NYU | Center for Data Science

Week 01.3: Introduction

DS-GA 1004: Big Data

Instructor: Brian McFee

Class meetings

Usual recipe (~1 hour):

1. Recap / supplement to videos and reading
2. Group problem solving
3. Open Q&A

COVID precautions

- Many of us are missing tonight (and the first few weeks)
- Many of us might have to miss classes or labs throughout the semester
- That’s fine! The class is designed to be flexible, everything is recorded
- We probably won’t go fully remote/zoom unless things get very bad

Slido polls

- We’ll do interactive classroom activities through Slido
- These are “optional” / ungraded, and anonymous
- Polling helps me see which points need more discussion

slido

What's the least interesting fact about yourself?

ⓘ Start presenting to display the poll results on this slide.

Polling flow

- I'll post a question on the screen
  - **DO NOT SHOUT THE ANSWER!**
  - **1: Initial poll**: your individual response
  - **2: Discuss** with your neighbors for a couple of minutes
  - **3: Final poll**: revise your answer after discussion
- **Guessing is okay**! Your guesses show me where we need to focus.

Recap from videos

- This class is *mostly* about **distributed storage and computation**
- Our focus will be on problems that are **too big to solve with one machine**
- **Too big** ultimately depends on both the **amount of data** and the **specific computation** you're doing
    - ⇒ *Data structures and clever algorithms can be a better solution than a cluster!*

slido

Which strategies have you used before when dealing with large datasets?

ⓘ Start presenting to display the poll results on this slide.

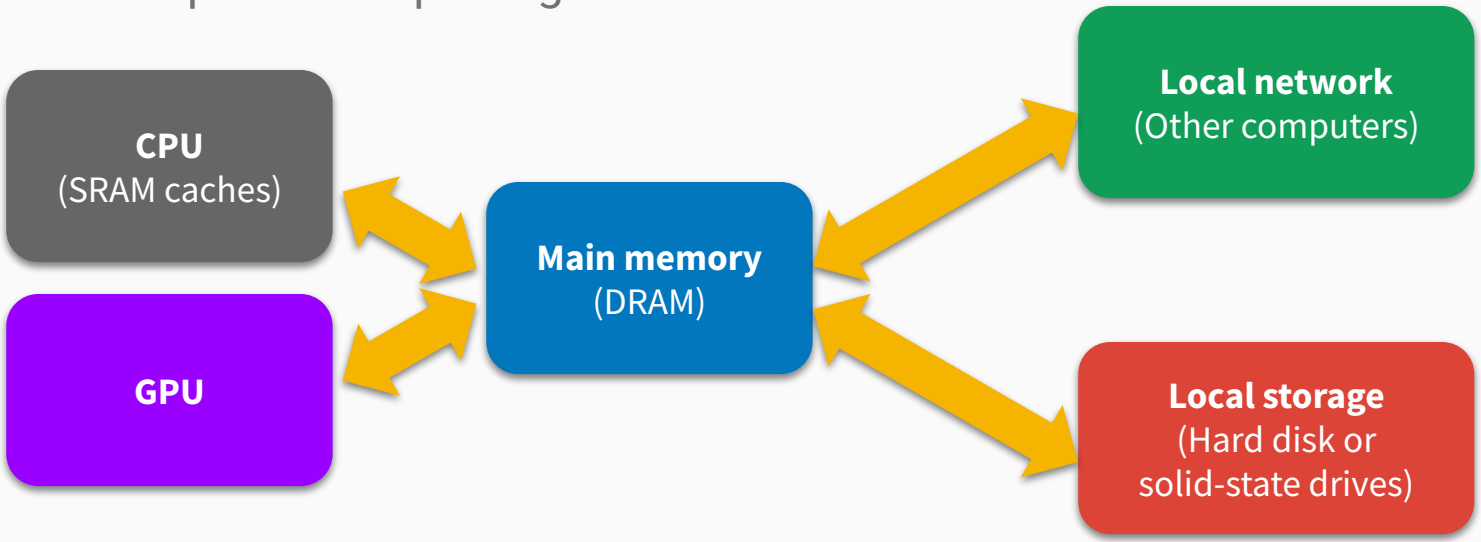
slido

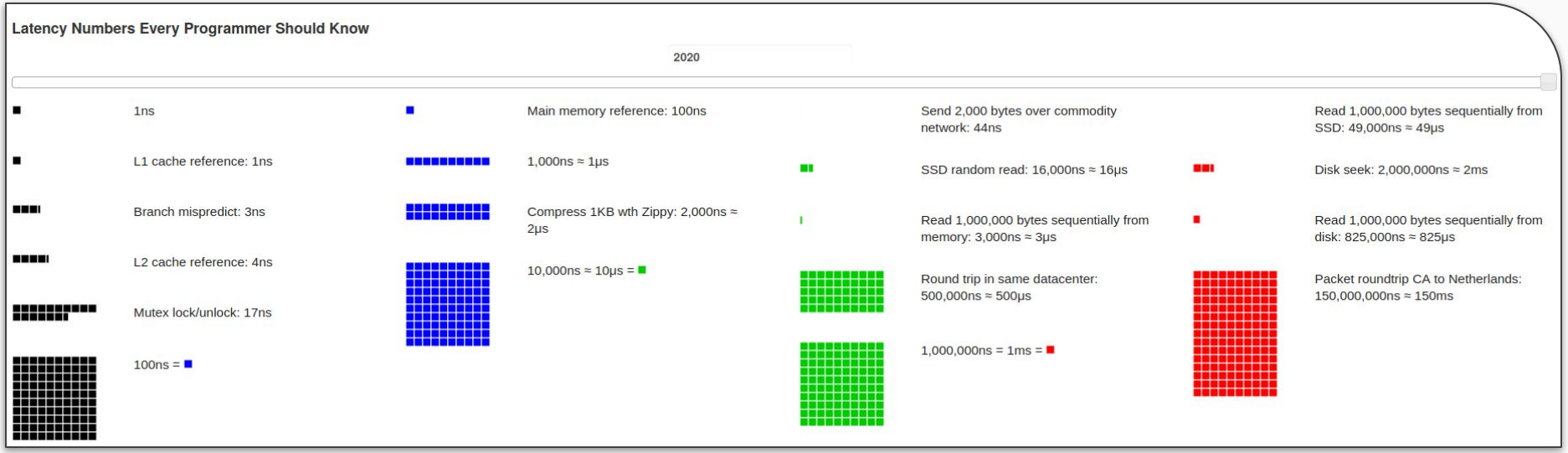
[2] Which strategies have you used before when dealing with large datasets?

ⓘ Start presenting to display the poll results on this slide.

Some basic computer organization

- To know what counts as “too big”, we’ll need to understand how computers are put together





L1 cache read	1 ns
L2 cache read	4 ns
Main memory read	100 ns
SSD random read	16,000 ns
HDD random read	2,000,000 ns

- Pre-fetching and pipelining can **accelerate sequential reads**
- **Main memory** is typically the first bottleneck
- The less communication (read/write) we have to do, the better

https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html

File-based storage

- Most data of note lives (permanently) **on disk / in the file-system**
- If it's small, we can load it into **main memory:**
  - df ← read_csv('my_data.csv')
  - analyze(df)
- If it's **too big**, we have several options:
  - **Sampling** / approximate computation
  - **Stream processing** (one or few records at a time)
  - **Data structures** / index structures
  - **Parallel computation**
  - **...**
  - Buy more memory 🙃

Good solutions often combine two or more of these strategies!

We'll see that often this semester.

Some things to consider

- Is an **exact answer** required?
  - Would an estimate be good enough?
  - Can we bound estimation errors if sampling?
- Do we expect **erroneous** or ill-formatted records?
  - What if errors are correlated with the data?
  - We may have to examine each record anyway
- Will this dataset grow over time?
  - Streaming might be beneficial
- Might we add more features to this analysis later on?

slido

Say you had to compute the mean and covariance matrix of a "large" collection of 100-dimensional vectors. How would you rank the following strategies (best to worst)?

ⓘ Start presenting to display the poll results on this slide.

slido

[2] Say you had to compute the mean and covariance matrix of a "large" collection of 100-dimensional vectors. How would you rank the following strategies (best to worst)?

ⓘ Start presenting to display the poll results on this slide.

Pros and cons of each strategy

- **Sampling** will give an approximate answer, but is likely to miss outliers
  - What if 0.001% of your records are corrupted?
- **Stream processing** will limit memory usage, but not CPU time
- **Indexing** doesn't help much: the problem inherently uses all the data
- **Parallelism** can bring down wall-clock time if you have many machines

Where we're going next

- **Database management systems** (DBMS)
  - Provide a standardized interface to store, load, and process data
  - We'll see many additional benefits next week
- The **relational model** imposes constraints on how data is organized
  - i.e., tables / spreadsheets / dataframes
- Putting these two ideas together = **RDBMS**!

Next time…

Relational model and databases

- Reading for next week
  [Garcia-Molina, Ullman, & Widom, 2009, ch2]
- Week 2 videos are up
- Wednesday (01/26):
  - Lab 0, environment setup

slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.