Thank You to the Code PaLOUsa Sponsors

Progress

WAYSTAR

Bastian SOLUTIONS
a TOYOTA ADVANCED LOGISTICS company

<code> LOUISVILLE

eblu SOLUTIONS

twilio

Scout

aws

cbts

<prosoft>

CGI

Friends of Code PaLOUsa

Asperitas

Couchbase

DATASTAX

mongoDB.

redis

ROCKET Mortgage

Couchbase

Cache Rules Everything Around Me

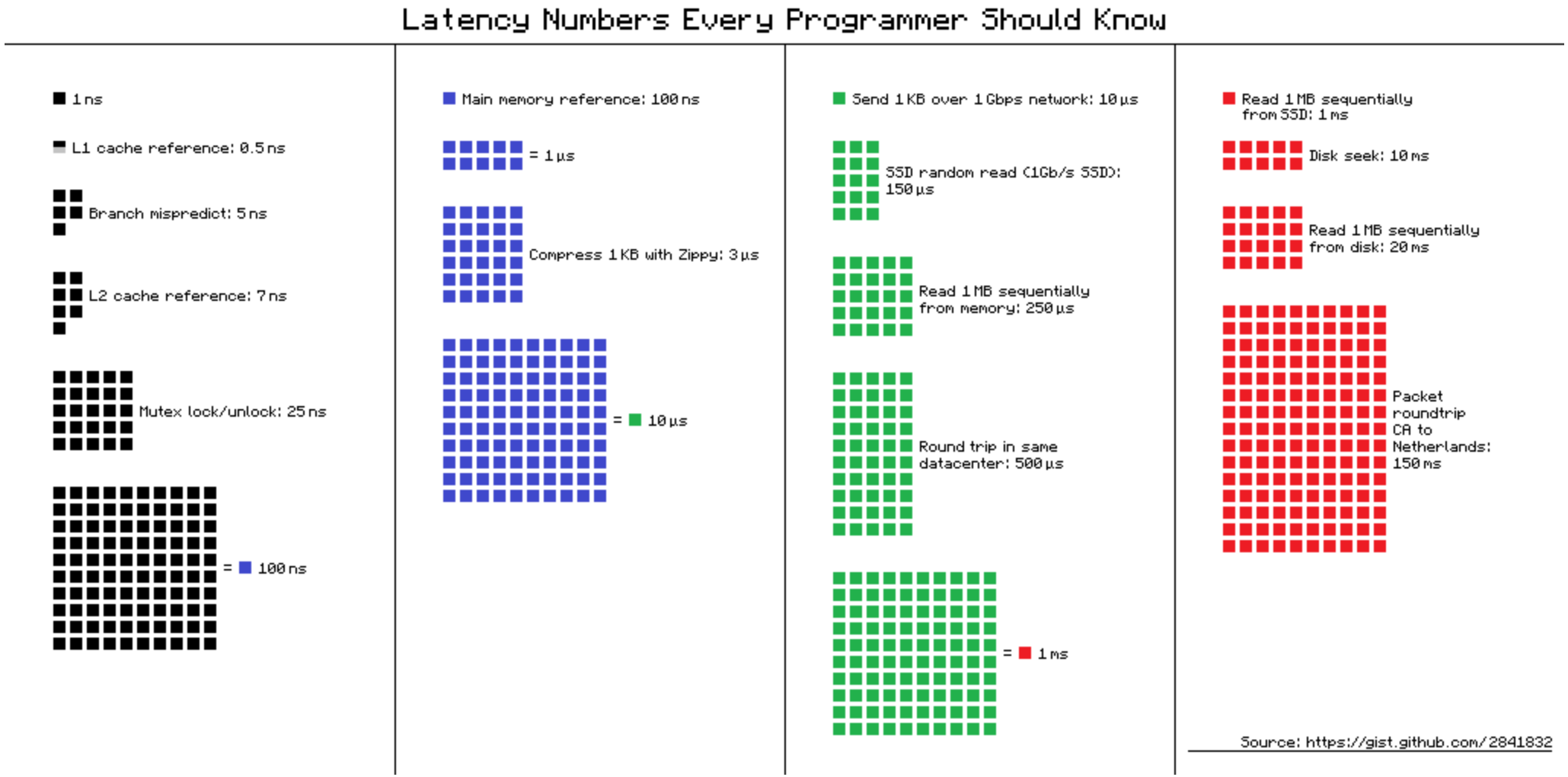
C.R.E.A.M. get the memory (dollar dollar bill ya'll)

Matthew D. Groves | Product Marketing Manager

aka “Passionate Bravo”

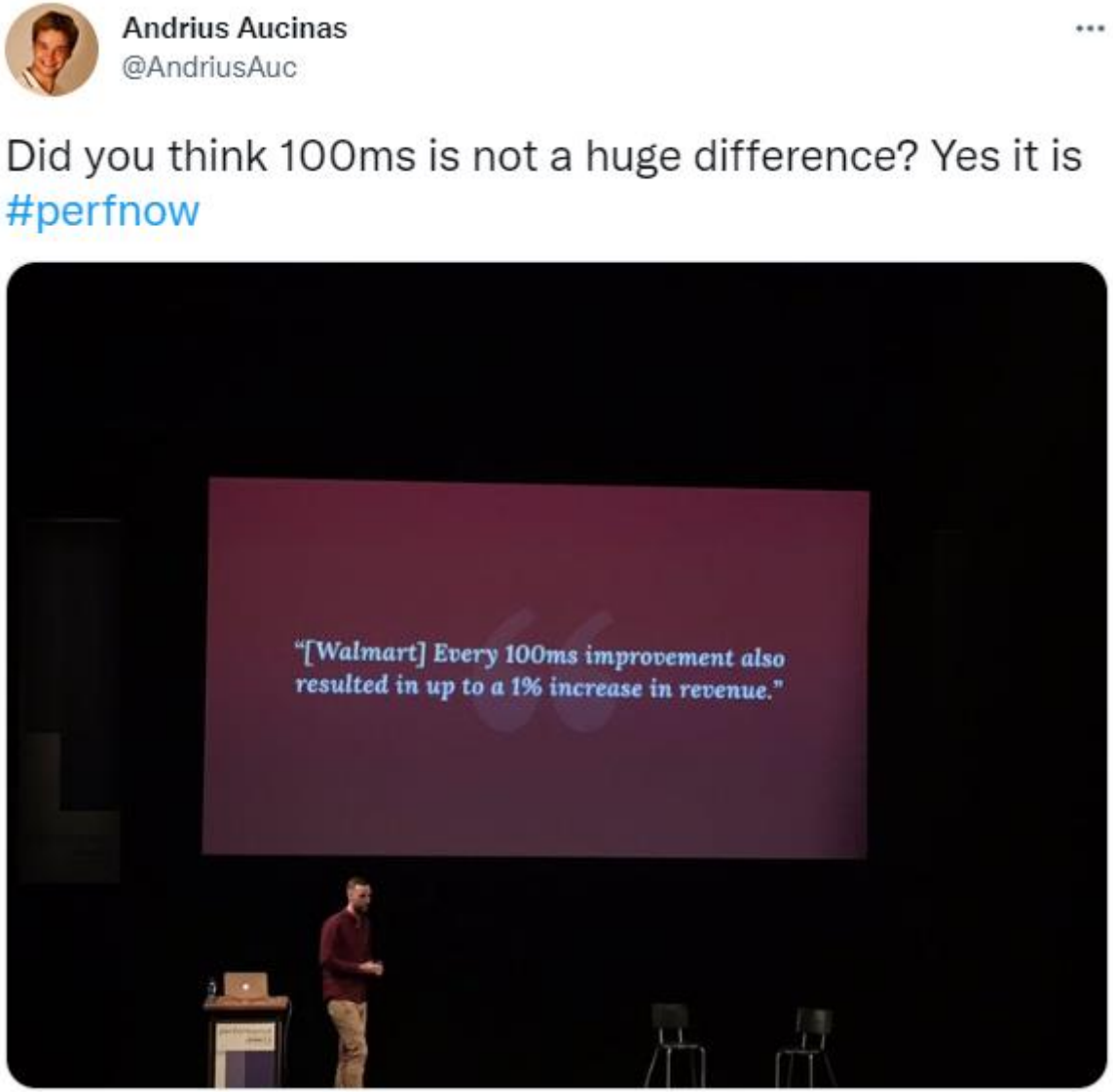
August 2022

Latency Numbers Every Programmer Should Know



https://stackoverflow.com/questions/4087280/approximate-cost-to-access-various-caches-and-main-memory/4087315#4087315

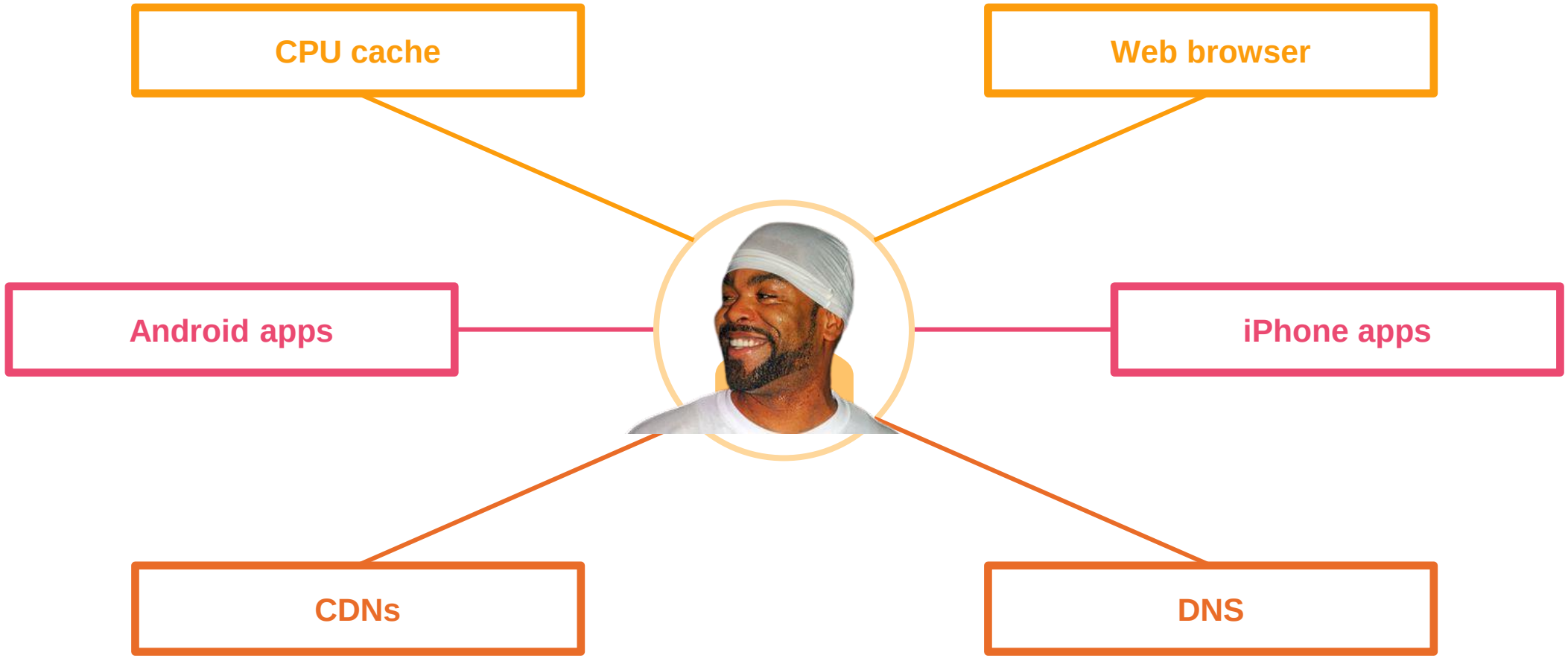
The Need for Speed



https://www.slideshare.net/devonauerswald/walmart-pagespeedslide

Cache Rules Everything Around Me

Dollar dollar bill ya'll



Agenda

01/ What is caching?

02/ Use Cases

03/ Reading from Cache

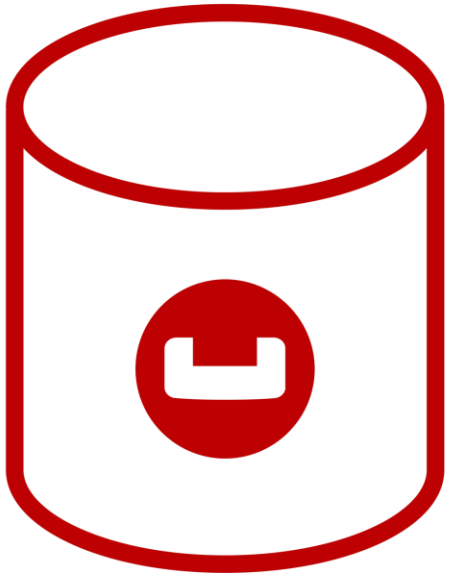
04/ Writing into Cache

05/ Cache Invalidation

06/ Gotchas

Who am I?

- Matthew D. Groves
- Microsoft MVP, author at Pluralsight, Manning, Apress
- Product Marketing Manager at Couchbase
- https://twitter.com/mgroves
- https://www.linkedin.com/in/mgroves/
- C# Advent: https://csadvent.christmas/



1 What is Caching?

Caching

What is Caching?

Collection of duplicated data to serve future requests faster.

- First formally described by Maurice Wilkes in 1965
- "Slave Memories and Dynamic Storage Allocation" https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=wilkes.pdf (https://bit.ly/WilkesPaper)

Records

Key	Data
1	Lorem ipsum dolor sit amet
2	**consectetur adipiscing elit**
3	sed do eiusmod tempor
4	**incididunt ut labore et dolore**
…	…

Cache

Key	Data
2	consectetur adipiscing elit
4	incididunt ut labore et dolore

Powerpoint as a Cache

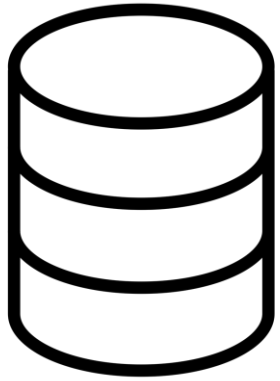
What is Caching?

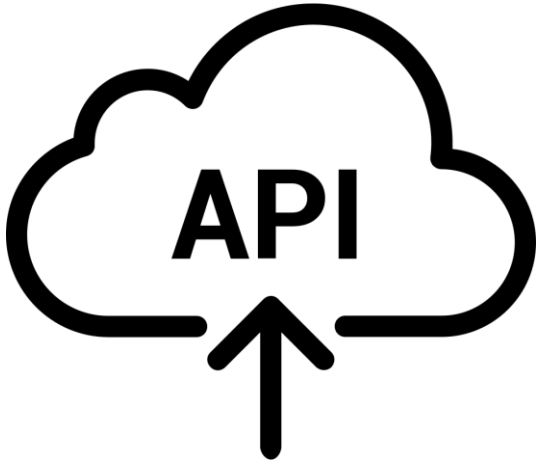
Key	Value

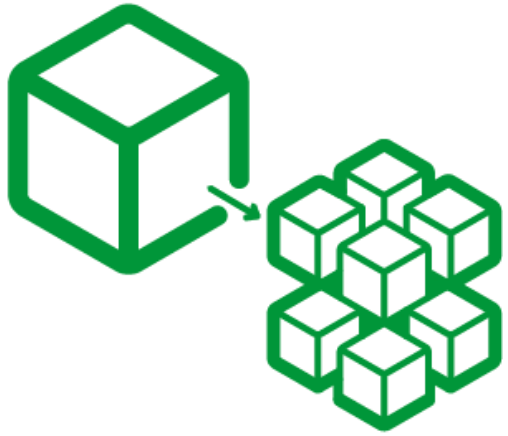
2 Caching Use Cases

Caching Use Cases

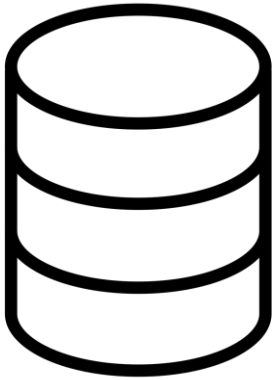
What is Caching used for?





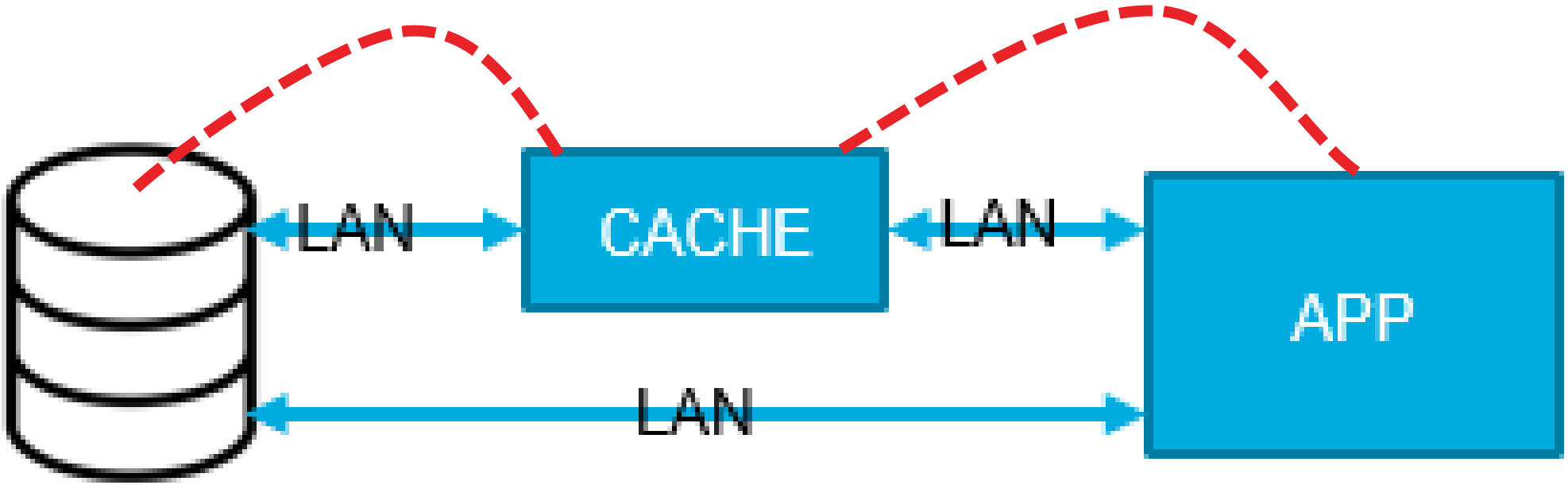


Caching Use Case 1: Faster Database Access



- Databases can rely heavily on disk
- Reading the same data over again could mean accessing the disk over again
- Store frequently accessed data in a cache, to reduce pressure on database/disk

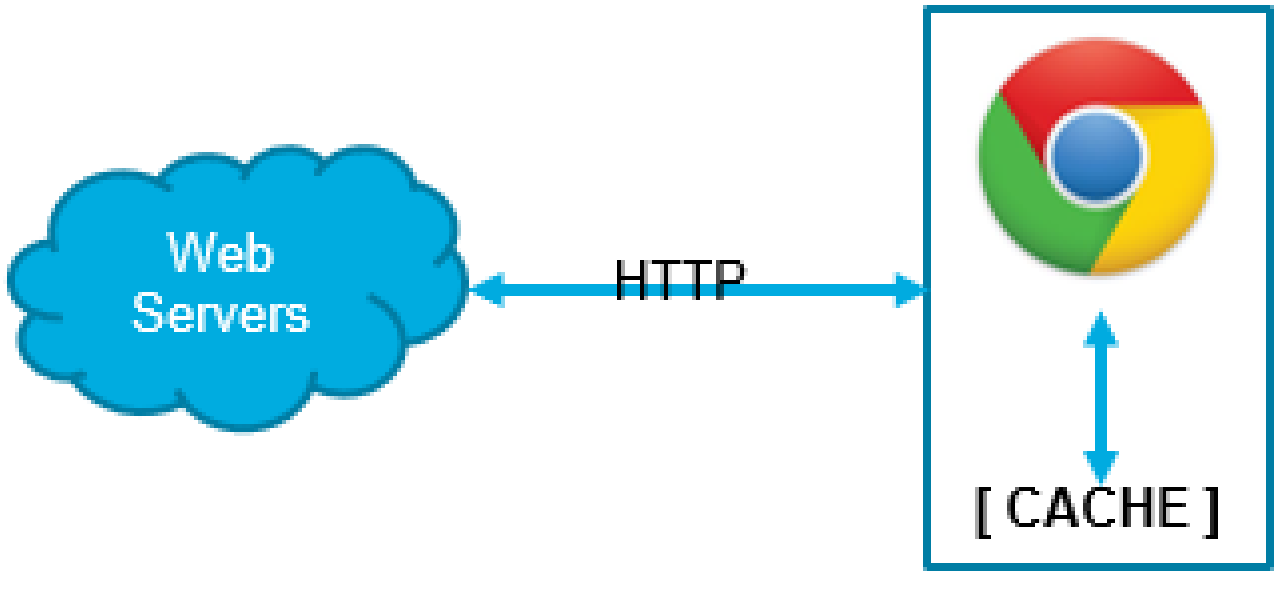
Caching Use Case 1: Faster Database Access



Caching Use Case 2: Faster Web Browsing

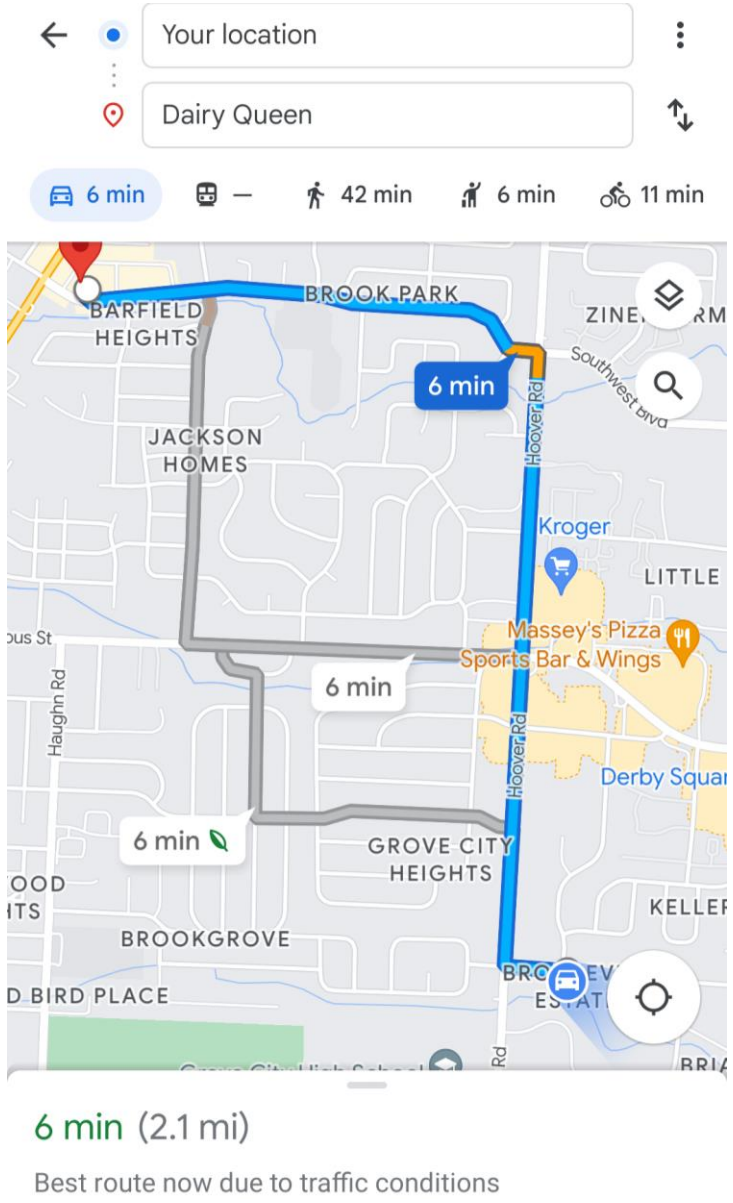
- Any given URL could result in hundreds of files that are loaded over HTTP
- A single JS file may be loaded by every page on a site.
- Cache static assets (JS, images, sound, icons, etc) on a user's laptop to avoid loading them over the network every time

Caching Use Case 2: Faster Web Browsing

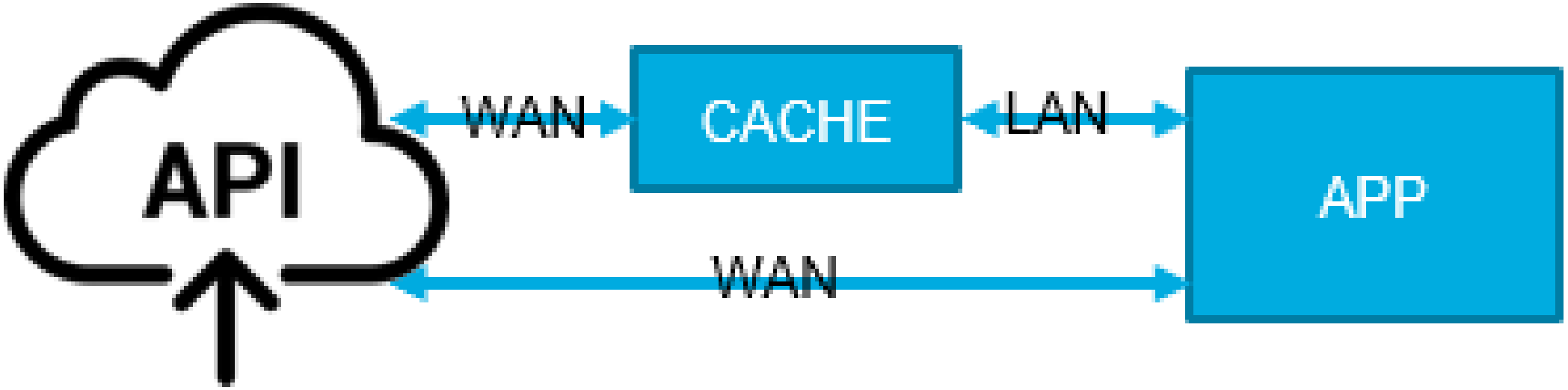


Caching Use Case 3: Efficient API Use

- API usage may involve an internet call (with latency)
- API calls may cost per use, or may be limited
- Storing results in cache may help reduce latency AND reduce costs

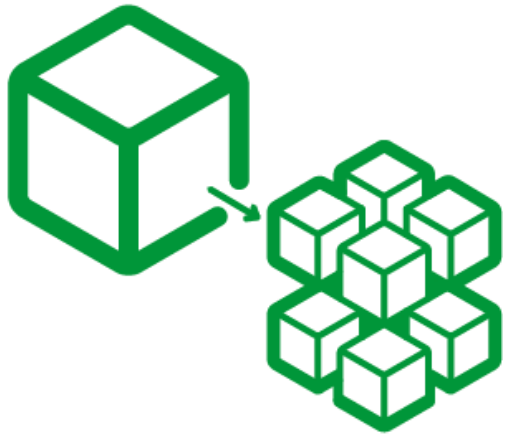


Caching Use Case 3: Efficient API Use

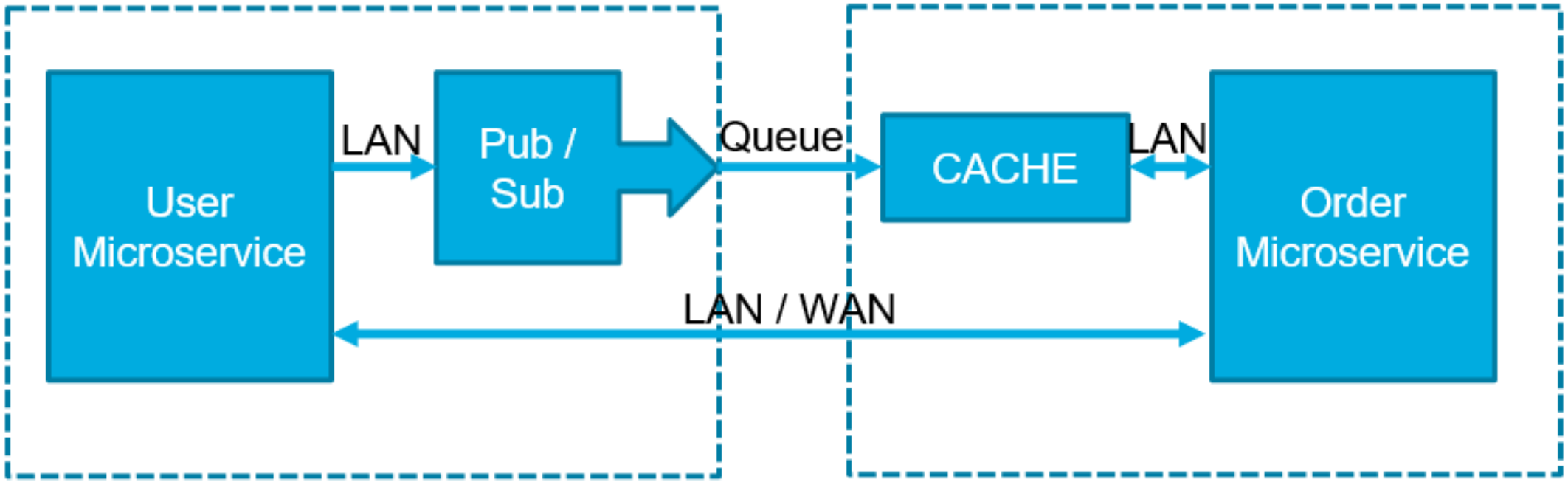


Caching Use Case 4: Maintaining Availability

- Retrieving data from other processes
- Microservices
- Speed is secondary concern

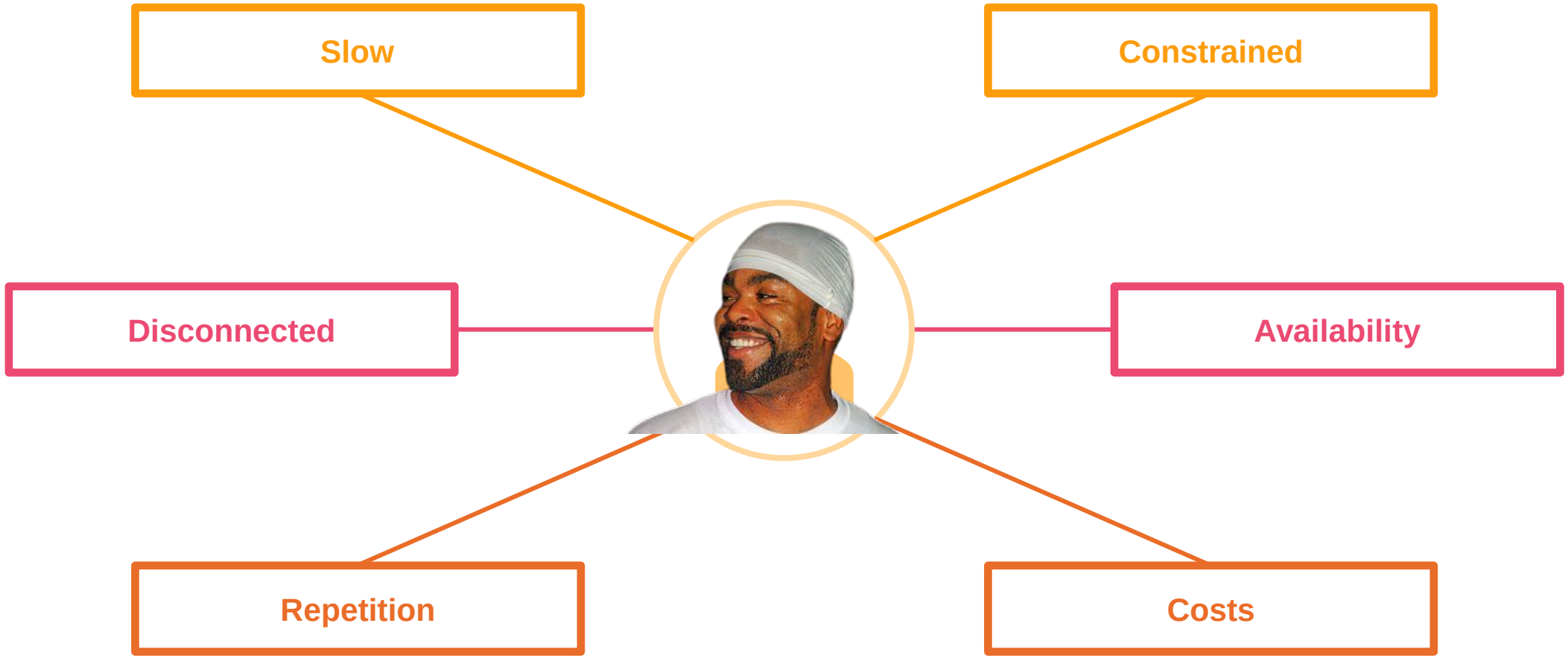


Caching Use Case 4: Maintaining Availability



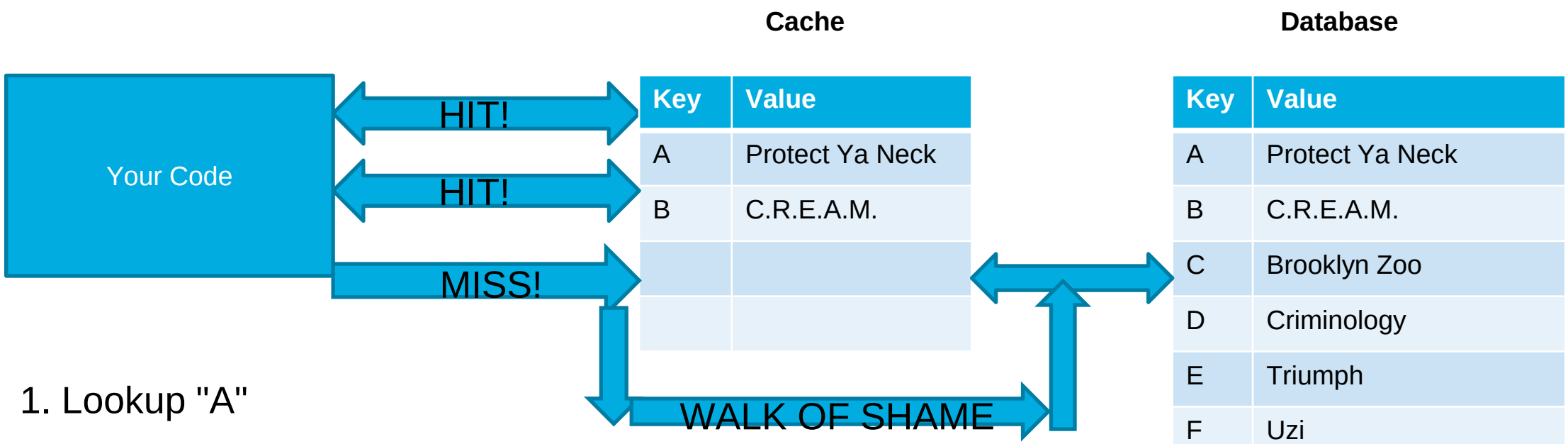
Cache Rules Everything Around Me

Dollar dollar bill ya'll



3 Reading From Cache

Using a Cache 101



1. Lookup "A"
2. Lookup "B"
3. Lookup "C"
4. Lookup "C" again?

4 Writing Into Cache

What about writes?

Stuff needs to get in the cache somehow

- Data is duplicated, stored in *cache* and *record*
- The *system* is responsible to keep the cache updated
- Commons methods:
  - write-through
  - write-around
  - write-back

write-through

1. System gets a new/updated piece of data
2. Write to cache and record "at the same time"
    1. Write to cache
    2. If that didn't succeed, the write failed.
    3. Write to record
    4. If that didn't succeed, the write failed.
    5. If the system reaches this point, the write succeeded!

Why or why not use this?

write-around

1. System gets a new/updated piece of data
2. Write the data to the record ONLY

Why or why not use this?

write-back

1. System gets a new/updated piece of data
2. Write the data to the cache.
3. Update the record asynchronously (don't wait on it to finish)
4. If the cache write succeeded, then consider the write succeeded.

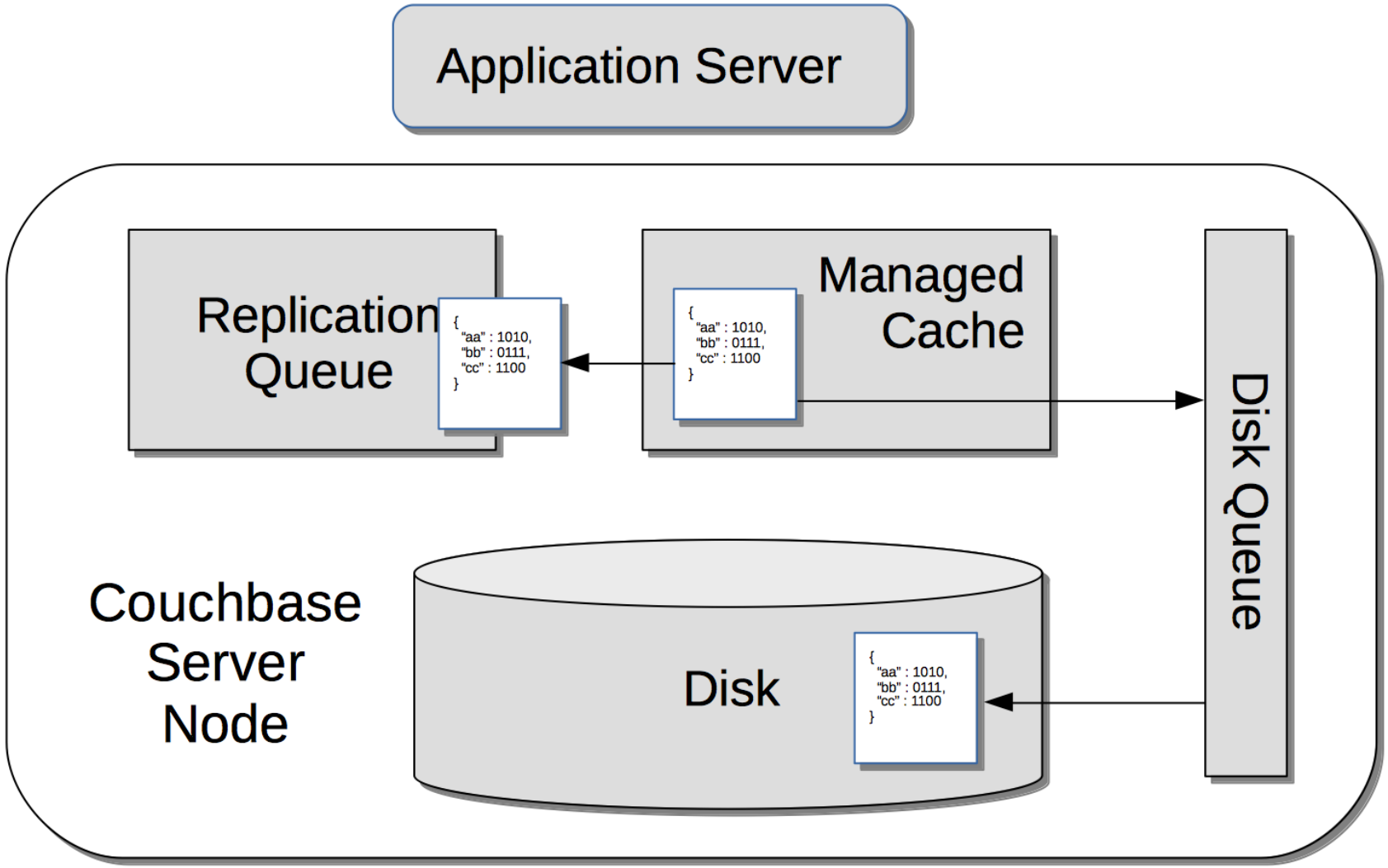
Why or why not use this?

Cache Writing Methods

Method	Write performance	Write integrity	Best for…?
write-through	—	+	High % reads
write-around	–	+	High % read later
write-back	+	–	Mixed reads / writes

Couchbase

A database with a built-in managed cache



https://docs.couchbase.com/server/current/learn/buckets-memory-and-storage/memory-and-storage.html

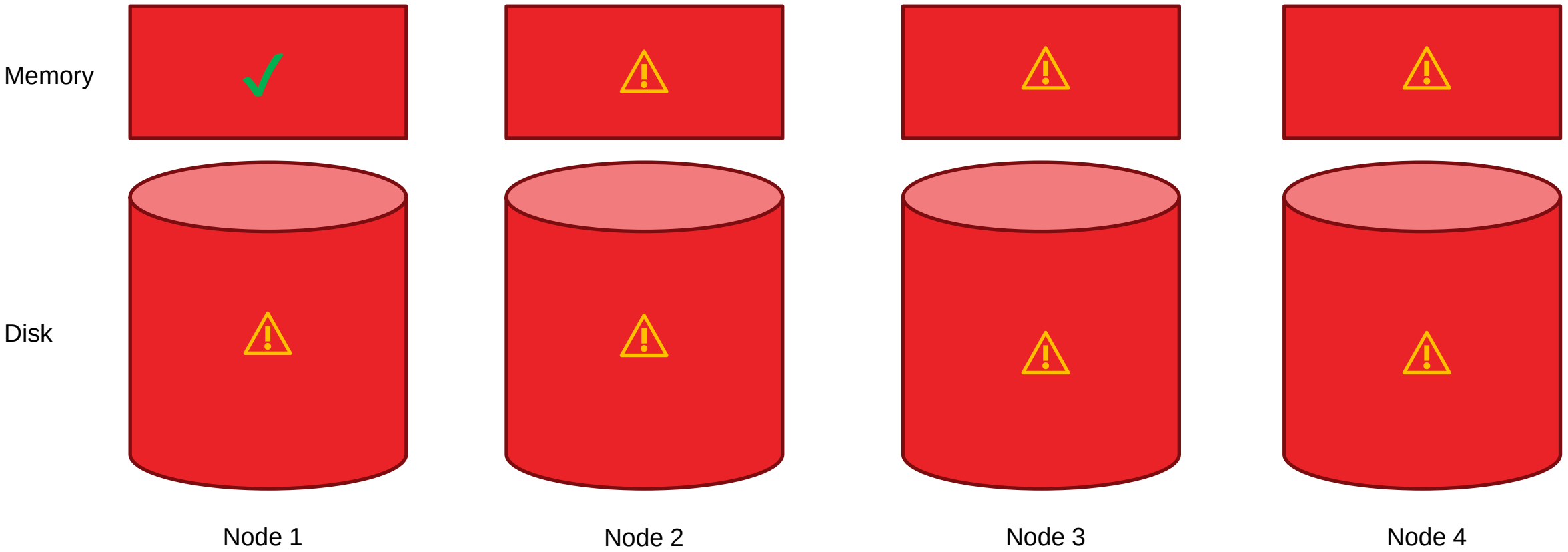
Couchbase and Writing to the Cache

```
await collection.InsertAsync("A", new { title = "Protect Ya Neck" });
```

```
await collection.InsertAsync("A", new { title = "Protect Ya Neck" }, options =>
{
    options.Durability(DurabilityLevel.None);
    // OR: options.Durability(DurabilityLevel.Majority);
    // OR: options.Durability(DurabilityLevel.MajorityAndPersistToActive);
    // OR: options.Durability(DurabilityLevel.PersistToMajority);
});
```

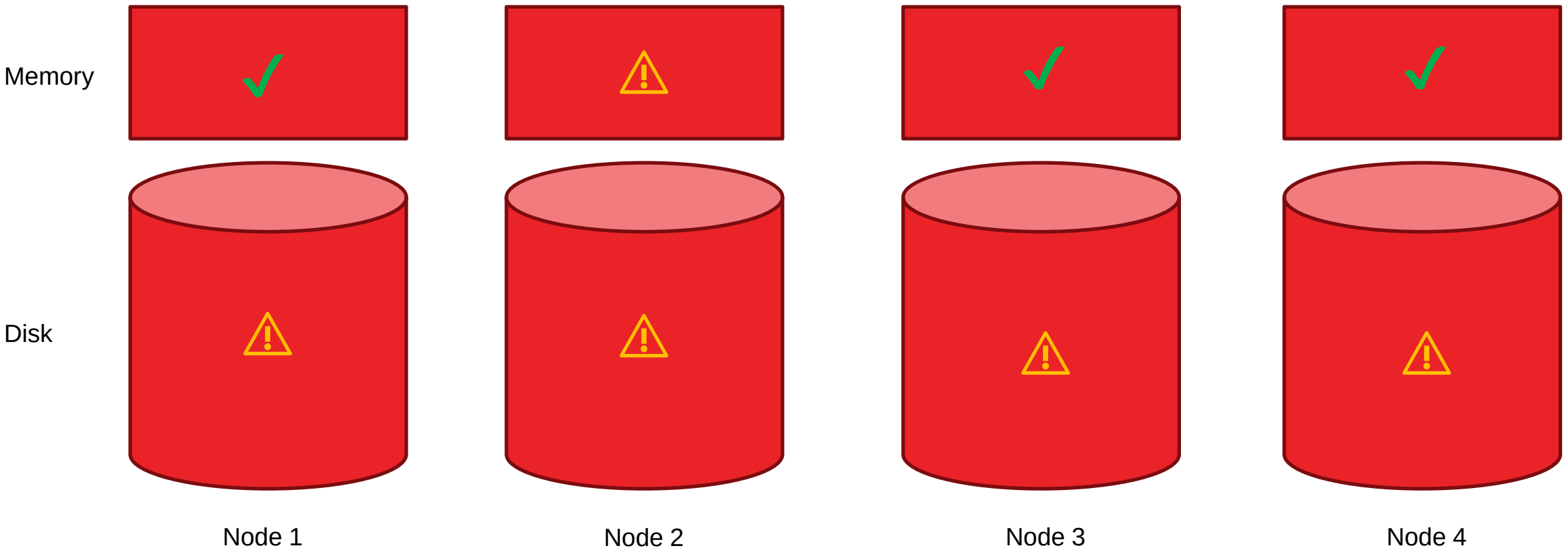
Durability

None



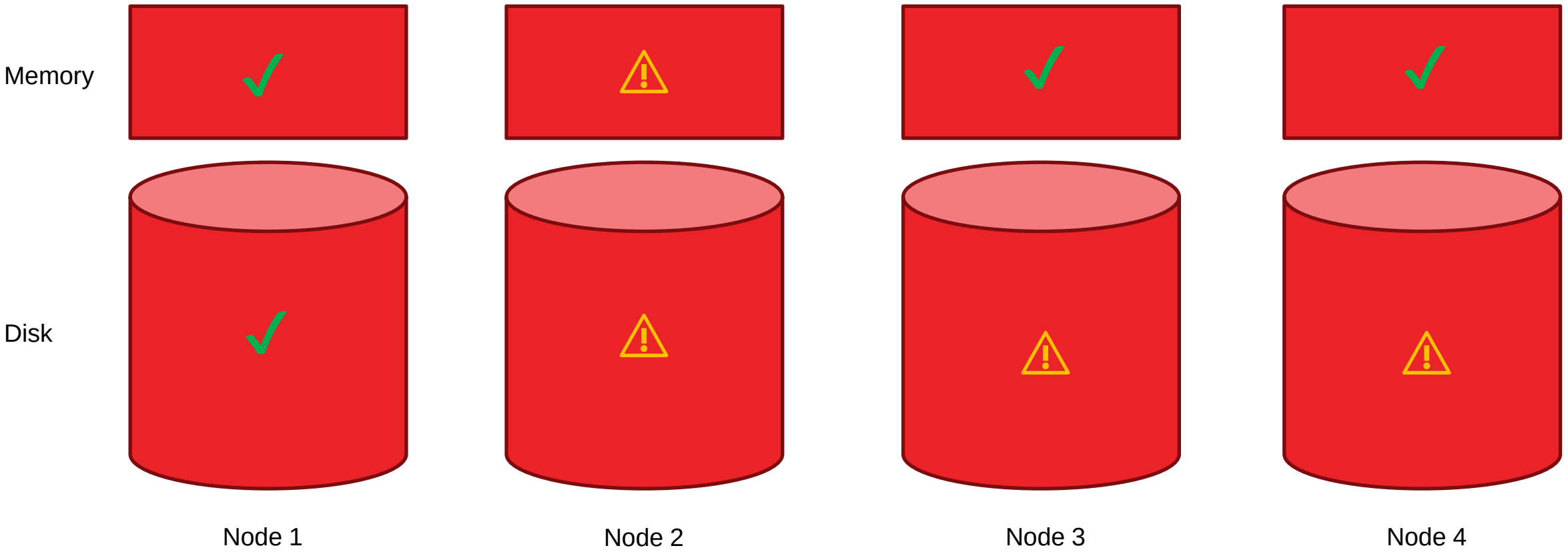
Durability

Majority



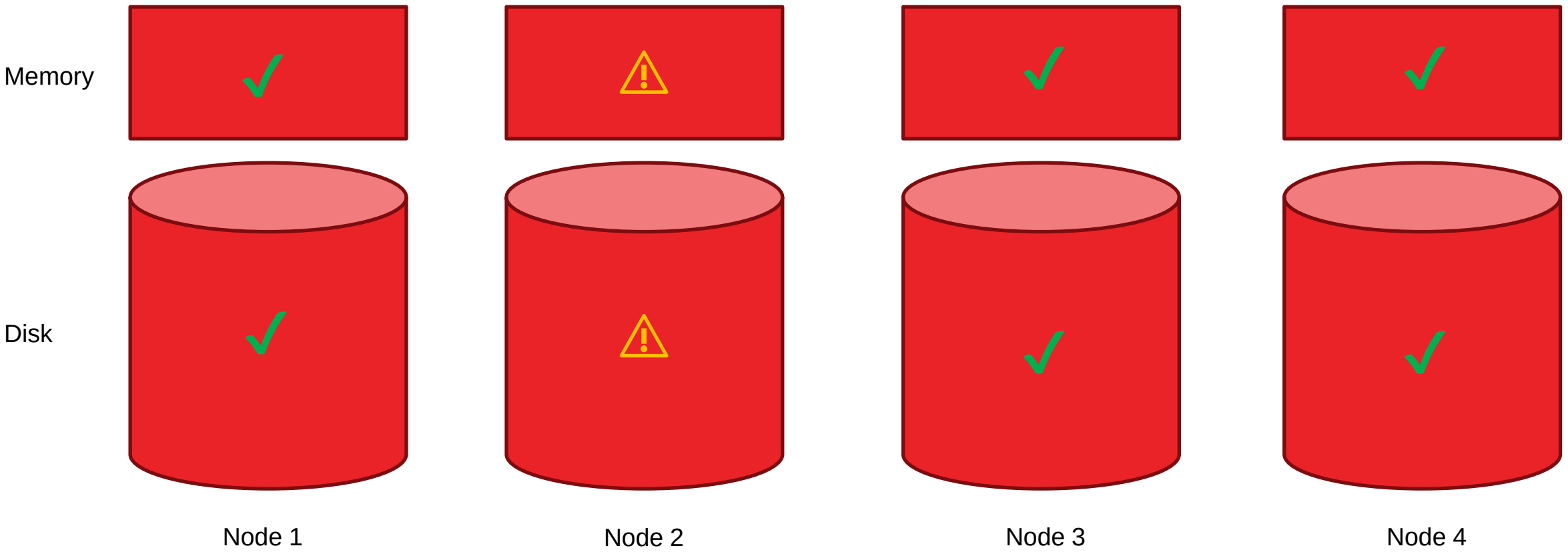
Durability

MajorityAndPersistToActive

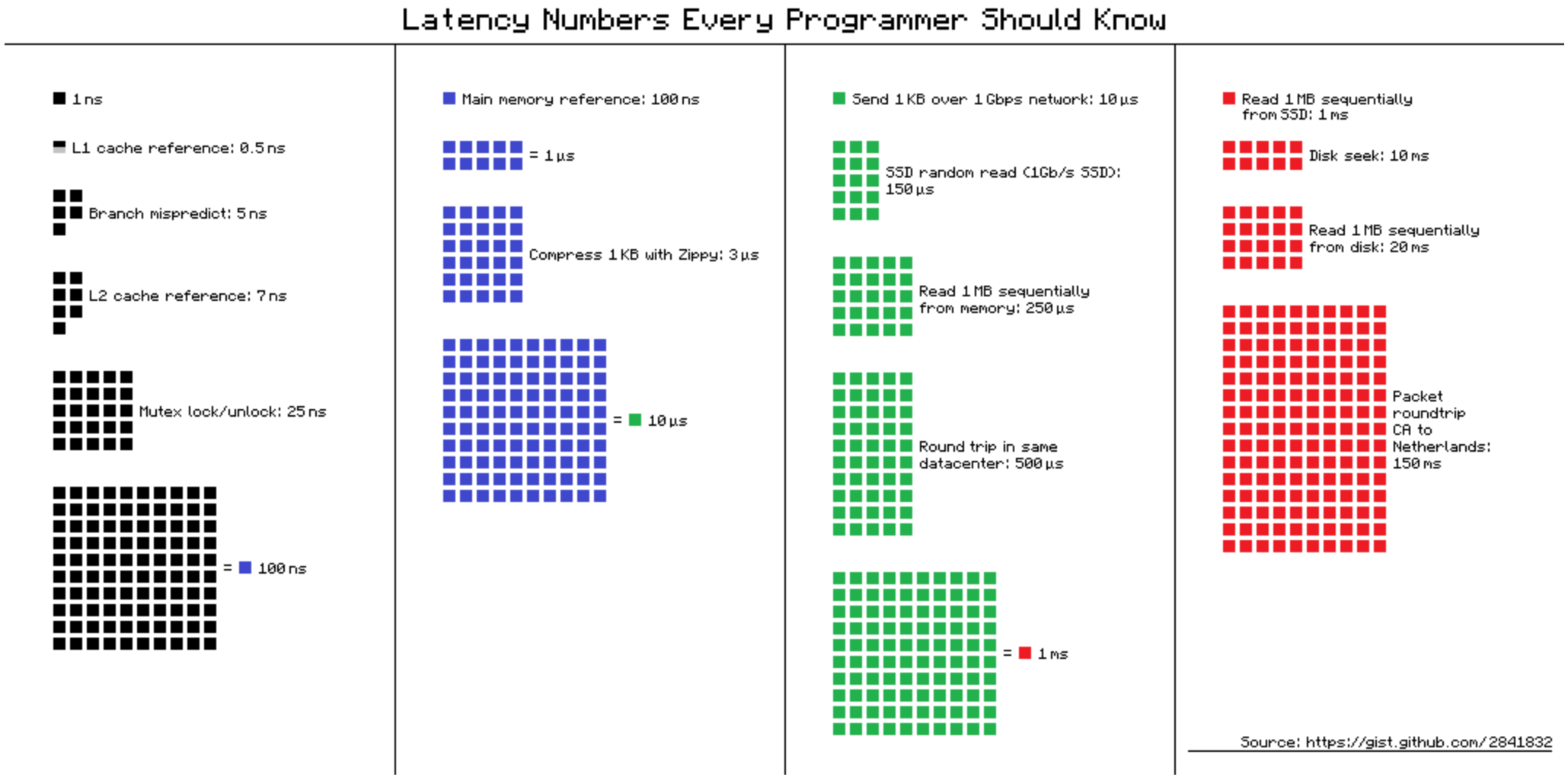


Durability

PersistToMajority



Latency Numbers Every Programmer Should Know



https://stackoverflow.com/questions/4087280/approximate-cost-to-access-various-caches-and-main-memory/4087315#4087315

5 Cache Invalidation

Powerpoint as a Cache

Key	Value
What is 30,795,738 divided by 42,891?	718

Cache Invalidation is Hard

There are only two hard things in Computer Science: ***cache invalidation*** and naming things.

-- Phil Karlton

https://www.nndb.com/people/400/000031307/

How to decide what to invalidate (what data to "evict"):

cache policy

Cache Policy 1: LRU (Least Recently Used)

Cache max size: 3
Order of reads: A, B, C, B, D

Key	How new?
	X

Cache Policy 2: NRU (Not Recently Used)

Lower scores evicted (at random)

Key	Referenced?	Modified?	Score
A	0	0	0
B	0	1	1
C	1	0	2
D	1	1	3

Cache Policy 3: None

Don't evict anything

More Cache Policies

- **RR –** random replacement
- **FIFO and LIFO –** treat a cache like a queue
- **MRU –** MOST recently used
- **LFU –** least frequently used
- **Belady's algorithm –** not possible to implement
- **Machine learning**
- https://en.wikipedia.org/wiki/Cache_replacement_policies

6 Gotchas

Gotcha 1: Cache Size

- **Problem**: too small or too big cache
- **Symptom**: High turnover and churn (many evictions) or long lifetimes (few evictions)
- **Solutions**:
  - Monitoring
  - Size the cache
  - "Value" vs "Full" eviction

Gotcha 2: Bad Eviction Policy

- **Problem**: using the wrong eviction policy
- **Symptom**: lots of overhead, cache checking
- **Solutions**:
  - Don't Write an Eviction Policy (or better yet don't write a Cache)
  - Benchmarking: "hit ratio" and "latency"
    - "Hit ratio" – how often a hit vs miss (e.g. "45% hit ratio")
    - "Latency" – how long it takes from request to response (e.g. 10µs latency)

Gotcha 3: "Close" Caching

- **Problem**: running the cache on the same machine as the application
- **Symptom**: redundancy, hot spots, inconsistency, availability
- **Solutions**:
  - Use a distributed cache
  - Running on its own machine(s) / cluster
  - Couchbase, of course!

References

- https://www.geeksforgeeks.org/not-recently-used-nru-page-replacement-algorithm/
- https://shahriar.svbtle.com/Understanding-writethrough-writearound-and-writeback-caching-with-python
- https://en.wikipedia.org/wiki/Maurice_Wilkes
- https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=wilkes.pdf
- https://stackoverflow.com/a/4087315/40015
- https://www.techtarget.com/searchstorage/definition/cache
- https://en.wikipedia.org/wiki/Cache_replacement_policies
- https://docs.couchbase.com/dotnet-sdk/current/howtos/kv-operations.html#durability
- https://docs.couchbase.com/dotnet-sdk/current/concept-docs/durability-replication-failure-considerations.html#durable-writes