[544] Caching

Tyler Caraza-Harter

Learning Objectives

- describe the cache hierarchy
- trace through access patterns with LRU and FIFO policies
- calculate cache performance metrics

Outline

Challenge: Latency

Cache Hierarchy

- CPU, RAM, SSD, Disk, Network
- Tradeoffs

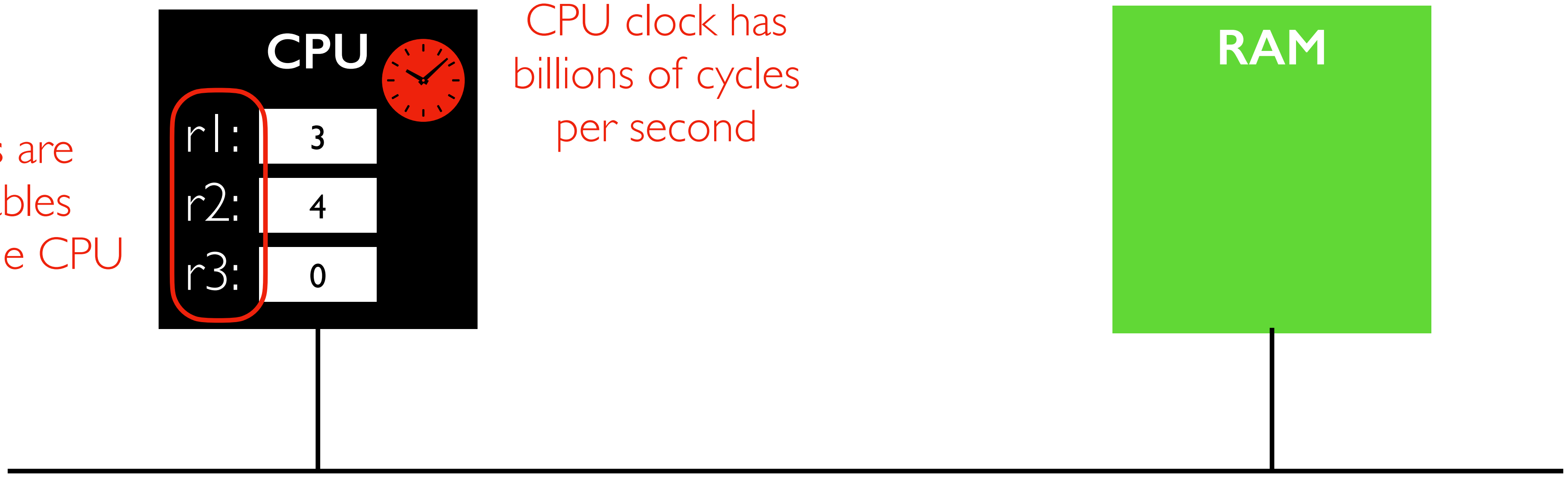
Policy: what data should be cached?

- manual
- expiration
- eviction policies: random, FIFO, LRU

Practice

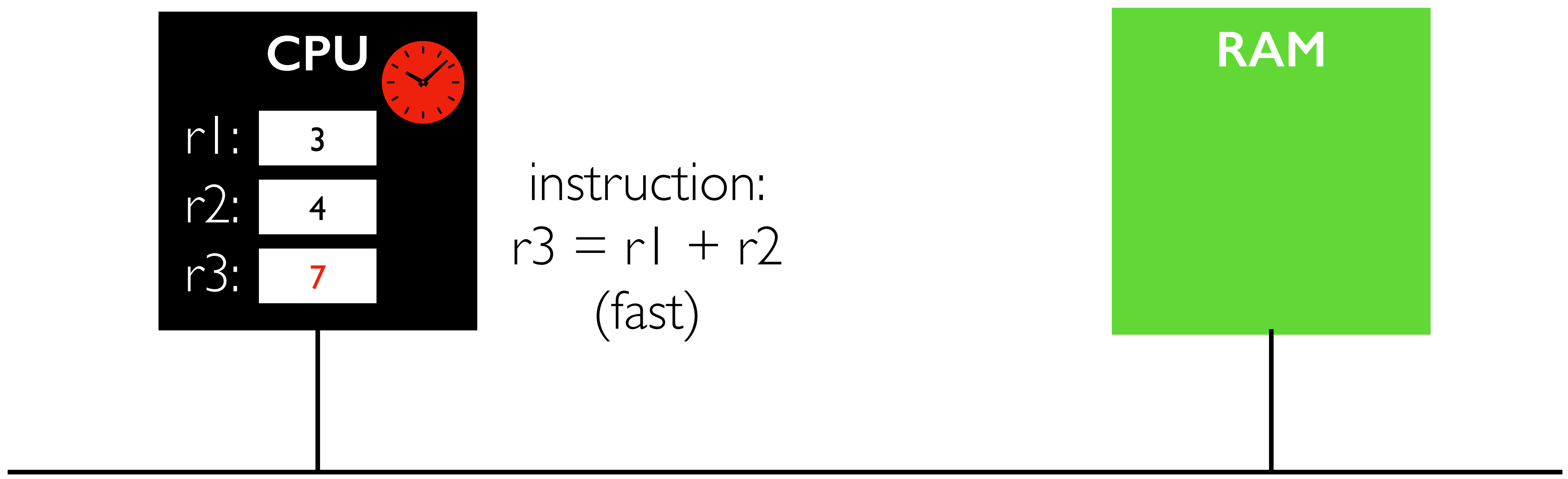
CPU and RAM

registers are like variables built into the CPU



CPU clock has billions of cycles per second

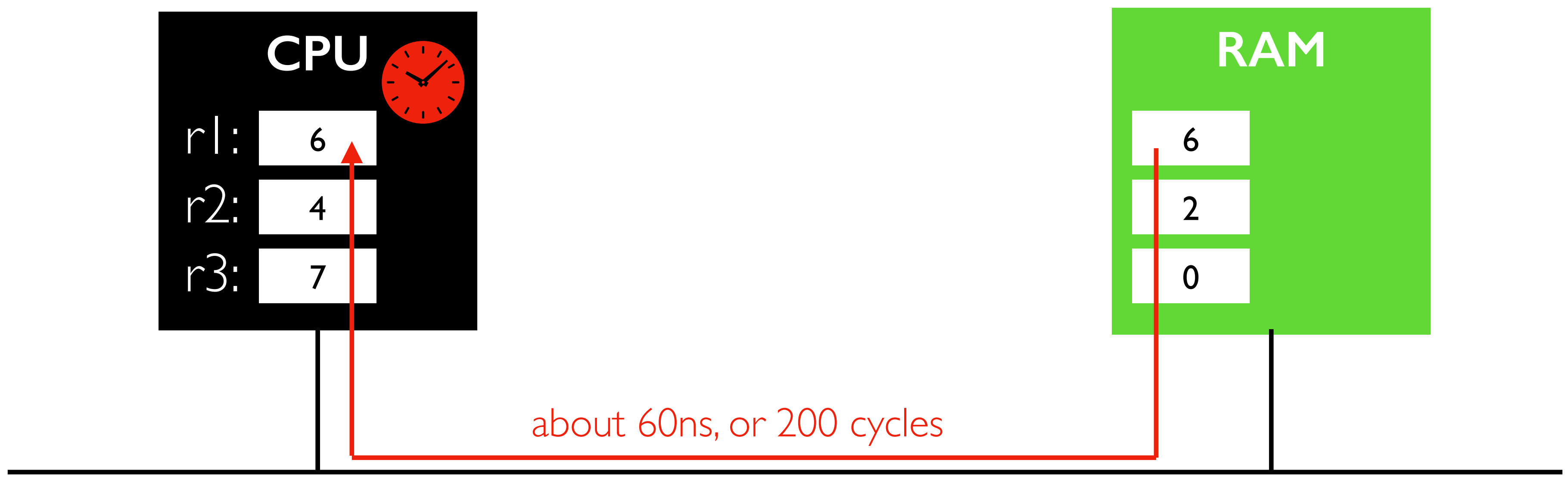
CPU and RAM



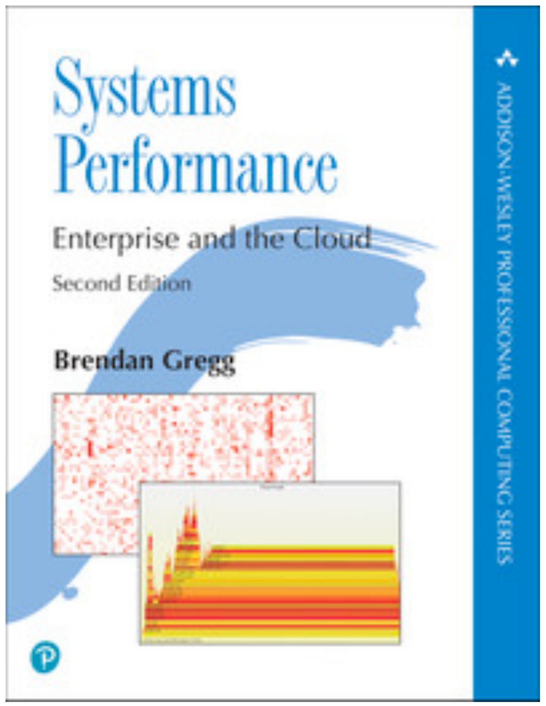
Load and Store

challenge: if we want to add some numbers stored in RAM, we need to **load** before adding and **store** after

Latency

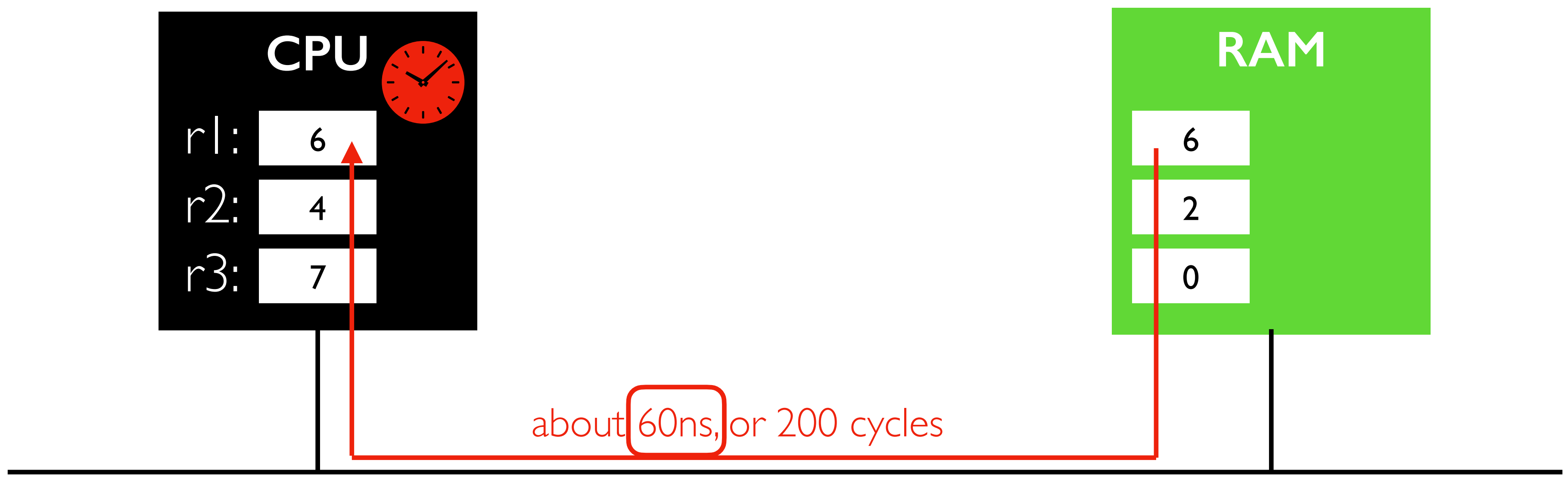


very slow, but not long enough to
switch to a different process...

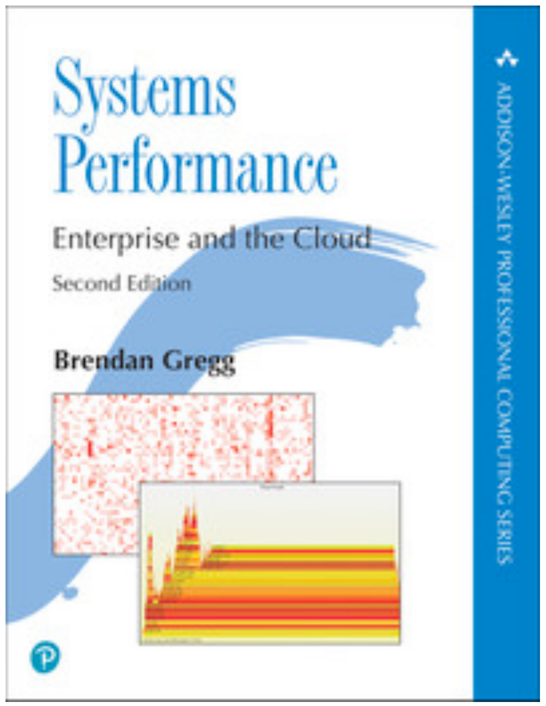


source: visuals, estimates

Latency



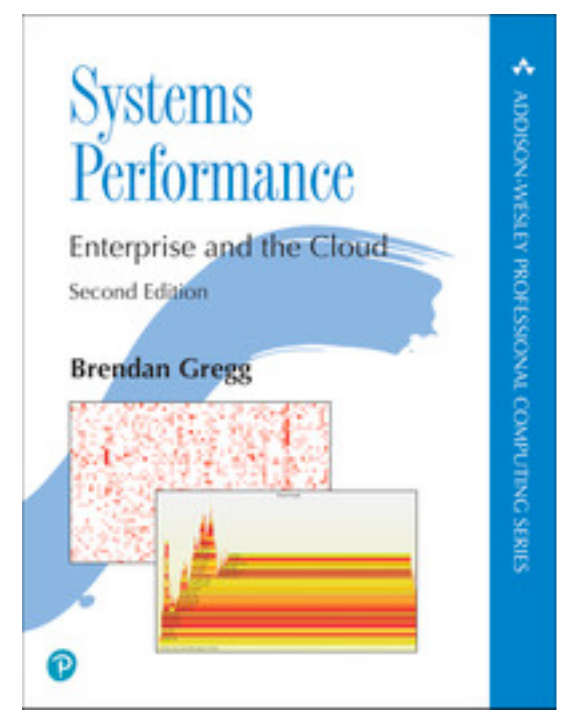
"how much time" is a latency measure.
Throughput (bytes/second) would depend on how many loads like these we can do simultanously.



source: visuals, estimates

Cache

Idea: CPUs can have a small/fast memory built in for data that is accessed frequently



source: visuals, estimates

Performance Measurements

Metrics

- throughput
- average latency
- median latency
- "tail" latency (99th percentile, 99.9th percentile, etc).

Which metrics do we expect caching to help with the most?

Outline

Challenge: Latency

Cache Hierarchy

- CPU, RAM, SSD, Disk, Network
- Tradeoffs

Policy: what data should be cached?

- manual
- expiration
- eviction policies: random, FIFO, LRU

Practice

Cache Hierarchy

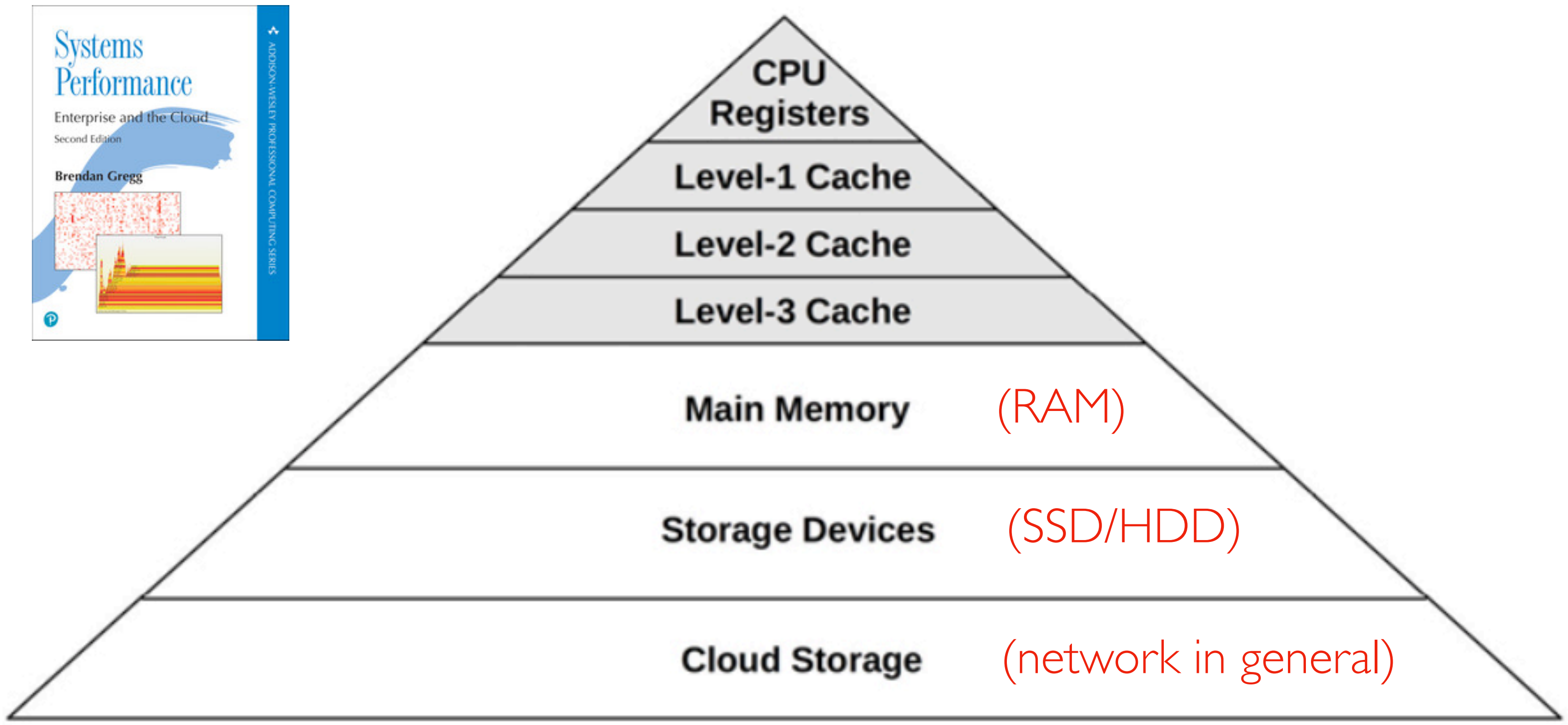
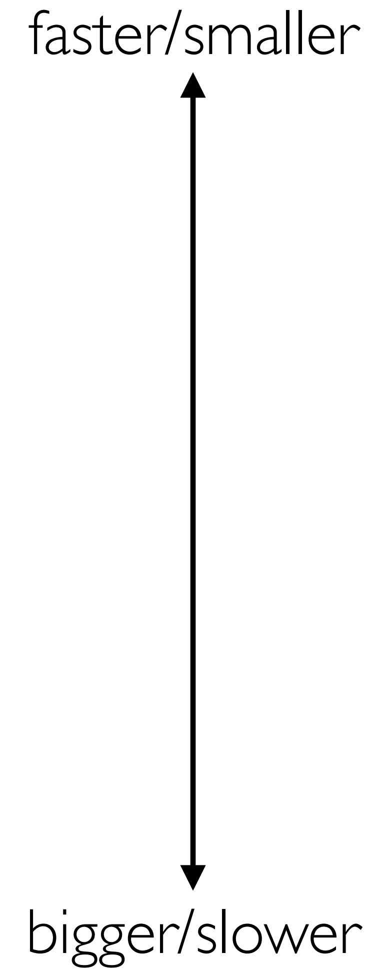


Figure 6.2 CPU cache sizes

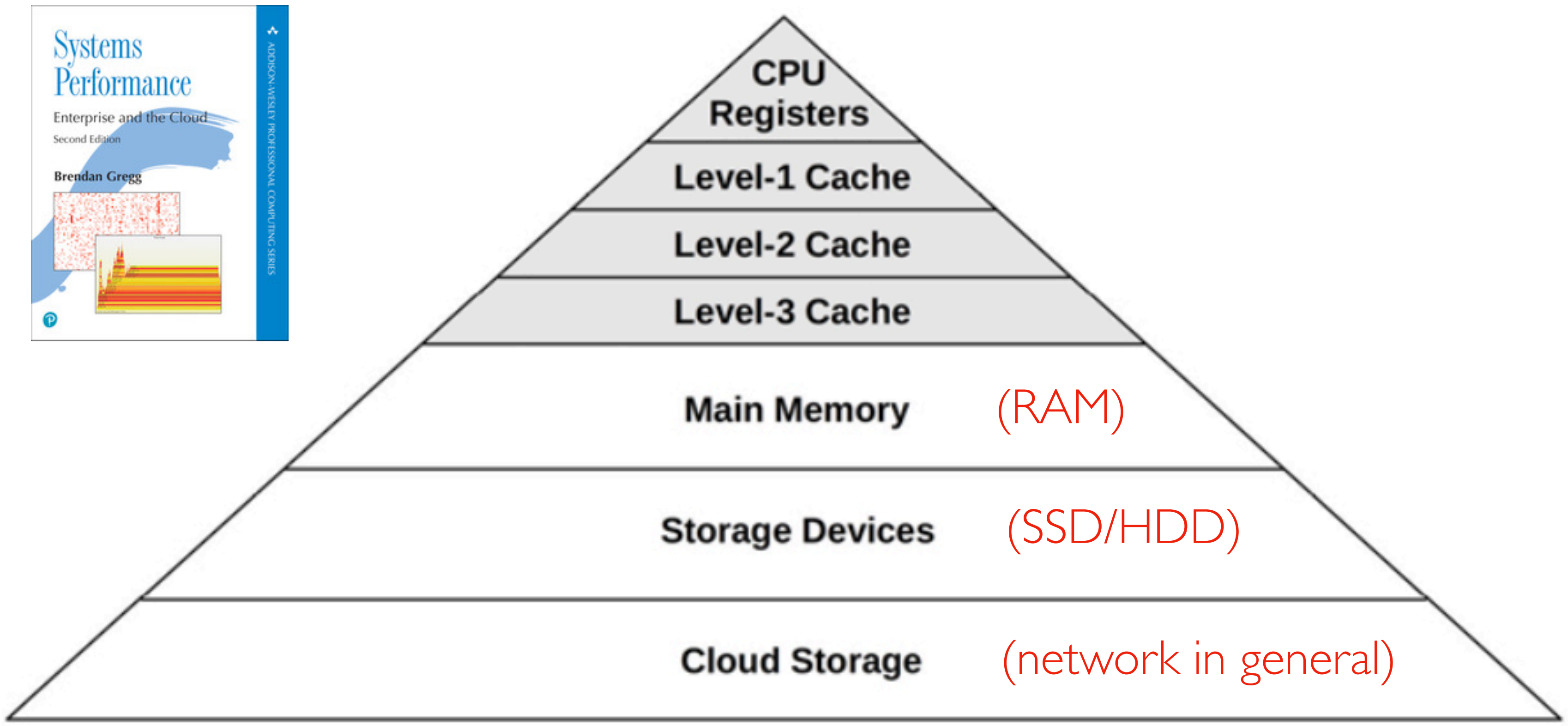


Example: Intel Xeon Platinum 9282 (2019)

- L1: 64 KB
- L2: 1 MB
- L3: 77 MB

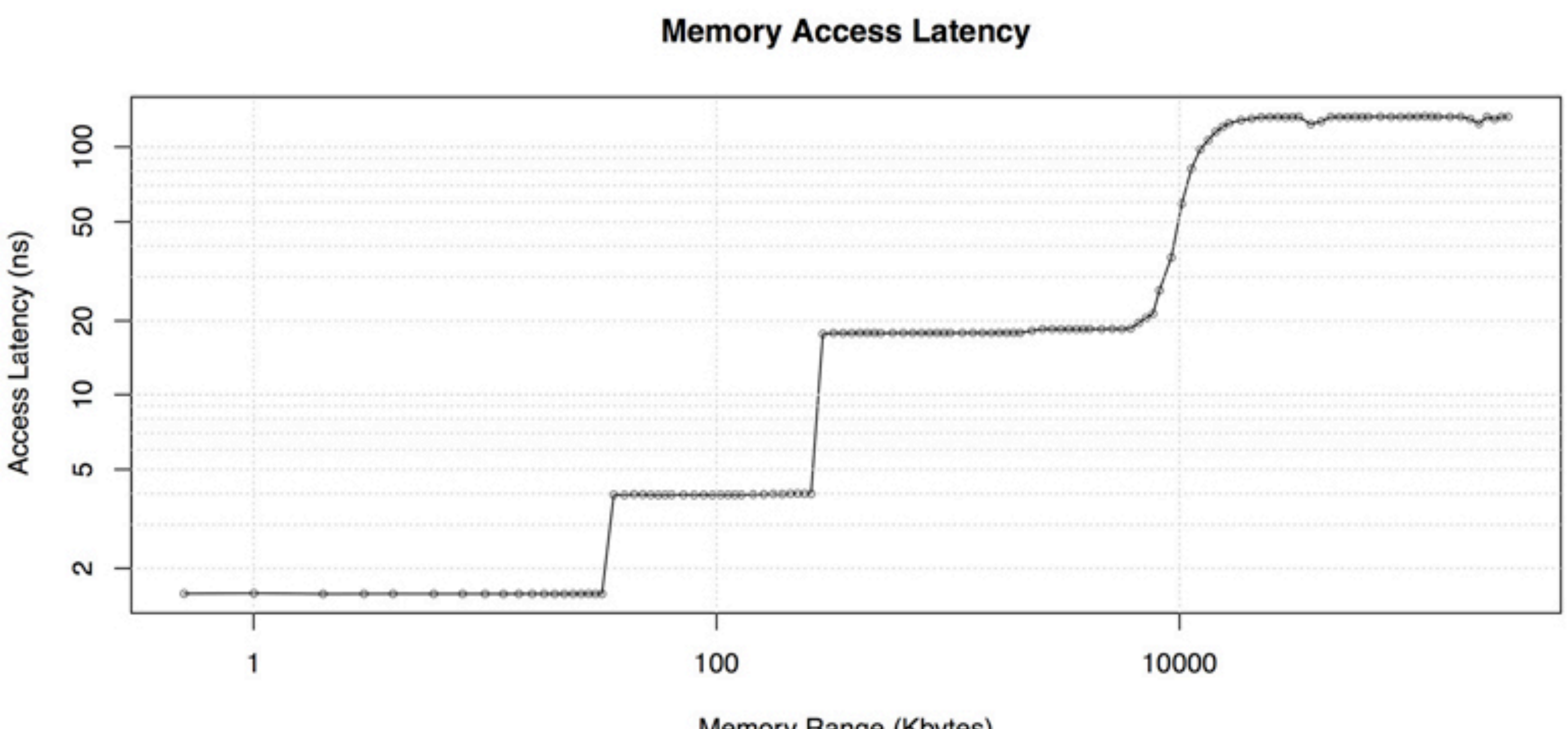
Cache Hierarchy

faster/smaller



bigger/slower

Figure 6.2 CPU cache sizes



about how big is the L3 cache? what is the latency for an L3 cache access?

Resource Tradeoffs

File system caches file data in RAM

- use memory
- avoid storage reads

Browser caches recently visited page as file

- uses storage space
- avoid network transfers

Python dictionary caches return values in a dict (key=args, val=return)

- uses memory space
- avoid repeated compute

```
cache = {}
def f(x):
    if not x in cache:
        cache[x] = g(x)
    return cache[x]
```

Outline

Challenge: Latency

Cache Hierarchy

- CPU, RAM, SSD, Disk, Network
- Tradeoffs

Policy: what data should be cached?

- manual
- expiration
- eviction policies: random, FIFO, LRU

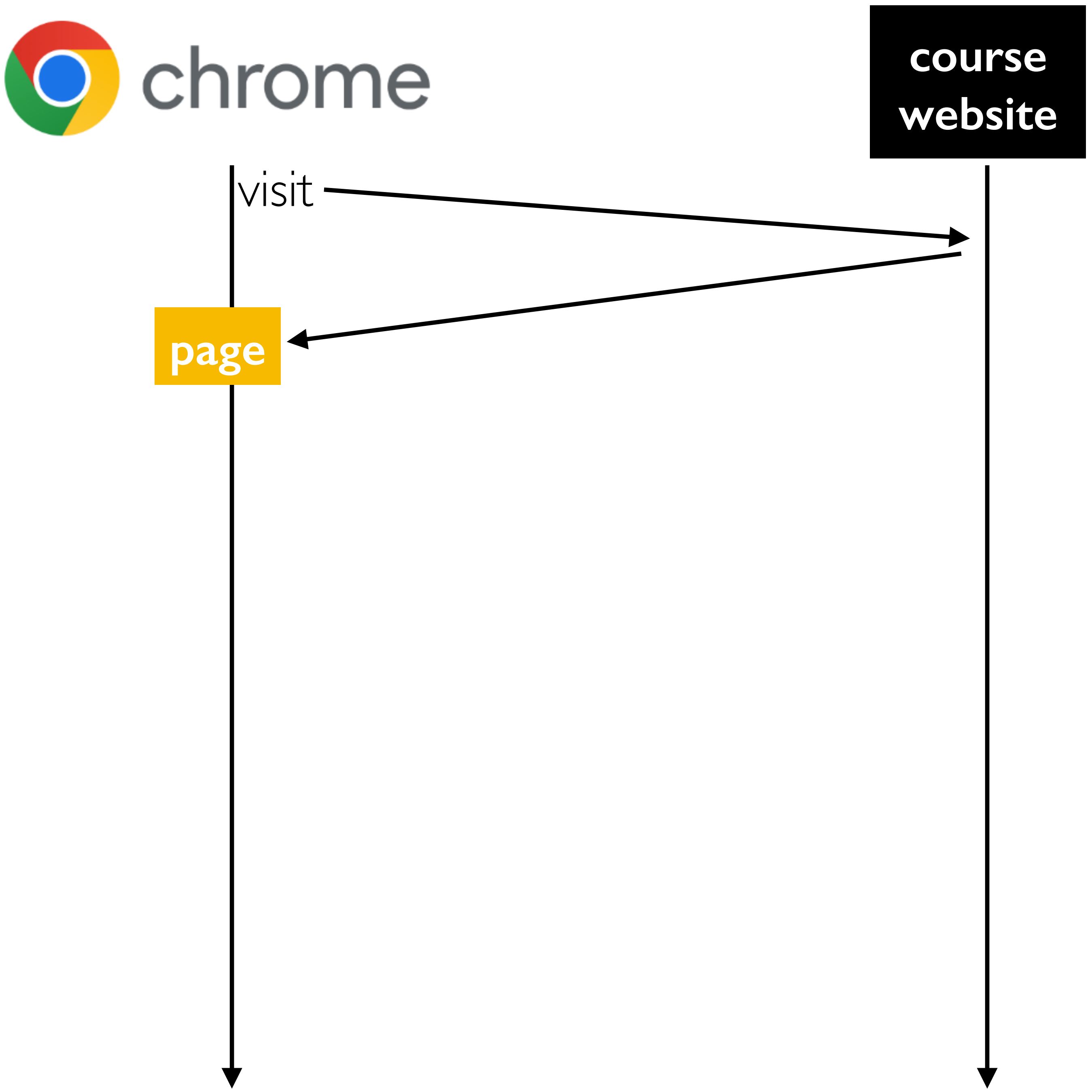
Practice

Manual Caching: Spark Example

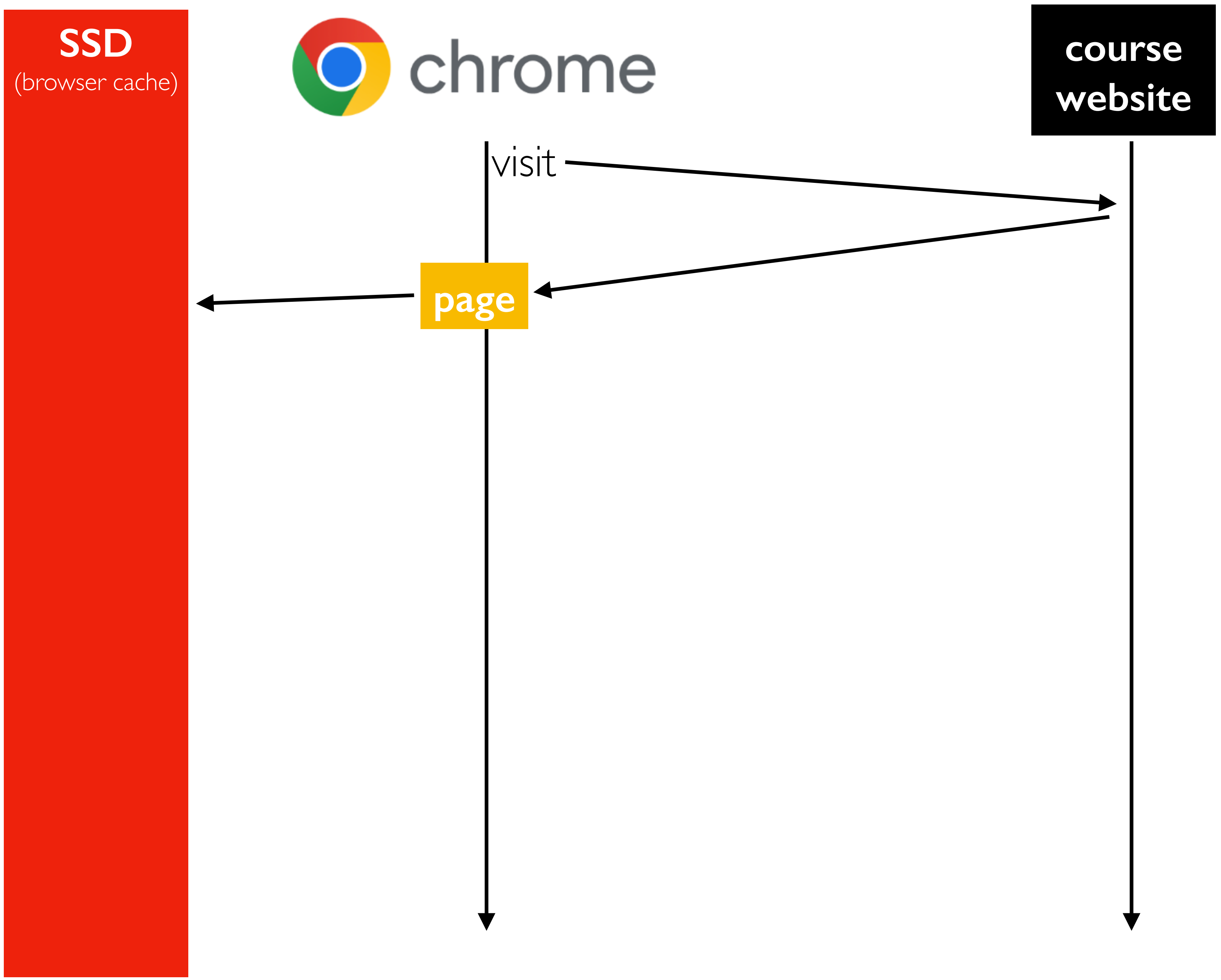
```
spark_df = ???? # not usually in memory

spark_df.cache() # put it in memory

# use spark_df for a lot of calculations

spark_df.unpersist() # free up memory
```

we'll be spending lots of time on Spark later in the semester

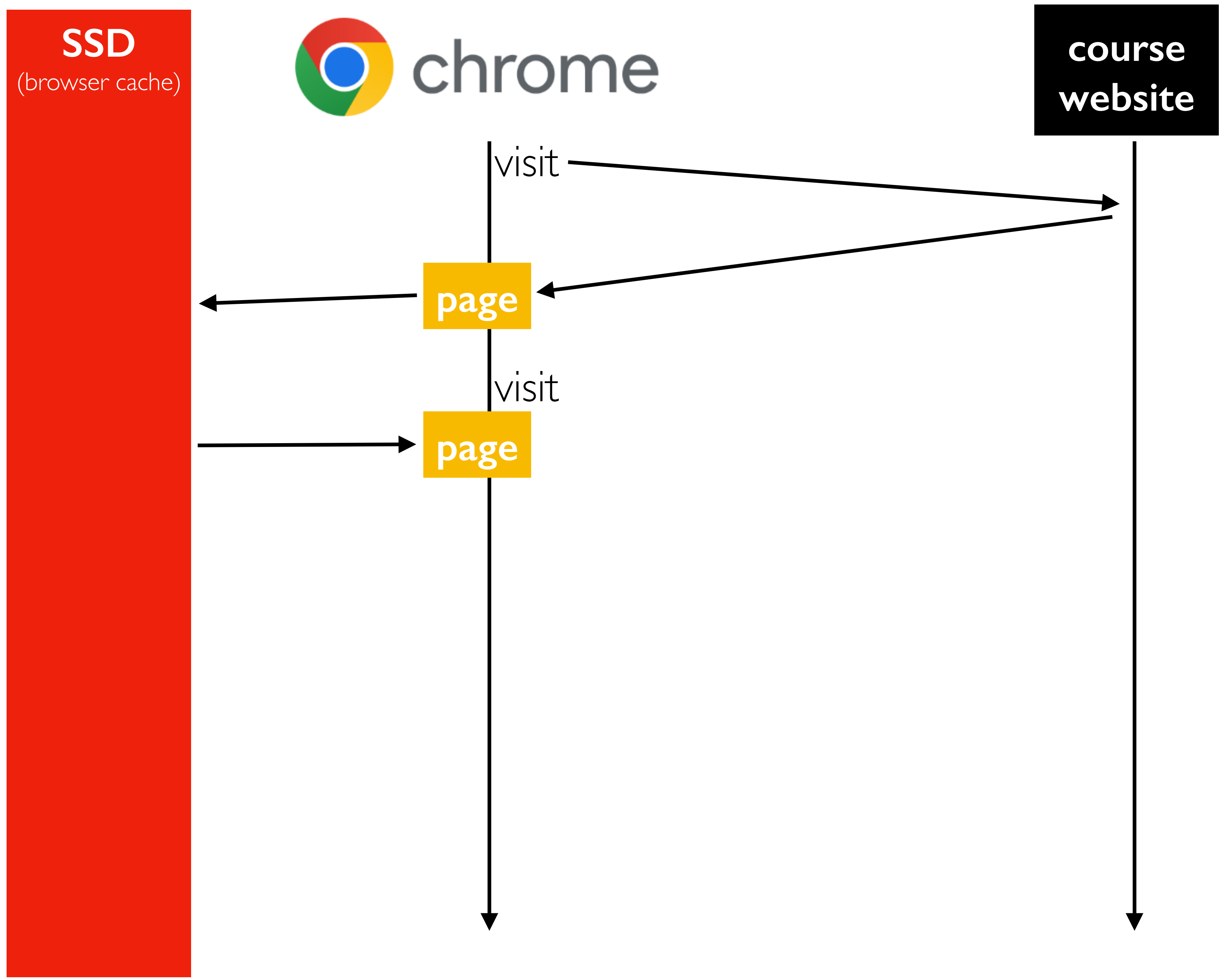
Expiration: Browser Example



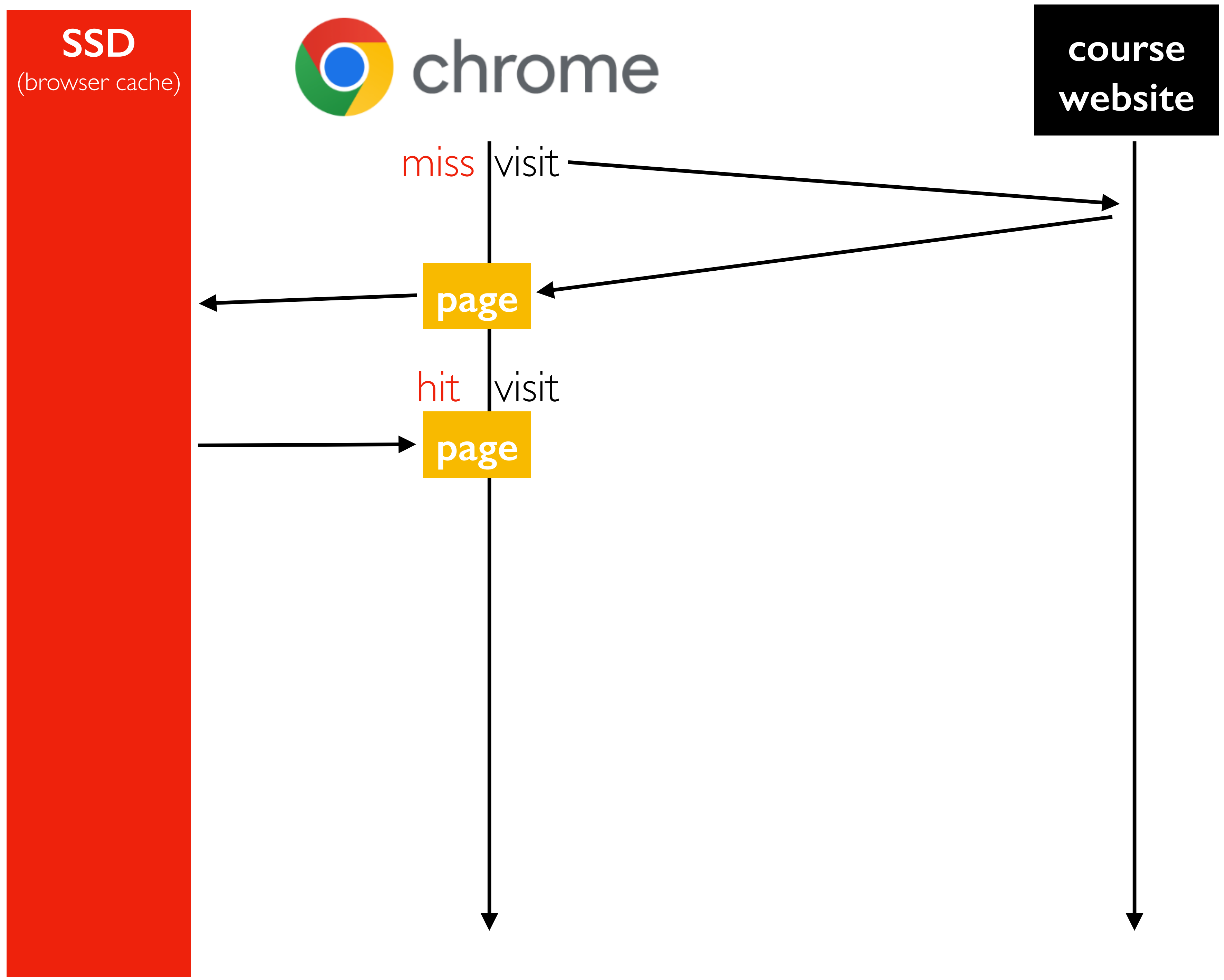
Expiration: Browser Example



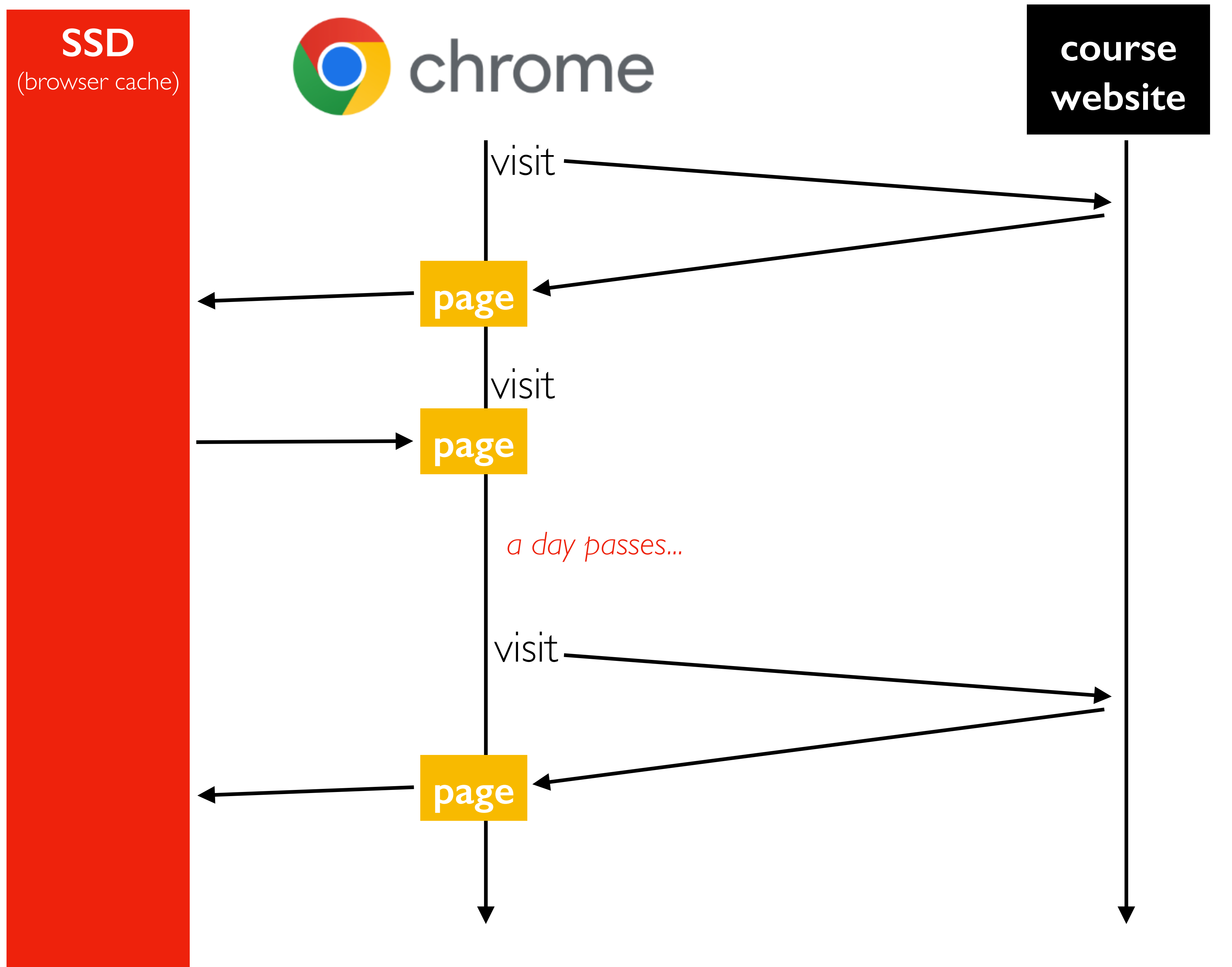
Expiration: Browser Example



Expiration: Browser Example



Expiration: Browser Example



stale data (past expiration) is deleted (or re-valitaded).
SSD is large so freshness is more important than space.

Outline

Challenge: Latency

Cache Hierarchy

- CPU, RAM, SSD, Disk, Network
- Tradeoffs

Policy: what data should be cached?

- manual
- expiration
- eviction policies: random, FIFO, LRU

Practice

Cache Policies

When to load data to a cache?

- whenever we read something, add it
- special exception: programmer knows it will never be read again
  - for example, F_NOCACHE option in Linux

When to evict data to a cache? Several policies

- random
  - select any entry at random as victim for eviction
- FIFO (first in, first out)
  - evict whichever entry has been in the cache the longest
- LRU (least recently used)
  - evict whichever entry has been used the least recently

Worksheet

Outline

Challenge: Latency

Cache Hierarchy

- CPU, RAM, SSD, Disk, Network
- Tradeoffs

Policy: what data should be cached?

- manual
- expiration
- eviction policies: random, FIFO, LRU

Practice