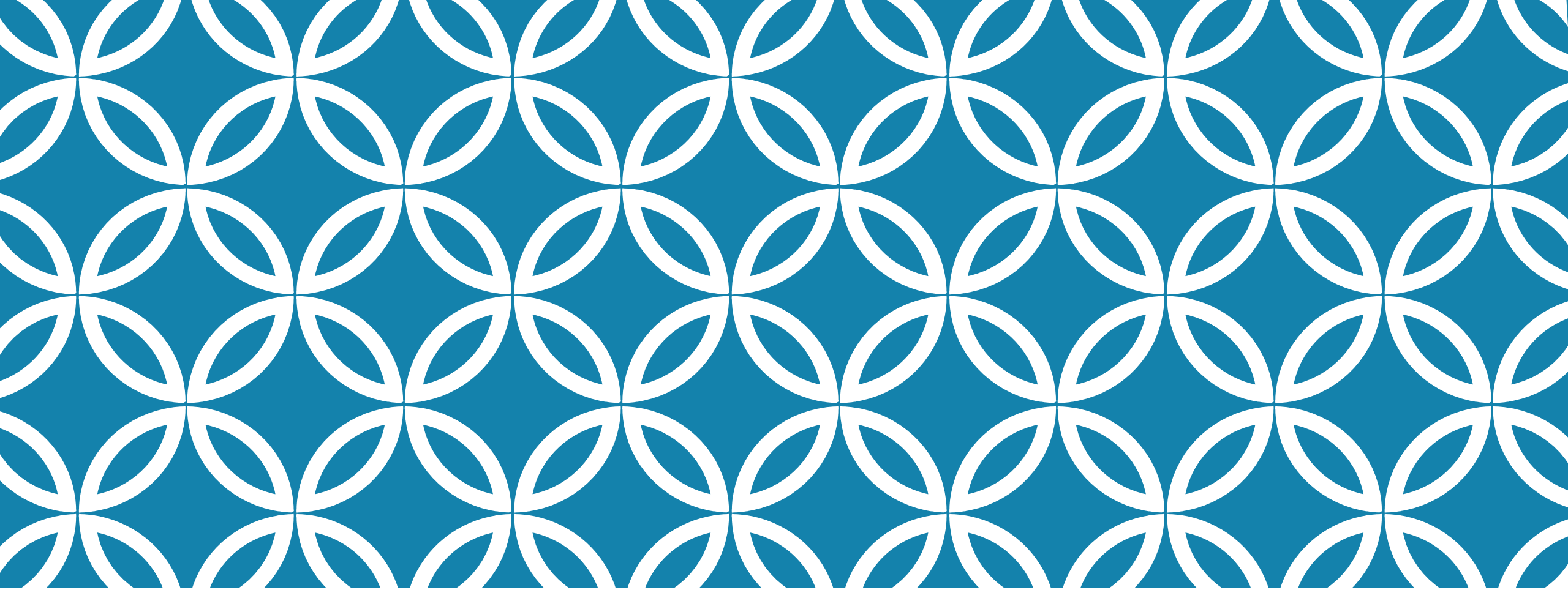


CS4414: RECITATION 12 – MEMCACHED (MOTIVATION AND USE)

Ricky Takkar

Friday, April 21, 2023

BOTTOM LINE SUMMARY (LECTURE 22: PREFETCHING AND CACHING)

A computer has a lot of capacity to do things concurrently.

Prefetching or preloading files is a huge win:

- The costs of data access aren't eliminated, but are mostly hidden
- The work of prefetching/preloading is often mostly in hardware
- We own the hardware… why not keep it busy?
- Tremendous variety of examples where the same basic ideas are employed for many different purposes.

A REAL WORLD NEED

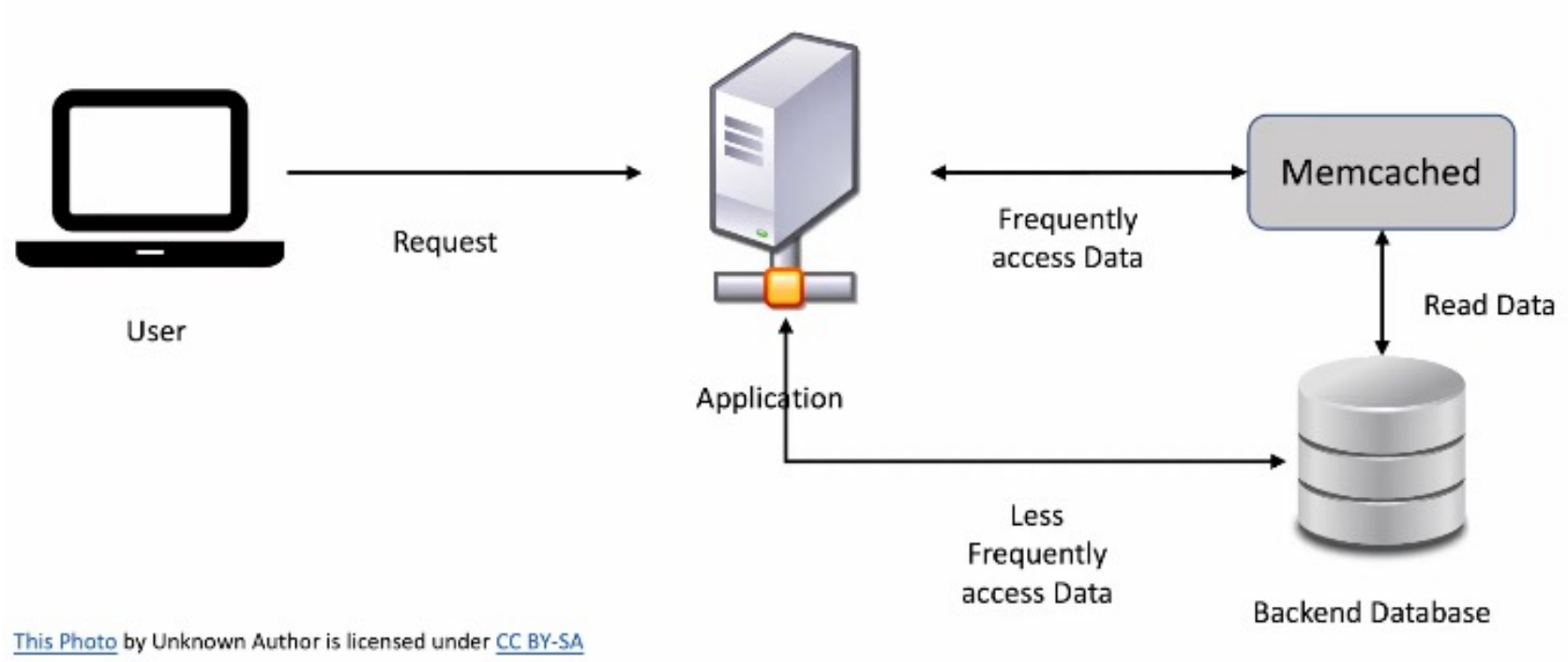
- Once upon a time, before Facebook, there was LiveJournal – a community-based journaling platform (OG social media) built by Brad Fitzpatrick's company, Danga Interactive (1998 ~ 2007)

- Memcached was first developed by Brad Fitzpatrick for his website LiveJournal, on May 22, 2003.[5][6] It was originally written in Perl, then later rewritten in C by Anatoly Vorobey, then employed by LiveJournal.[7] Memcached is now used by many other systems, including YouTube,[8] Reddit,[9] Facebook,[10][11] Pinterest,[12][13] Twitter,[14] Wikipedia,[15] and Method Studios.[16] Google App Engine, Google Cloud Platform, Microsoft Azure, IBM Bluemix and Amazon Web Services also offer a Memcached service through an API.[17][18][19][20]

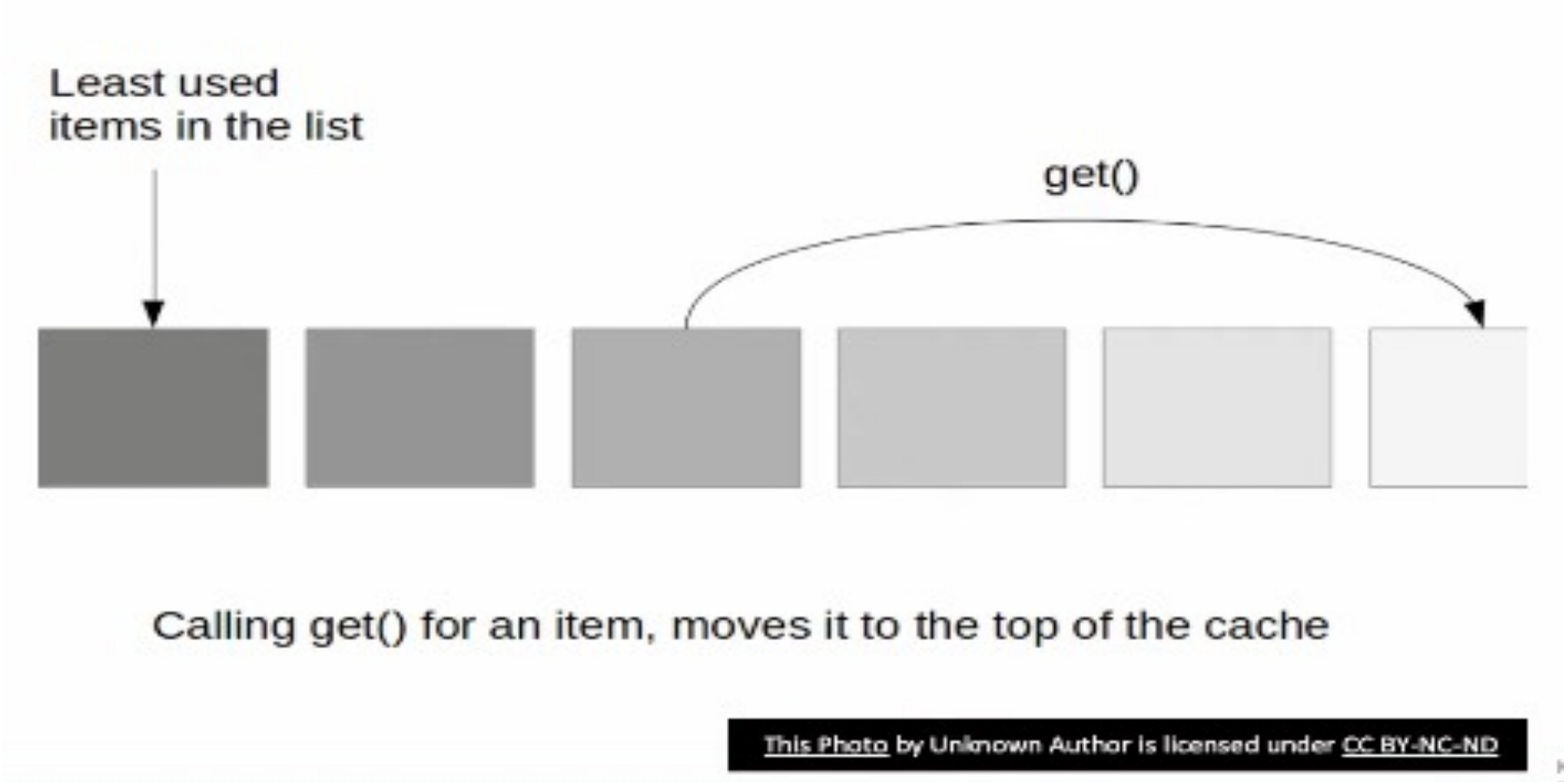
https://en.wikipedia.org/wiki/Memcached

MOTIVATION

- Reduce load on backend DB
-



LEAST RECENTLY USED (LRU) CACHE



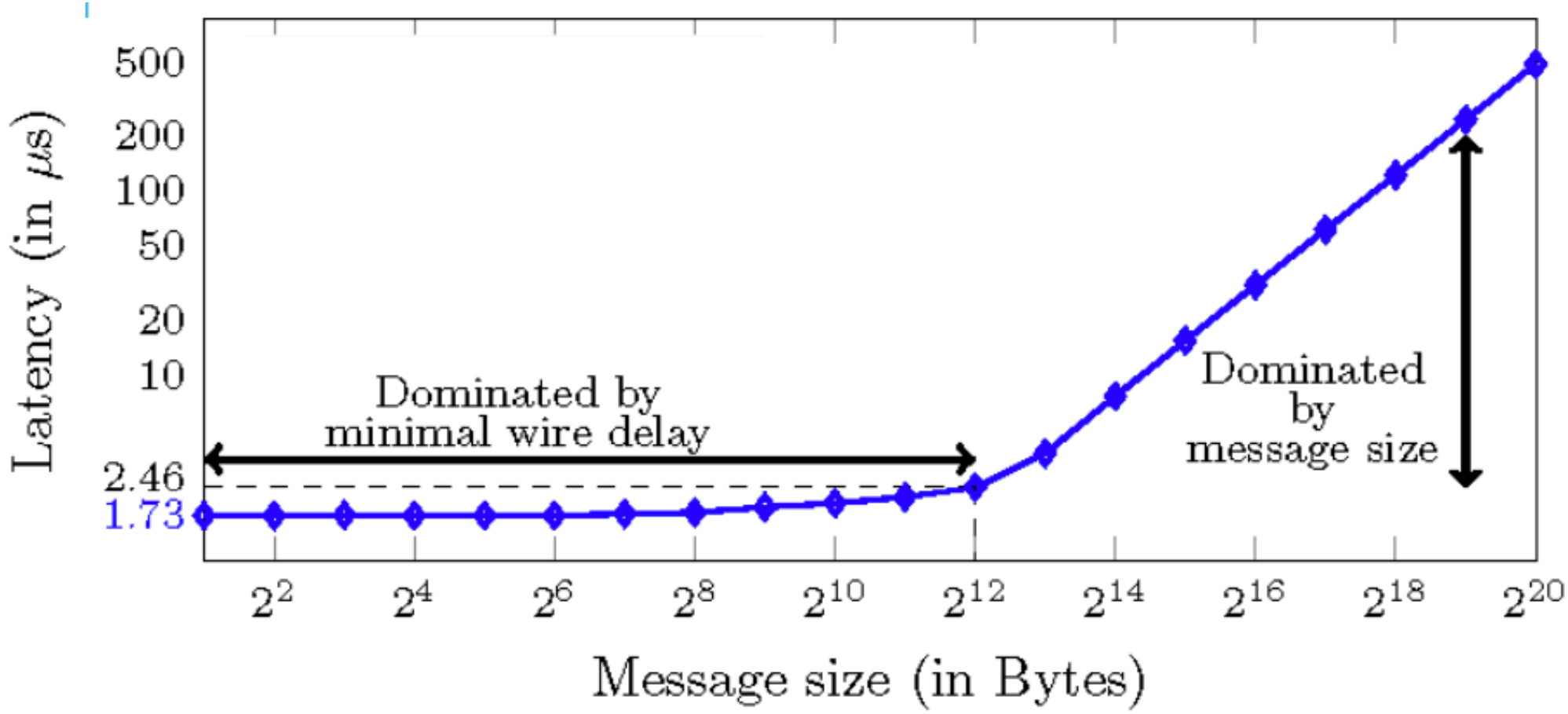
WAIT… WHY NOT JUST STORE ALL DATA INSIDE MEMCACHED?

For starters: Memcached is a cache system, *not* a storage system

While it provides fast access to *frequently* accessed data, it is not built for:

- Persistent storage (cache is "alive" *only* as long as system is ⏰)
- Updating data
- **Large amounts of "tiny" objects**

WHY ARE YOU AGAINST STORING MANY TINY OBJECTS IN MEMCACHED?



MEMCACHED TARGETS

- Data access operations in O(1) scale
- Run queries in < 1ms
- "High end" servers can serve millions of keys/second
- Performance also benefits from:
  - Memcached servers being independent of one another
    - 0 overhead related to consensus

MEMCACHED CONCEPT (MEMORY CACHE DAEMON) – REVIEW (LECTURE 23)

The (entire!) API of MemCacheD:

```
MemCacheD::put(string key, object value)

object = MemCacheD::get(key)
```

Put saves a copy of the pair (key,value), replacing prior value.

Get will fetch the object, if it can be found.

STORAGE COMMANDS

set, *add*, *replace*, *append*, *prepend* and *CAS (Check-And-Set or Compare-And-Swap)*

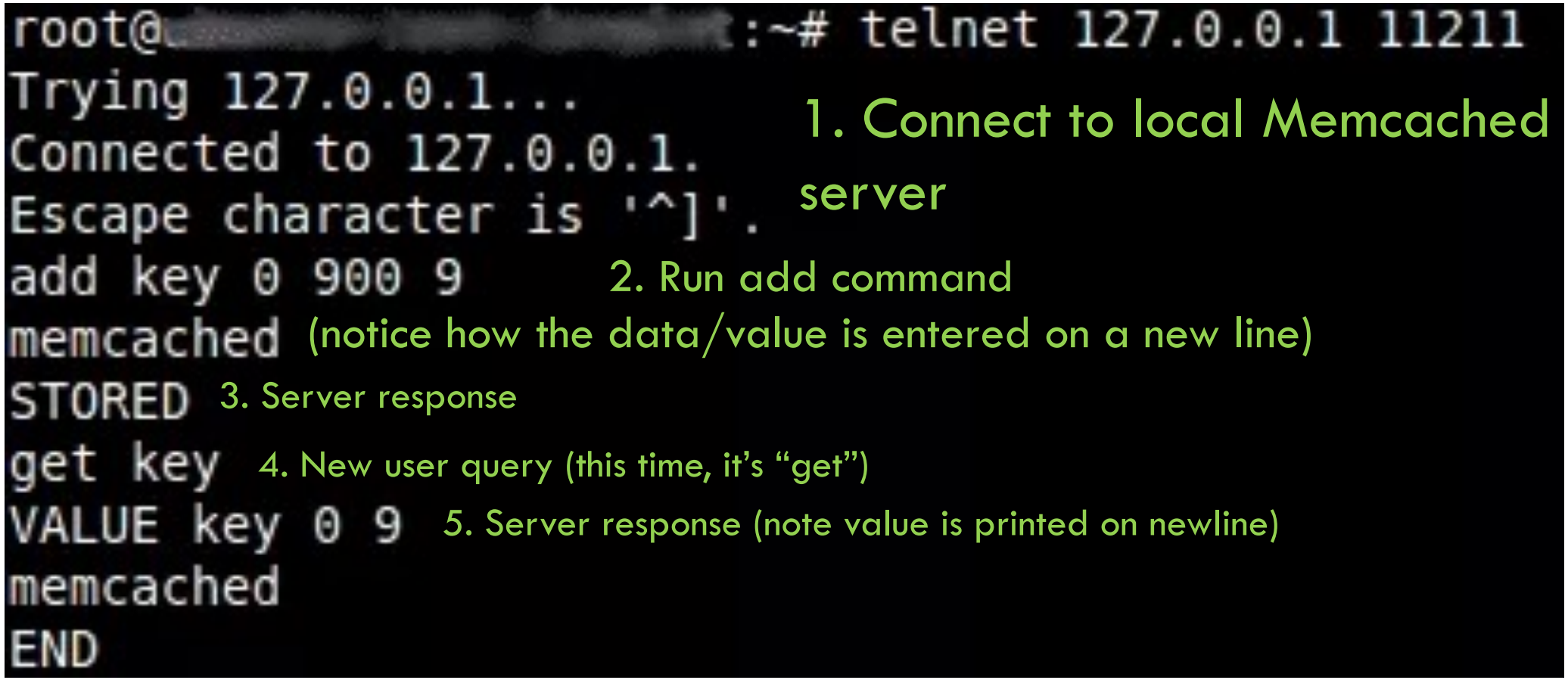
Sample add command: `add key flags exptime bytes [noreply]`

Parameters:

- **key** – This is the key name by which we can store and retrieve data from Memcached.
- **flags** – This is a 32-bit unsigned integer. The server stores this flag with the data set by the user. Also, it returns the flag when we retrieve the data.
- **exptime** – It is the *expiration time* in seconds.
- **bytes** – The number of bytes in the data block. Usually, this is the length of the data that we store in Memcached.
- **noreply** – This optional parameter informs the server not to send any reply.
- – It is the data we want to store. Always enter the data on a new line after executing the set command.

https://bobcares.com/blog/memcached-expiration-time/

THAT ADD COMMAND IN ACTION



https://bobcares.com/blog/memcached-expiration-time/

WHERE DOES MEMCACHED IMPLEMENTATION LOGIC LIVE

- Distributed between client/server
  - Clients typically know which server to access for fetching data
- Servers live a *simple* life
  - They just return data therein and keep cache "fresh" (LRU)

SO HOW DOES MEMCACHED WORK IN A DISTRIBUTED (REAL-WORLD) SETTING

See Lecture 23: slides 27-35

https://www.cs.cornell.edu/courses/cs4414/2023sp/Slides/23-MemCacheD.pdf

SOUNDS GOOD. BUT HOW DO WE INSTRUMENT THESE IDEAS? DEMO!

1. Download from memcached.org

2. Let's code! (last demo of the semester)

3. See code zip on course website after recitation for step-by-step instructions