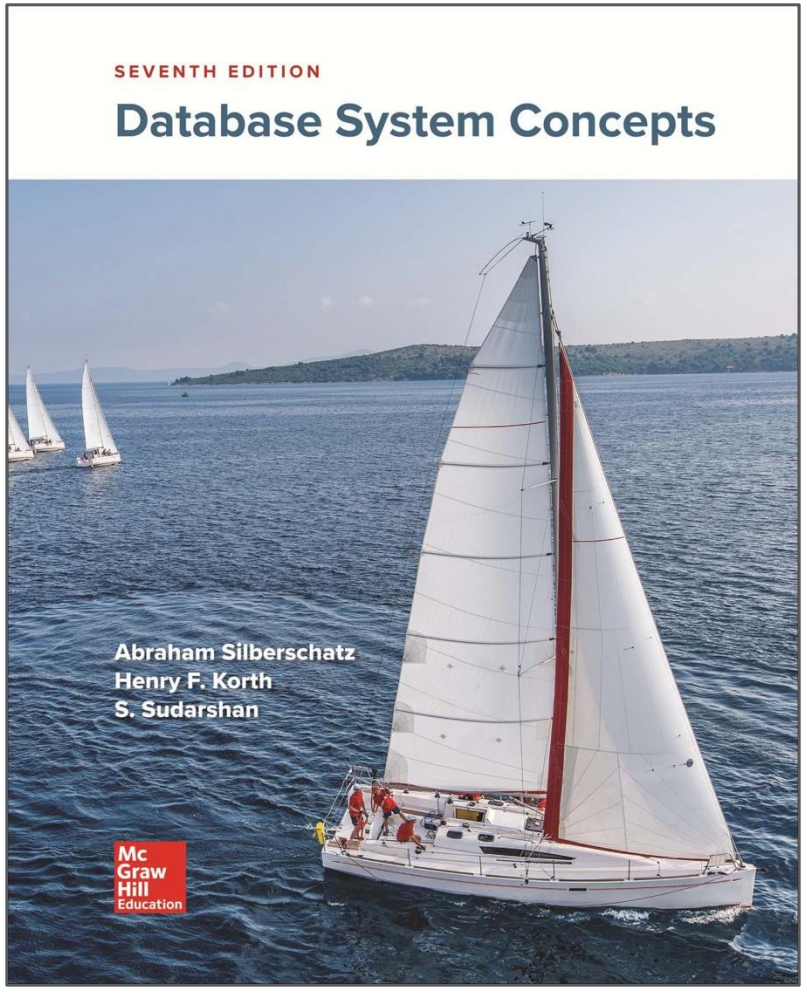
C++ Bootcamp

- Need to get up to speed quickly on C++. Three main approaches
  - **Reading.** Links on website to online tutorials, examples, books
  - **Recitation.** Instructor-led sessions to go through C++ features, examples, pitfalls. For students coming from Java / C.
  - **Writing.** Project 0 will give a tour of C++ and the dev environment for the semester. Future projects will be harder.

Textbook

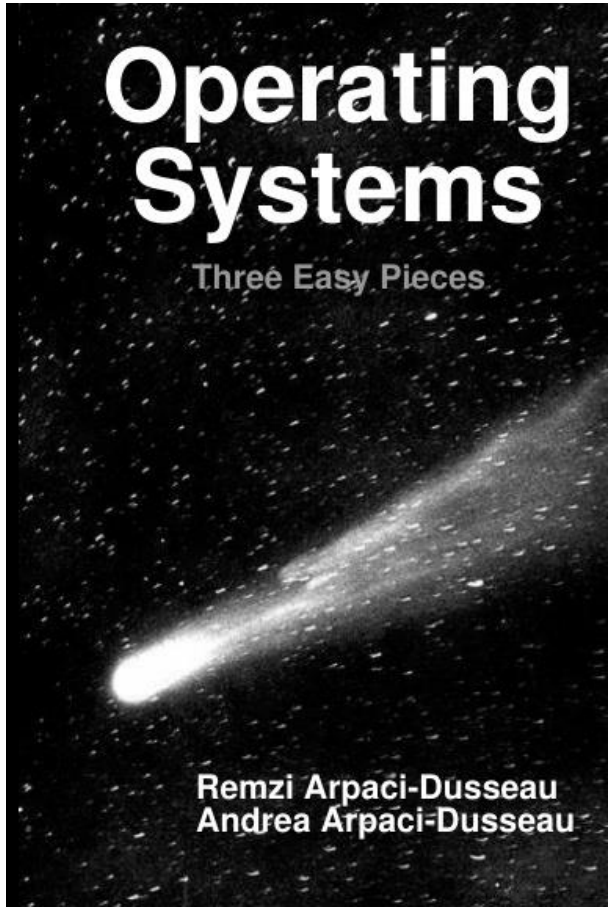
- **Database System Concepts**
  7th Edition
  Silberschatz, Korth, & Sudarshan

Official Course Textbook



Some other good books

- **Operating Systems: Three Easy Pieces (OSTEP),** (Arpaci-Dusseau)[2]
  - Free online at: https://pages.cs.wisc.edu/~remzi/OSTEP/
- A Tour of C++, Stroustrup
- Effective modern C++, Meyers
- Links on website



C++ Bootcamp

- Need to get up to speed quickly on C++. Three main approaches
  - **Reading.** Links on website to online tutorials, examples, books
  - **Recitation.** Instructor-led sessions to go through C++ features, examples, pitfalls. For students coming from Java / C.
  - **Writing.** Project 0 will give a tour of C++ and the dev environment for the semester. Future projects will be harder.

Bootcamp 1: Week of 8/25

Two identical 1 hour sessions (pick one):

8/25 @ 6:30 pm
SN014

8/26 @ 6:30 pm
SN014

Room might change!

We will provide food:

Pizza? Burritos? Other?

RSVP Now!

Bootcamp 2

- Will do a similar thing on more advanced C++ features
- Will coincide with release of Project 1
- Will try and incorporate feedback from Bootcamp 1 / Project 0

C++ Bootcamp

- Need to get up to speed quickly on C++. Three main approaches
  - **Reading.** Links on website to online tutorials, examples, books
  - **Recitation.** Instructor-led sessions to go through C++ features, examples, pitfalls. For students coming from Java / C.
  - **Writing.** Project 0 will give a tour of C++ and the dev environment for the semester. Future projects will be harder.

Project 0 (P0): Goals

- Get you started on C++, so you are not surprised later.
- Get you thinking about algorithms and (a bit) about concurrency.
- P0 is about building a HyperLogLog data structure.
  - Probabilistic data structure used for sketching in large-scale a distributed DMBSs
  - The data structure is cool, but is not really the point
- P0 is published; due 9/3/25 @ 11:59:59.
- **No late days allowed for P0.**

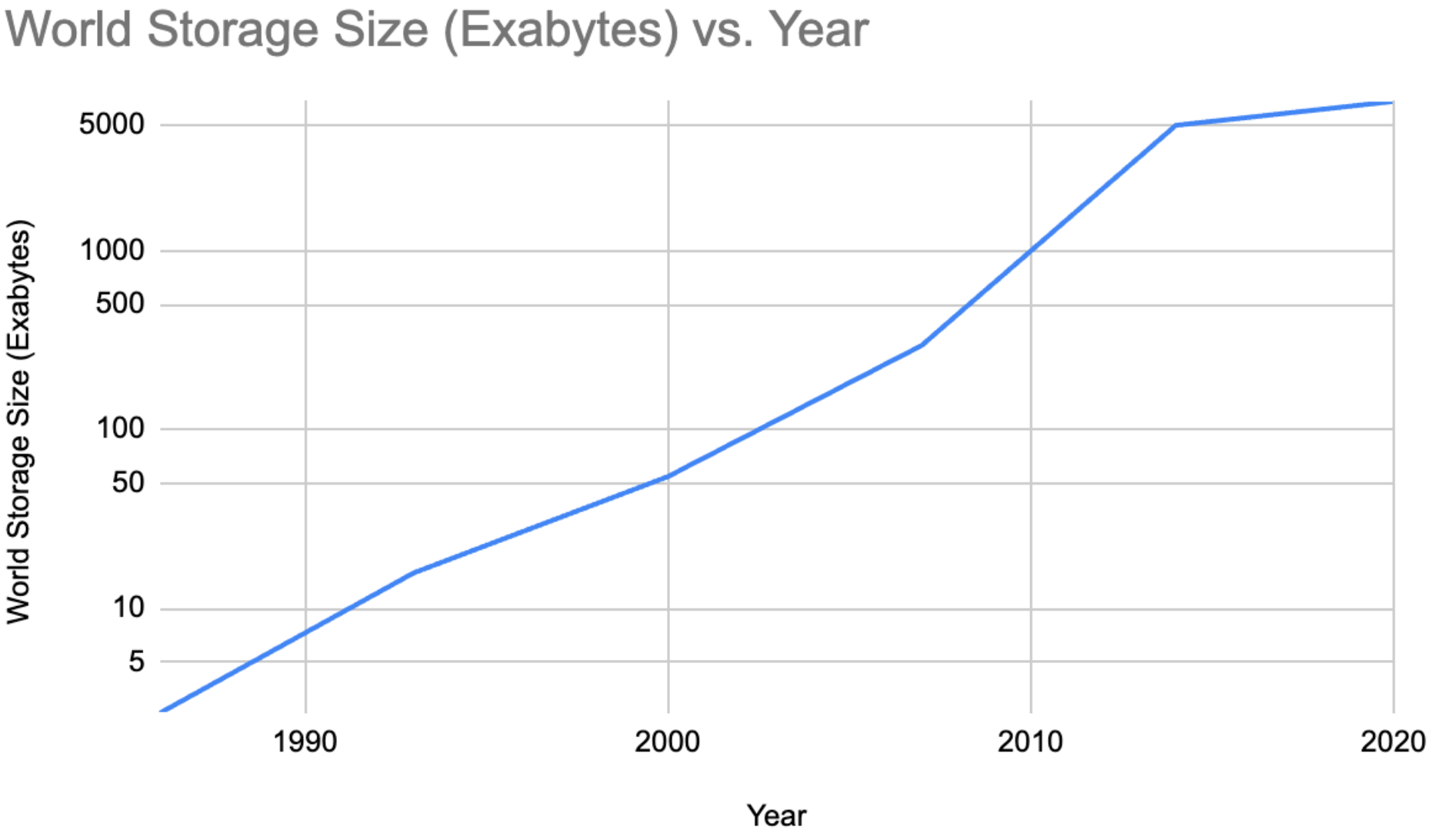
If you have a bad time on this project,
you're gonna have a bad time in this course

Project 0 (P0): Tips and Tricks

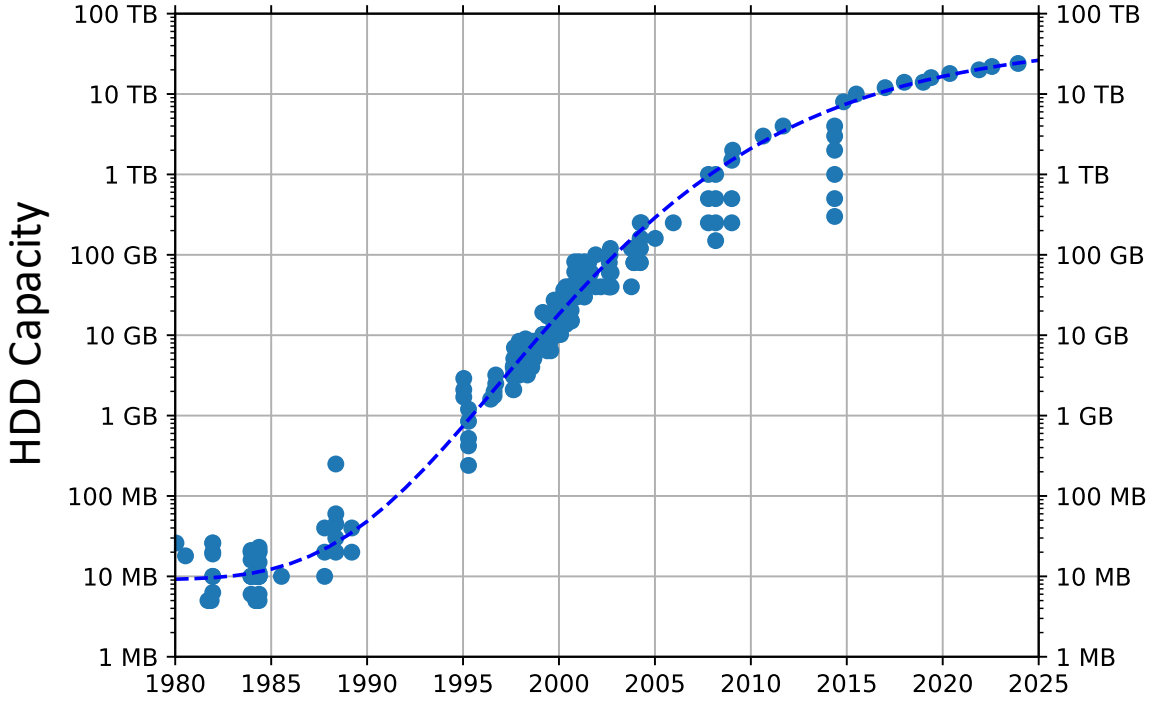
- Start now so you can get the most out of Bootcamp 1 and be ready when these projects get harder
- Follow instructions on either the P0 page or the BusTub github to set up your dev environment and build BusTub.
  - In general, we won't ask you to mess with the build system (cmake), but you should understand what the different options do
- Learn how to use the GTest testing framework used for testing BusTub.
  HINT: You will probably have to run the `hyperloglog_test` suite locally, and look into `test/primer/hyperloglog_test.cpp` to figure out how to pass all tests.
  - Future projects might not make all tests visible. Writing your own tests will become super helpful.

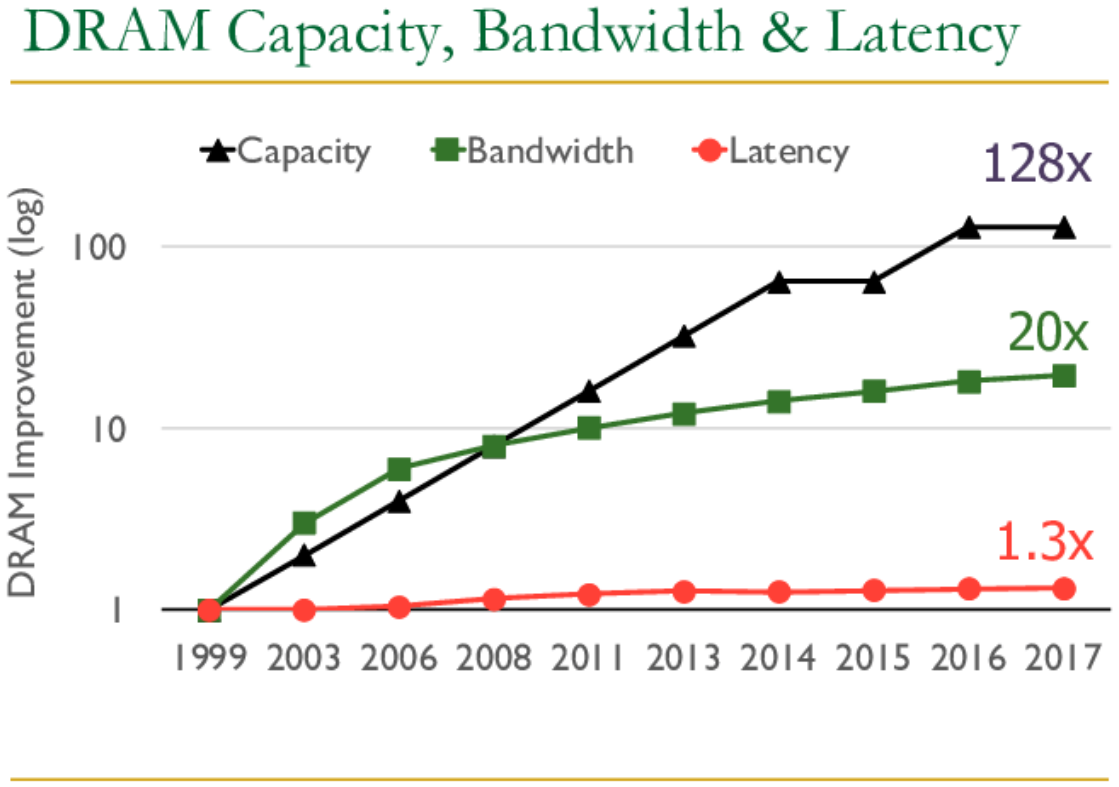
<course>

High-level Problem



High-level Problem





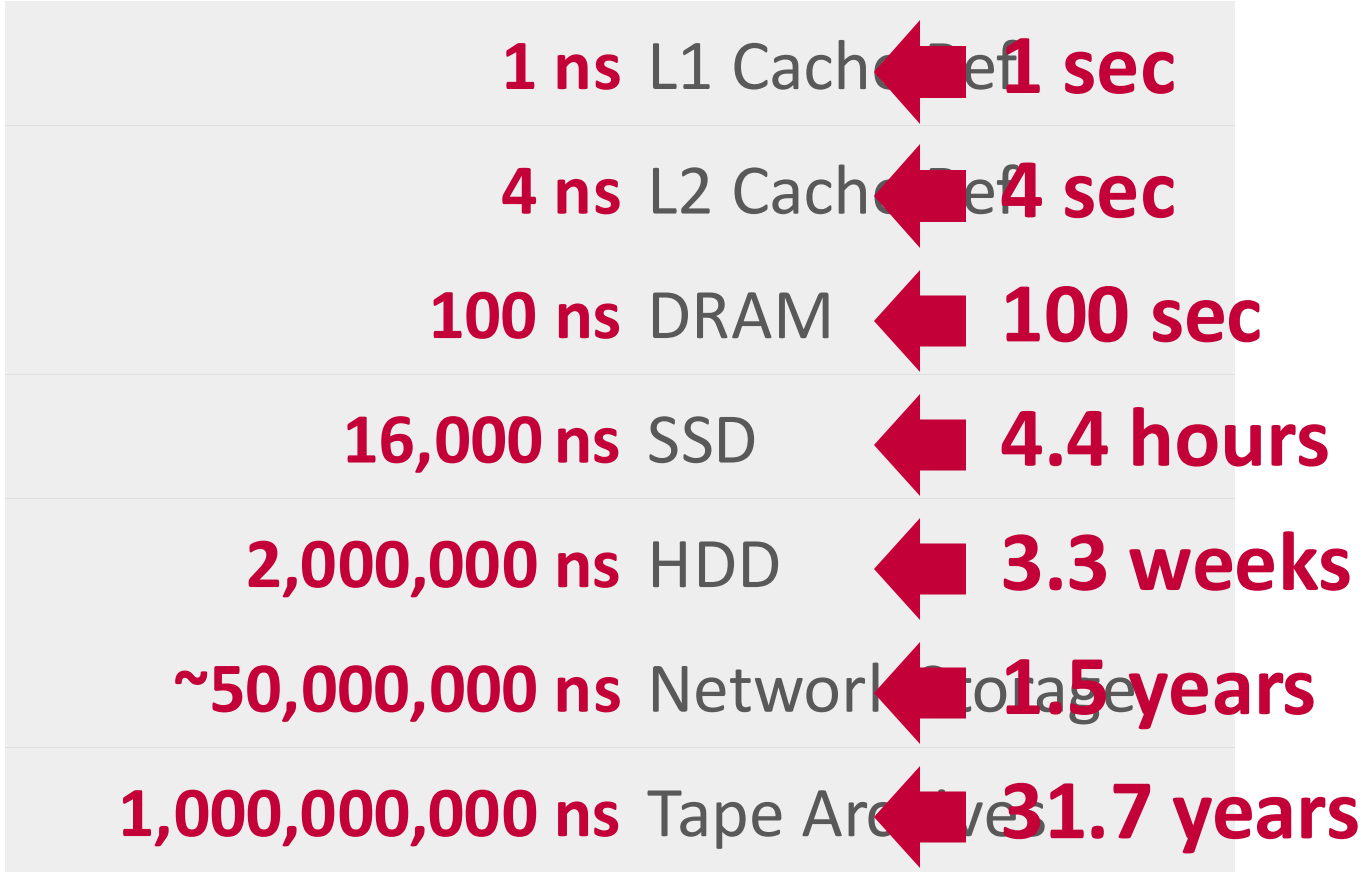
Memory/storage have grown

Storage capacity = ~100x DRAM remains common

Today: 10's – 100's GB DRAM : 1's -10's TB of SSD/HDD per server

Latency Numbers Every Programmer Should Know

Or: the whole story of COMP 421 in one table



Source: Colin Scott

Latency Numbers Every Programmer Should Know

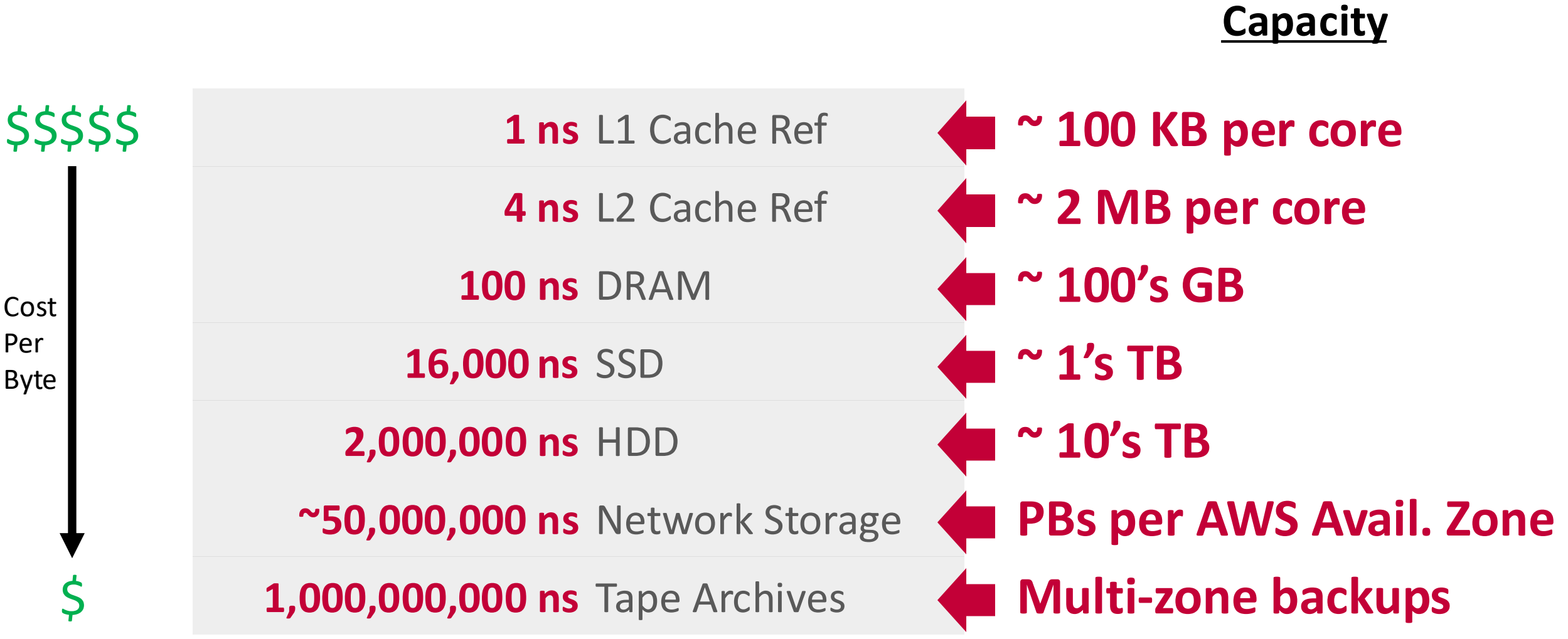
1 ns	L1 Cache Ref	Volatile
4 ns	L2 Cache Ref	
100 ns	DRAM	
16,000 ns	SSD	Non-volatile, single node
2,000,000 ns	HDD	
~50,000,000 ns	Network Storage	Non-volatile, off-node
1,000,000,000 ns	Tape Archives	

Source: Colin Scott

The Database Dilemma

- Want persistent storage and durable updates (storage), but also fast, complex data processing (memory)

Latency Numbers Every Programmer Should Know



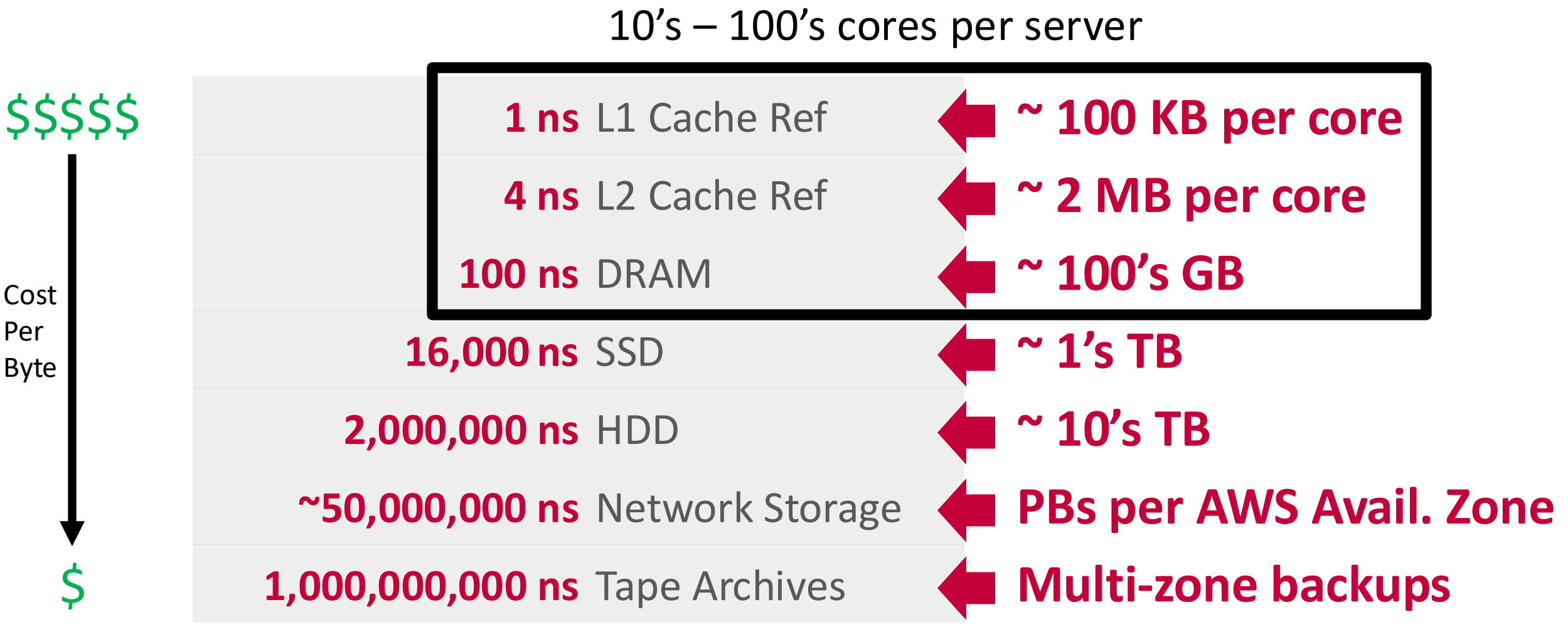
Source: Colin Scott

The Database Dilemma

- Want data persistence and durable updates (storage), but also fast/complex data processing (memory)

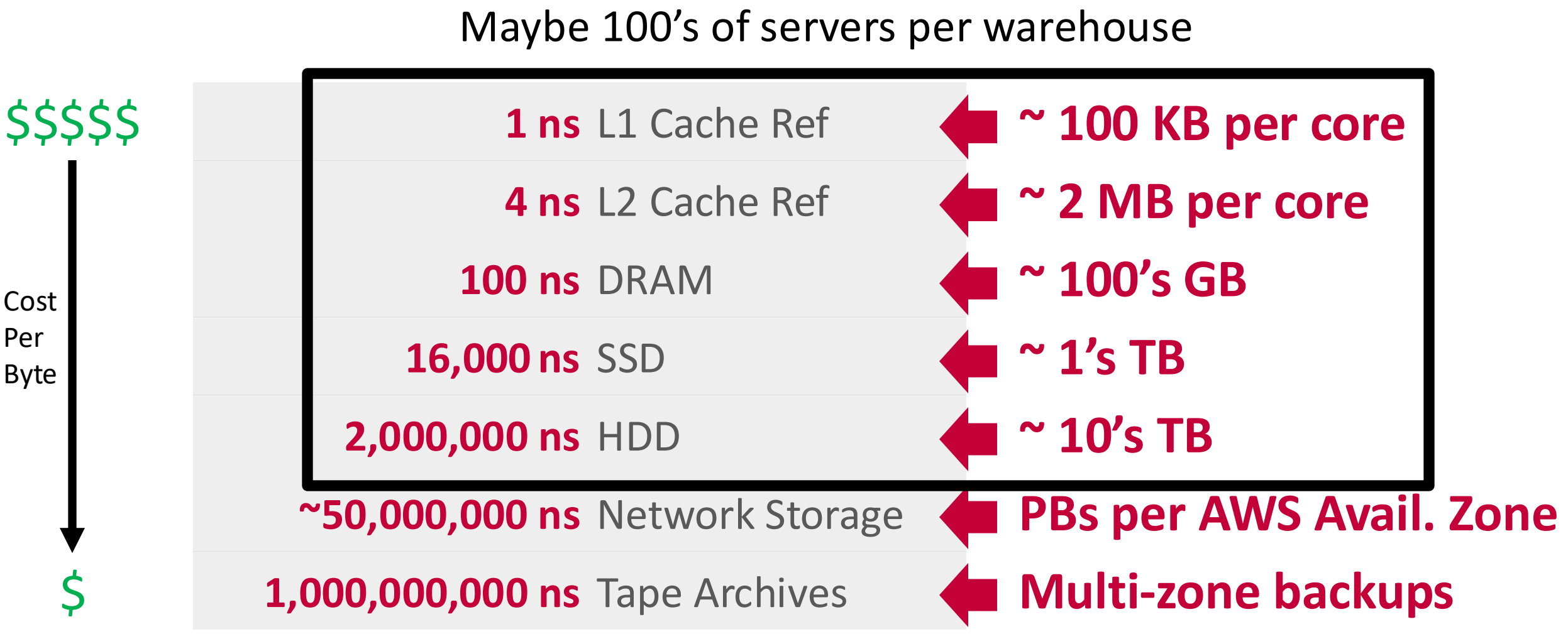
- Want to process a drive worth of data (e.g. TB) using a server's worth of memory (e.g. 64 GB)

Latency Numbers Every Programmer Should Know



Source: Colin Scott

Latency Numbers Every Programmer Should Know



Source: Colin Scott

The Database Dilemma

- Want data persistence and durable updates (storage), but also fast/complex data processing (memory)
- Want to process a drive worth of data (e.g. TB) using a server's worth of memory (e.g. 64 GB)
- To leverage available compute, need concurrency at every level: intraquery, interquery, single-node, distributed database (multi-node)
- Want abstractions to hide hardware/system design from users

Latency Numbers Every Programmer Should Know

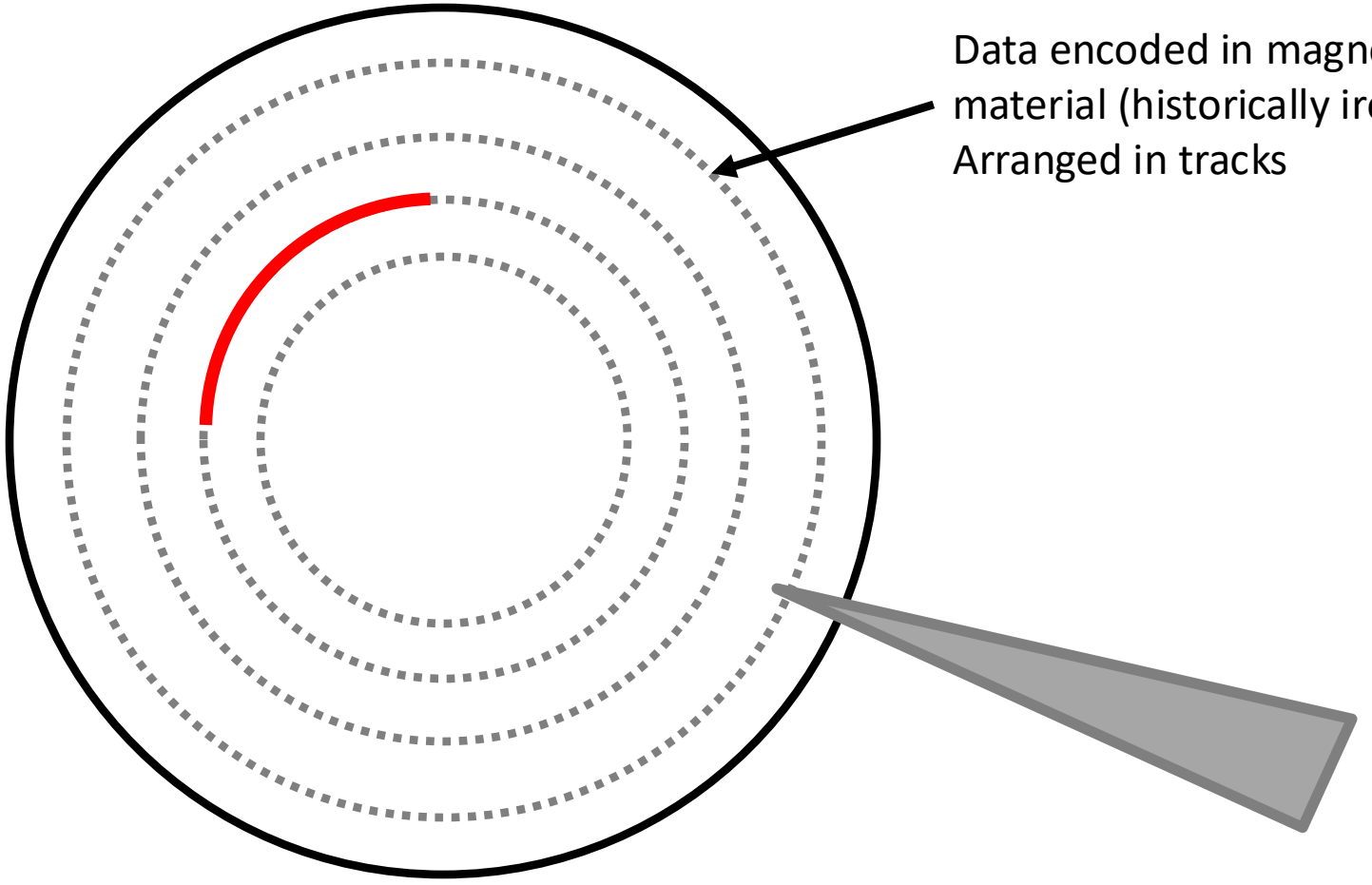
1 ns	L1 Cache Ref
4 ns	L2 Cache Ref
100 ns	DRAM
16,000 ns	SSD
2,000,000 ns	HDD
~50,000,000 ns	Network Storage
1,000,000,000 ns	Tape Archives

Example: disk performance under different workloads

Source: Colin Scott

Spinning Rust

Platter, spinning @ 15k RPM



Data encoded in magnetic material (historically iron oxide) Arranged in tracks

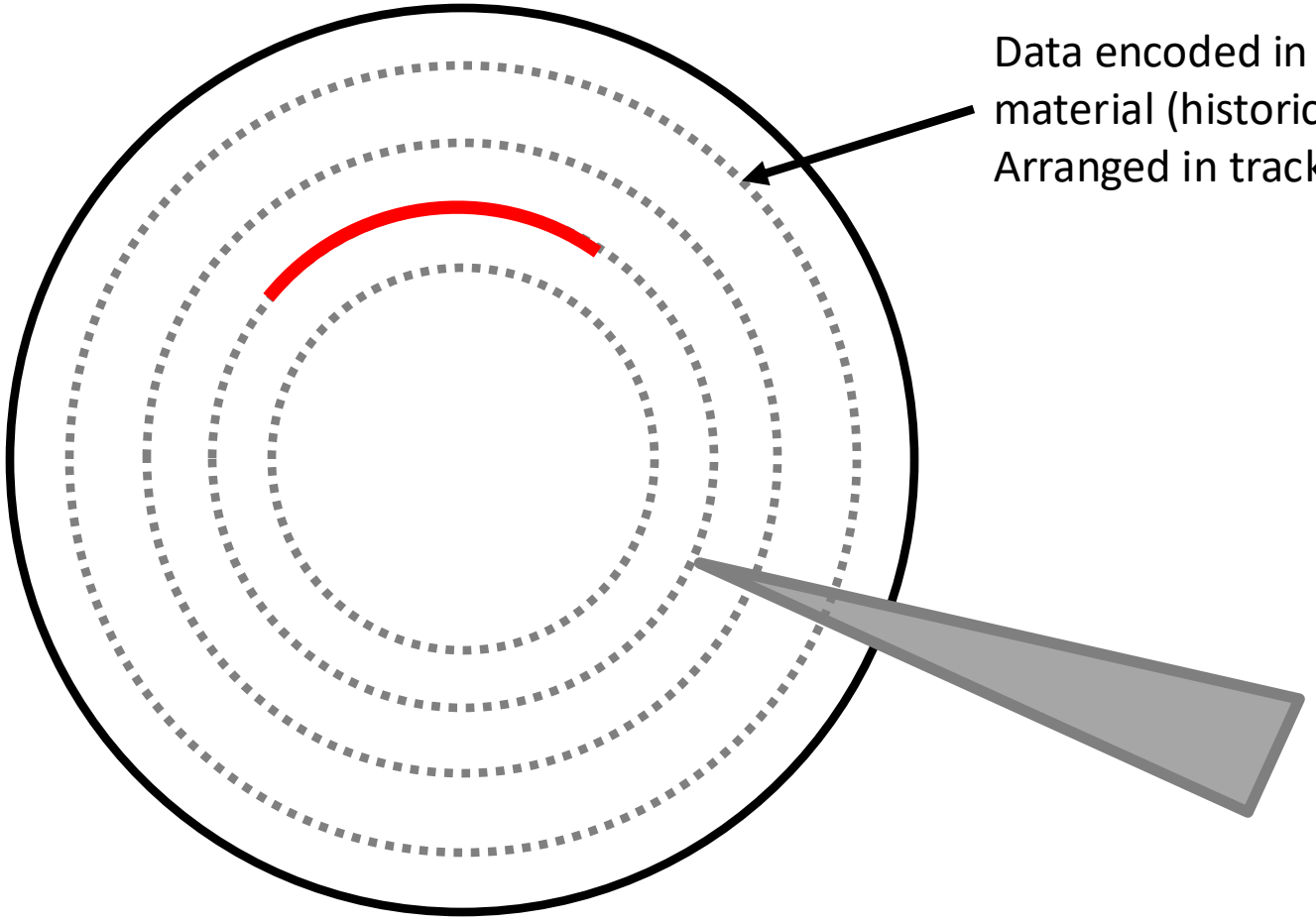
Actuated read head seeks between tracks

To read (simplified):

1. Seek to correct track
2. Wait for red data to rotate to the read head

Spinning Rust

Platter, spinning @ 15k RPM

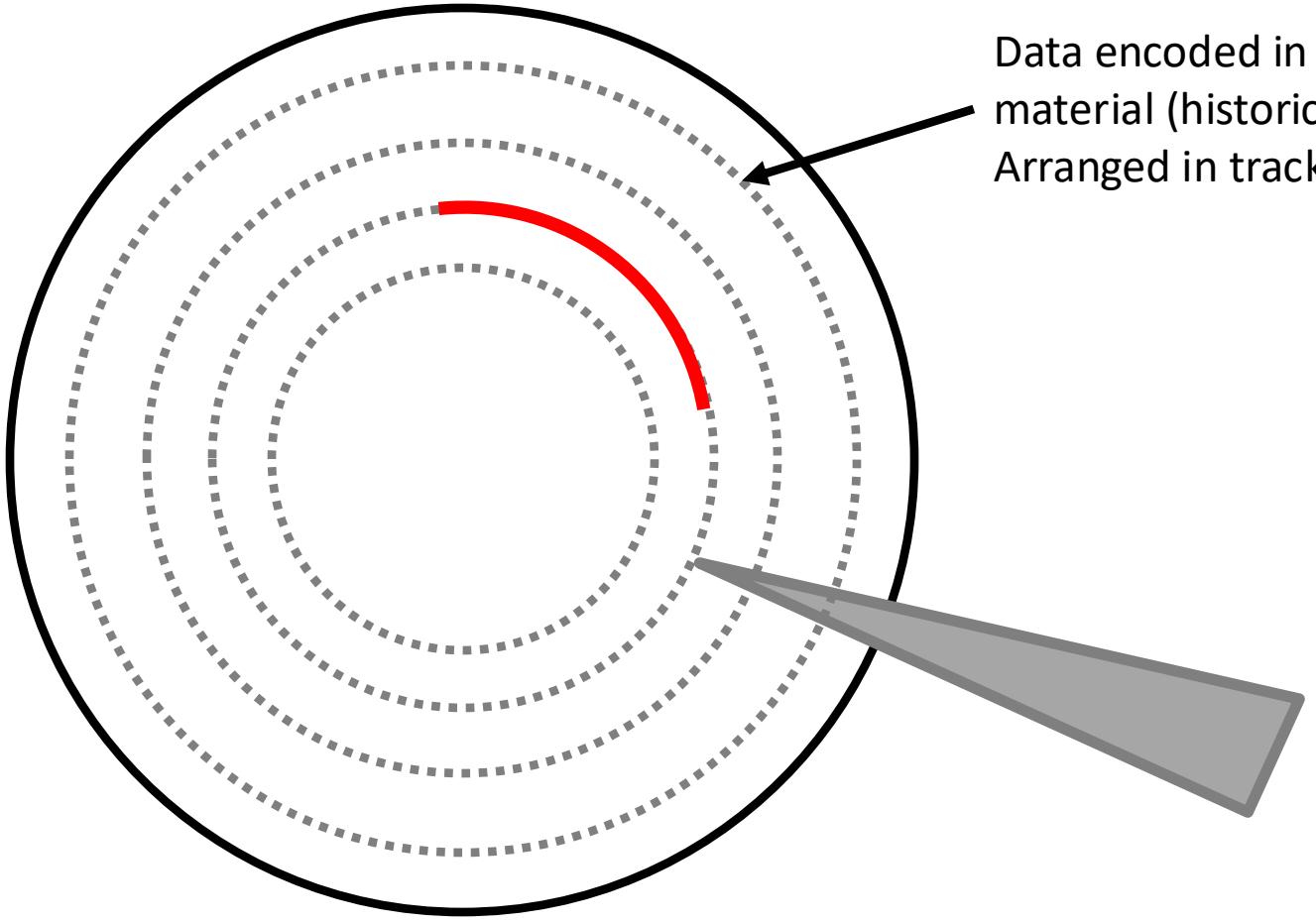


To read (simplified):

1. Seek to correct track
2. Wait for red data to rotate to the read head

Spinning Rust

Platter, spinning @ 15k RPM



Data encoded in magnetic material (historically iron oxide) Arranged in tracks

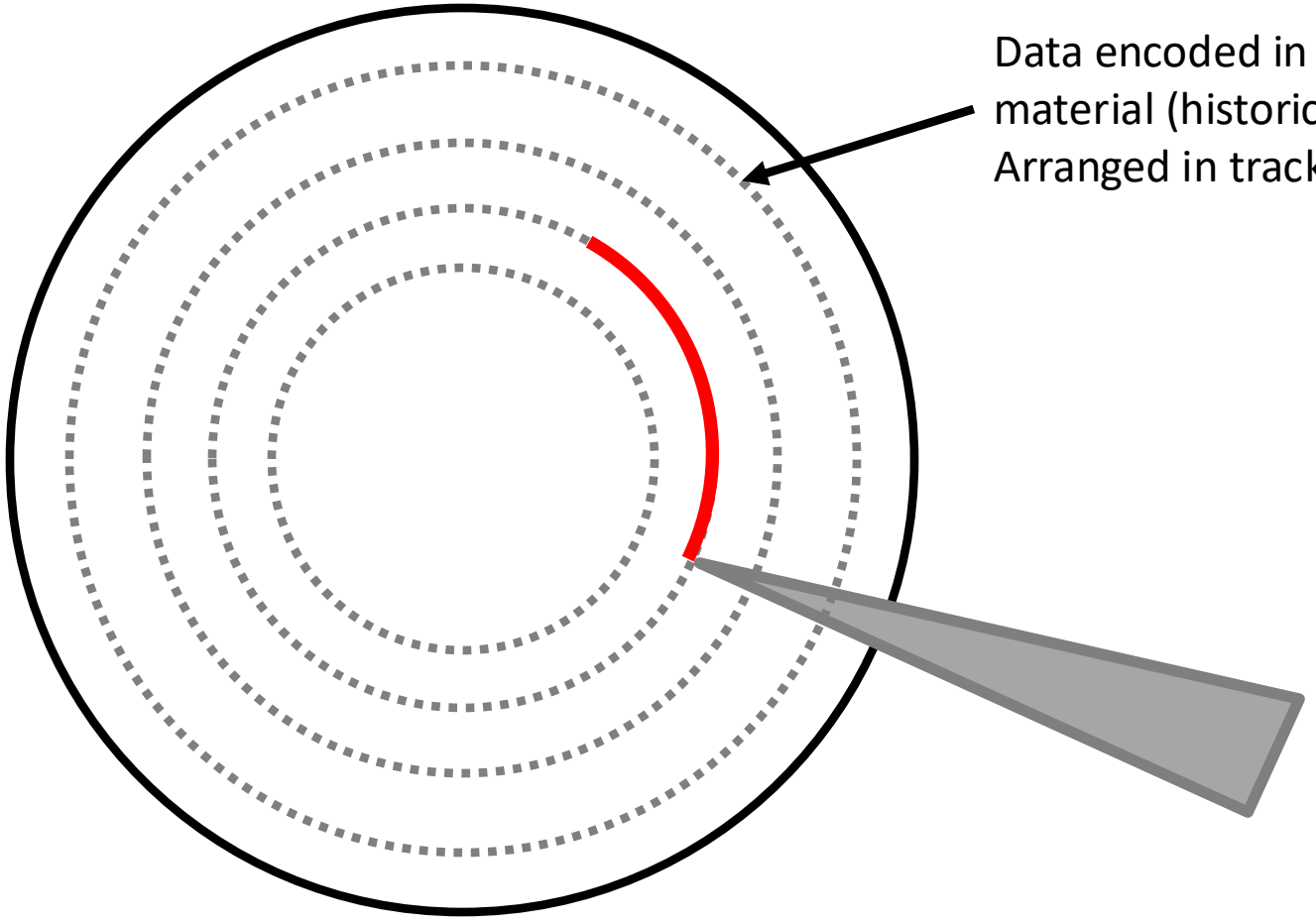
Actuated read head seeks between tracks

To read (simplified):

1. Seek to correct track
2. Wait for red data to rotate to the read head

Spinning Rust

Platter, spinning @ 15k RPM

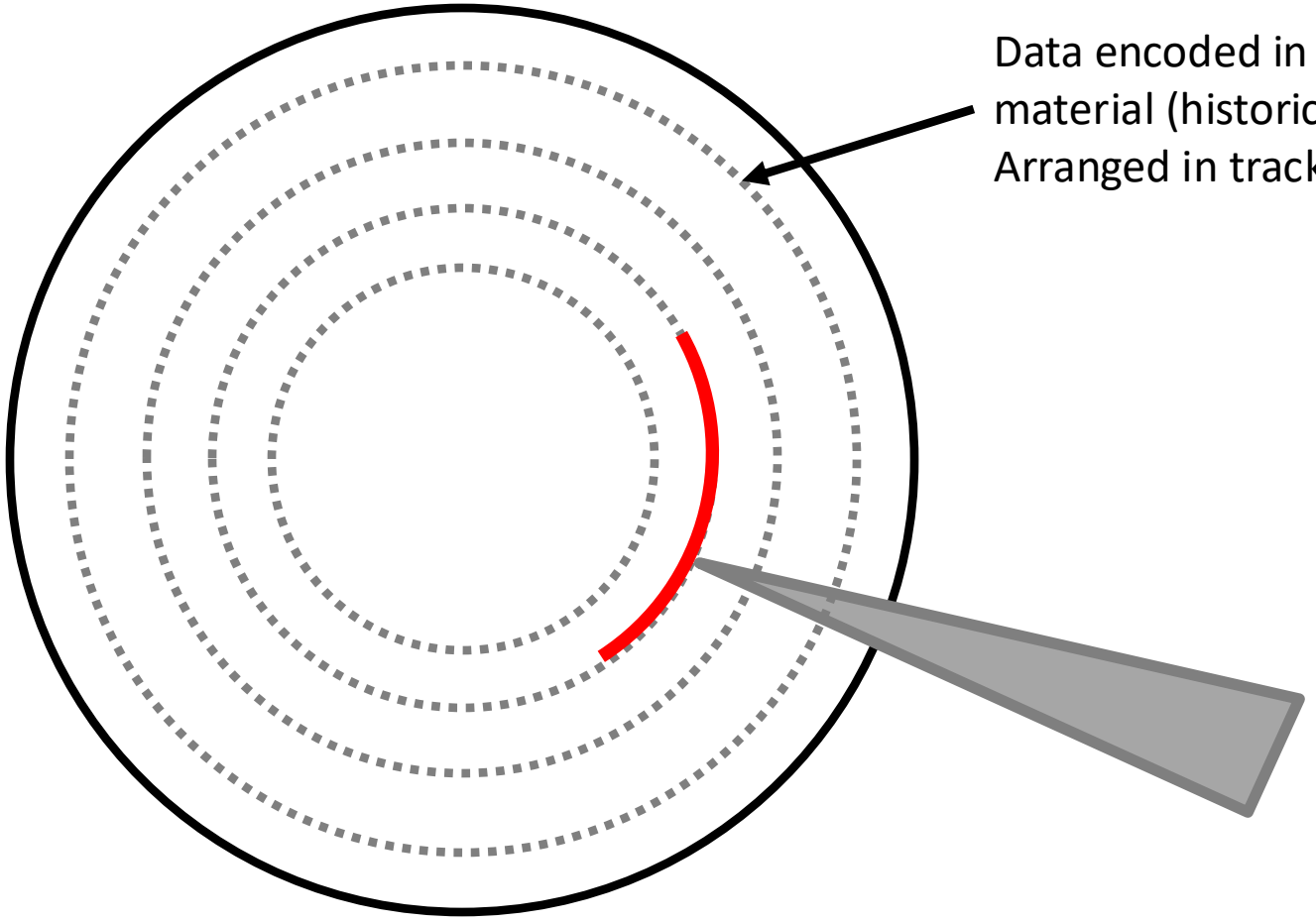


To read (simplified):

1. Seek to correct track
2. Wait for red data to rotate to the read head
3. Read data is it rotates by

Spinning Rust

Platter, spinning @ 15k RPM

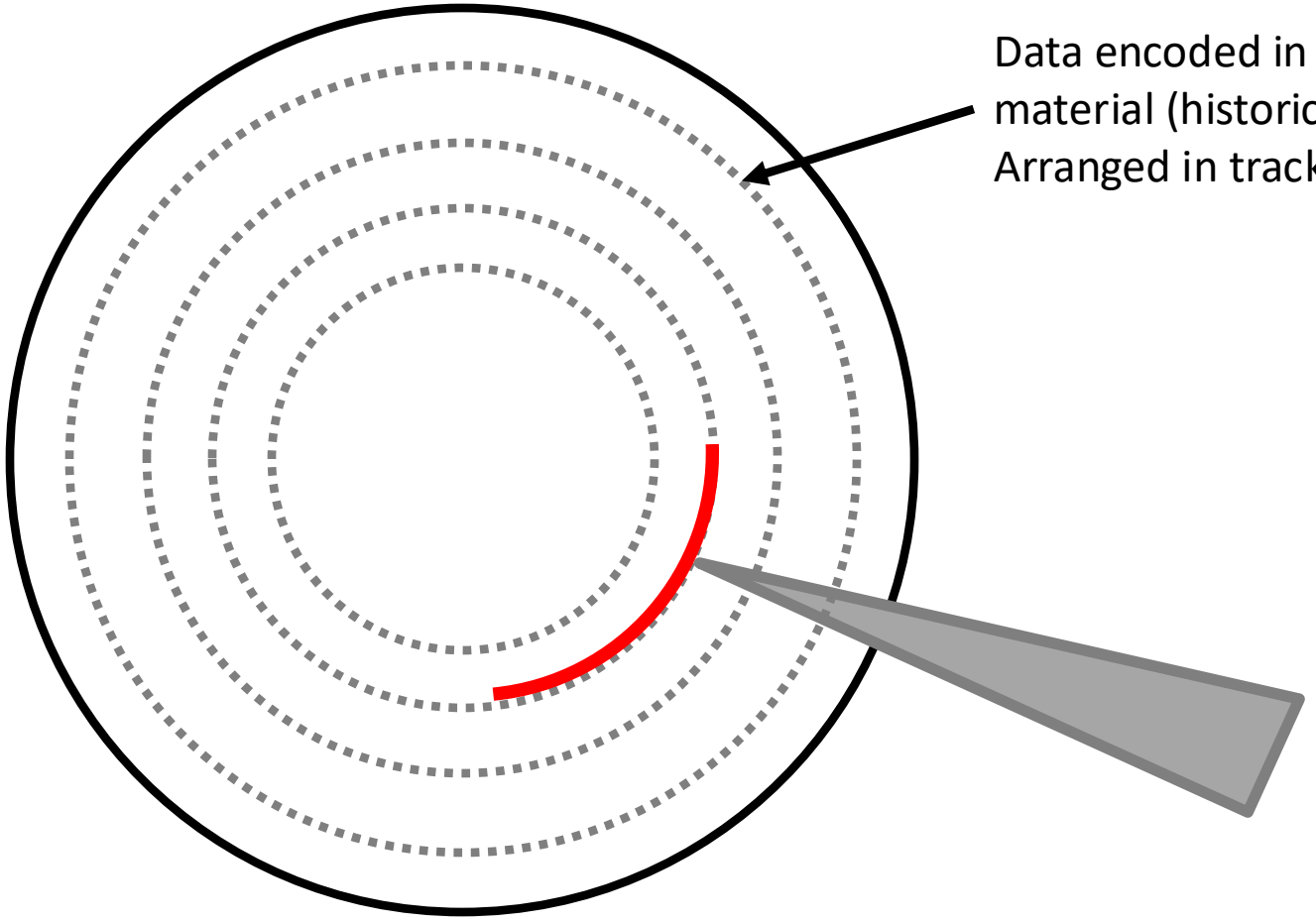


To read (simplified):

1. Seek to correct track
2. Wait for red data to rotate to the read head
3. Read data is it rotates by

Spinning Rust

Platter, spinning @ 15k RPM

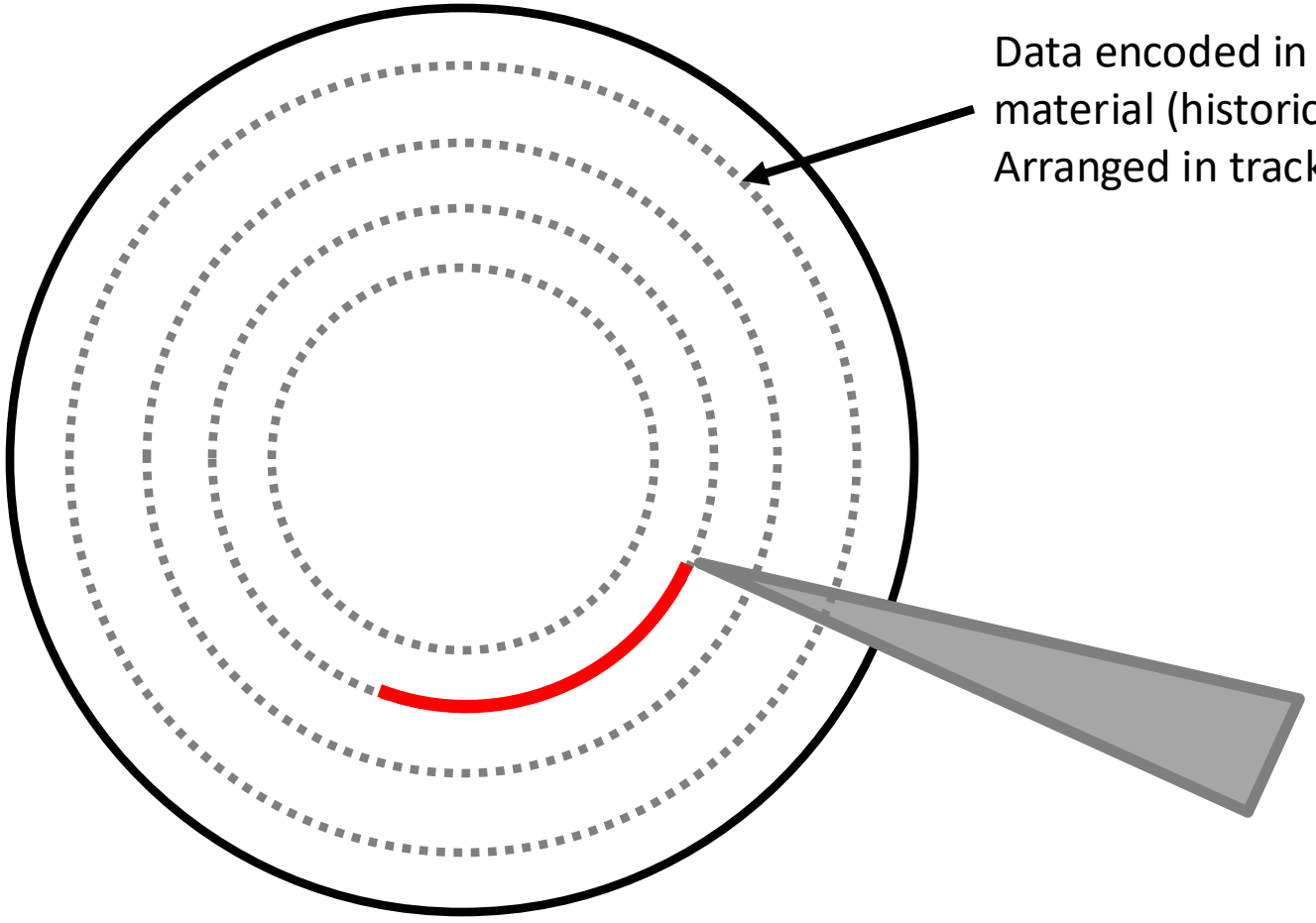


To read (simplified):

1. Seek to correct track
2. Wait for red data to rotate to the read head
3. Read data is it rotates by

Spinning Rust

Platter, spinning @ 15k RPM

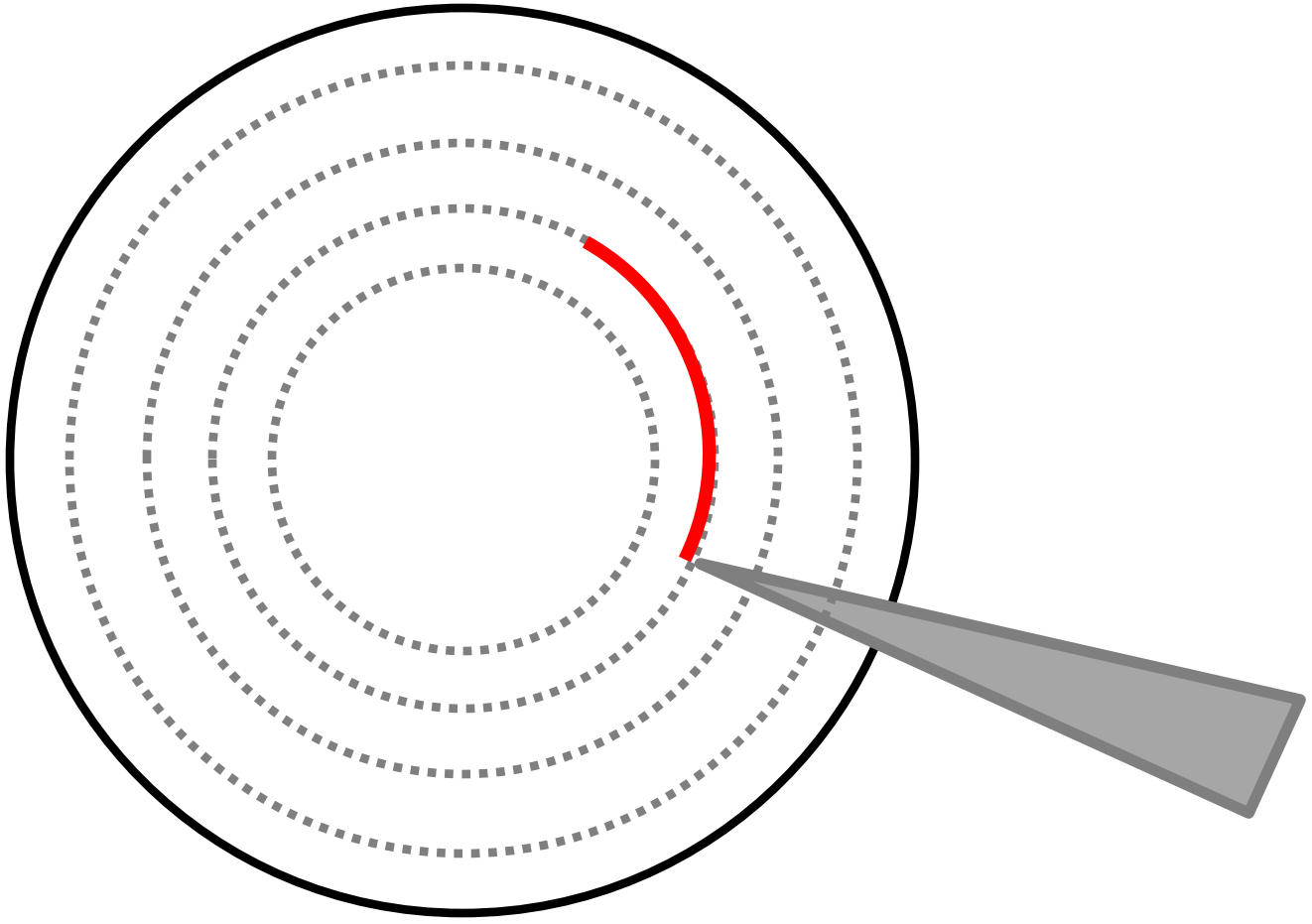


To read (simplified):

1. Seek to correct track
2. Wait for red data to rotate to the read head
3. Read data is it rotates by

Spinning Rust

Platter, spinning @ 15k RPM



Some math…

Total time to read X bytes:

T_{seek} = Time to move the read head to right track

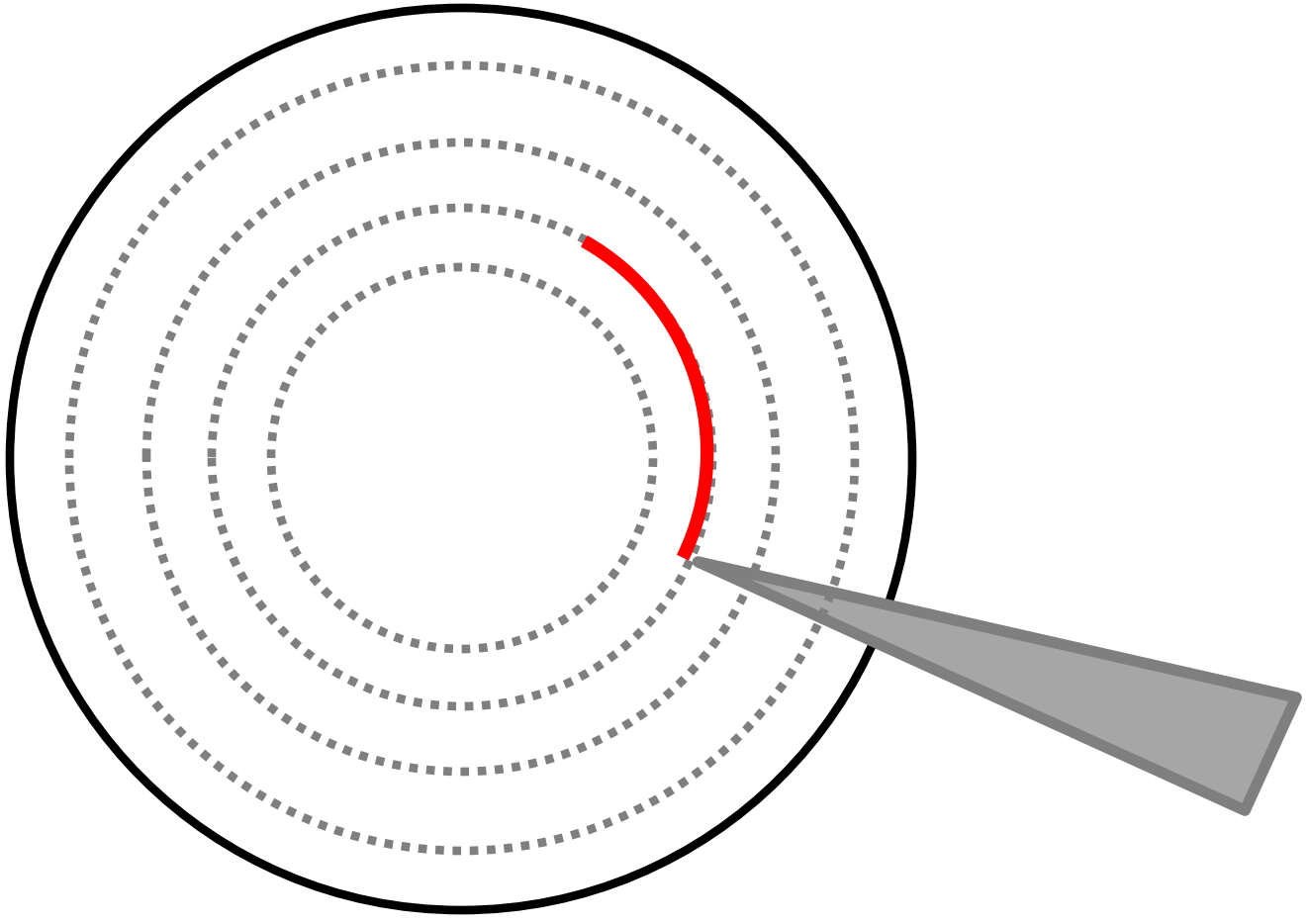
T_{rot} = Time to rotate to desired data (sector)

T_{read} = Time for read head to pass over data

Access Time $\approx T_{seek} + T_{rot} + T_{read}$

Spinning Rust

Platter, spinning @ 15k RPM



Some math…

-Average seek time: 2ms

-Average rotational time:

1/(15k RPM / 60)=4ms/rotation

we wait 2 ms on average

-Sequential read speed: 300 MB/s

To read 4KB of data:

2ms+ 2ms +.013 ms = 4.013ms

~1 MB/s

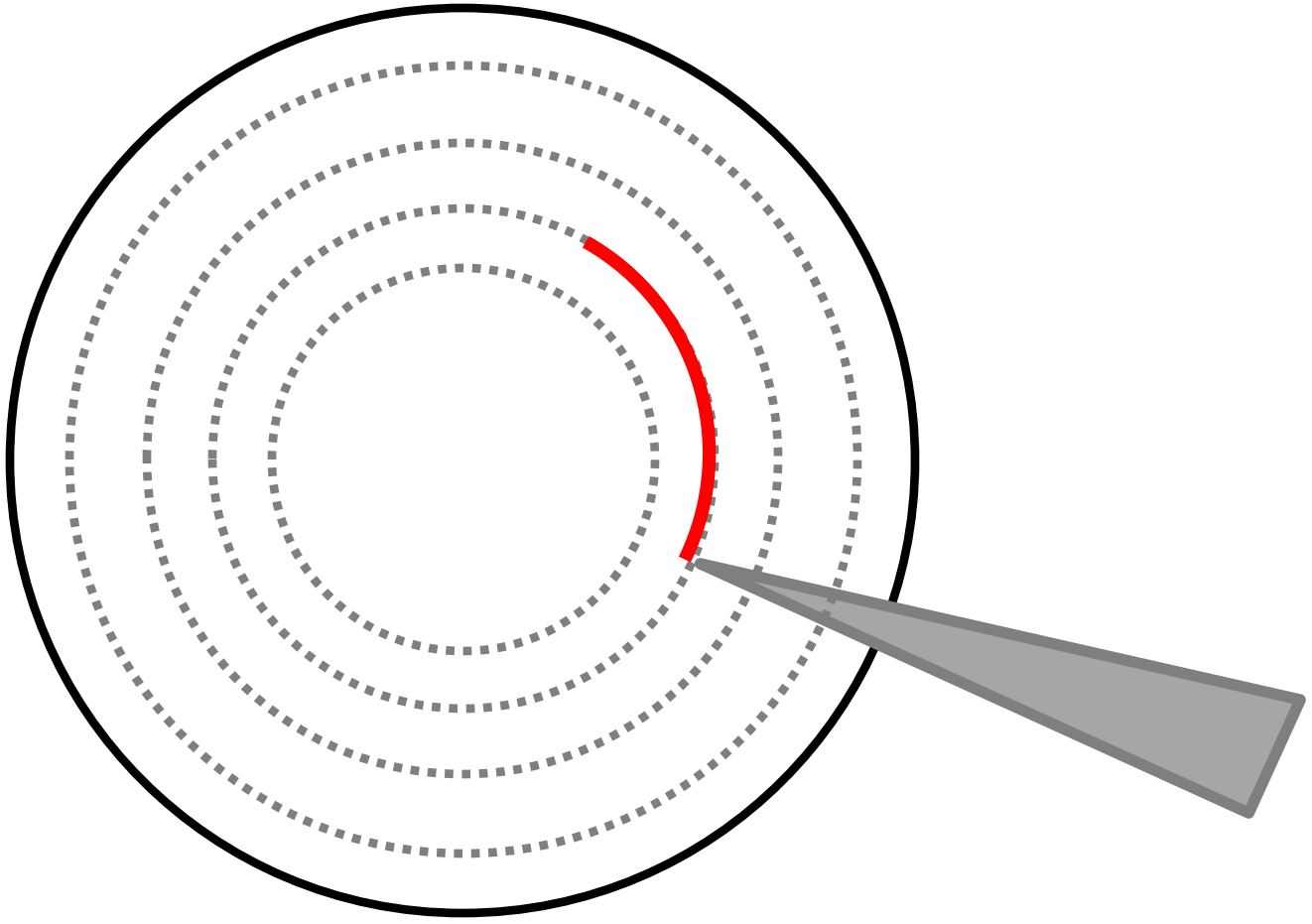
3GB of sequential data:

2ms+ 2ms + 10s = 10.002 s

299.94 MB/s ≈ 300 MB/s

Spinning Rust

Platter, spinning @ 15k RPM



- Sequential reading/writing of storage is essential
  - 2 orders of magnitude difference for HDD!
  - Less difference for SSD, other reasons to write sequentially
- Want to hide this from users
  - Don’t want to think about this while writing queries
- Want abstractions to help ourselves as system programmers

In This Course….

- Want data persistence and durable updates (storage), but also fast/complex data processing (memory)

  Lec. 3-4, Storage manager Lec. 6, Memory mgmt. Lec. 21, Recovery

- Want to process a drive worth of data (e.g. TB) using a server's worth of memory (e.g. 64 GB)

  Lec. 7-12, I/O optimized data structures, algorithms. Lec. 13-15, Query Exec./Opt.

- To leverage available compute, need concurrency at every level: intraquery, interquery, single-node, distributed database (multi-node)

  Lec. 16-19, Concurrency Control Lec. 22-24, Distributed databases

- Want abstractions/architecture to hide hardware/system details from users

  Lec 1 - 2.5, Relational algebra, SQL Lec 5, Storage Models Lec 20, Logging

Next Class…

- The relational algebra (a brief history of databases)
  - The basic data and programming model of a relational database
  - Why an idea from 1969 is still popular today
  - Some popular would-be alternatives