

Techniques for Scaling

Techniques for scaling

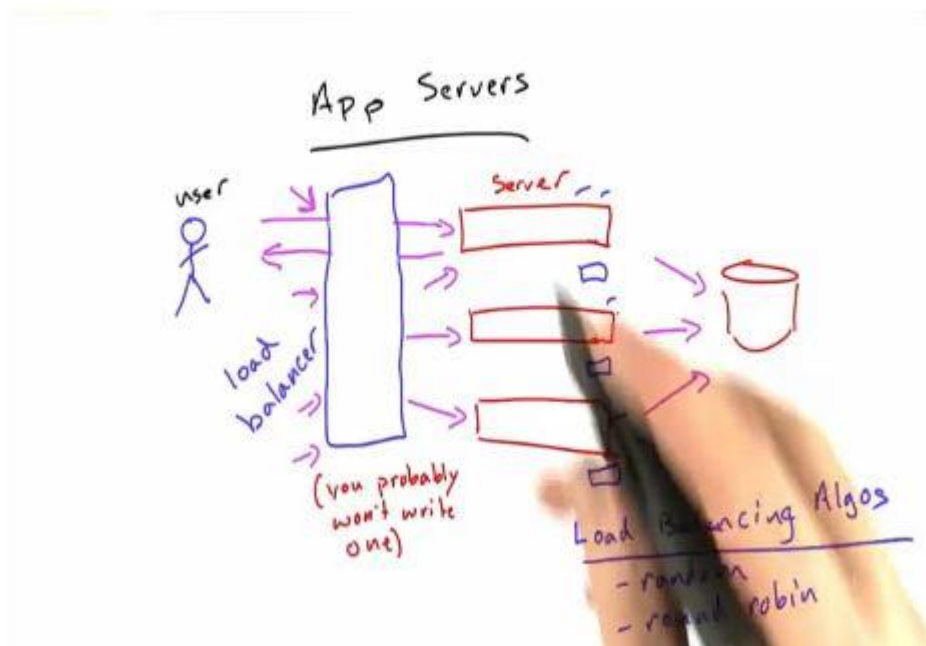
- optimize code
- cache complex operations
- upgrade machines
- add more machines

what do we scale?

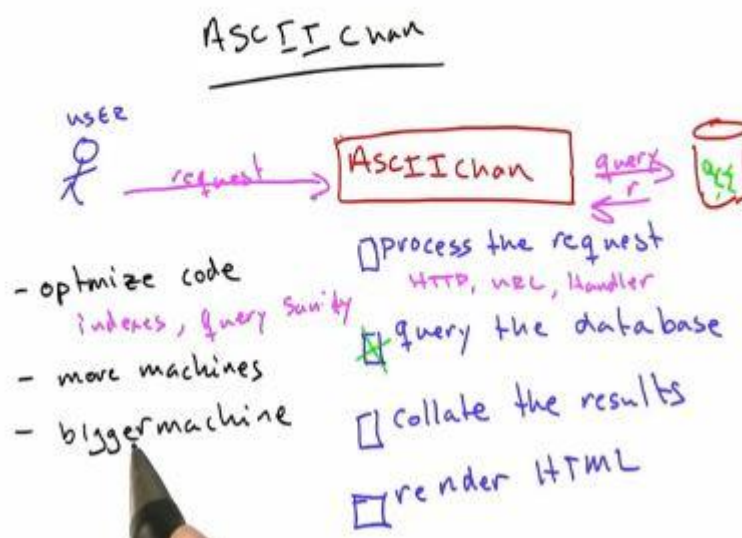
- ☒ bandwidth
- ☒ computers (memory, CPU)
- ☒ power
- ☒ storage



App Server Scaling



Optimizing Queries



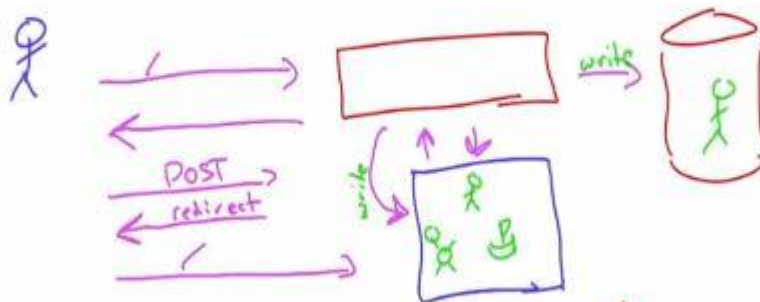
Quiz

why is our dictionary cache problematic with multiple app servers?

- ☐ it's not! this is a trick question
- ☒ multiple app servers = multiple caches
how do we keep them in sync?
- ☒ each app server may have to hit the db to update its cache
- ☐ we'll be caching data redundantly

Caching Techniques

cache updating

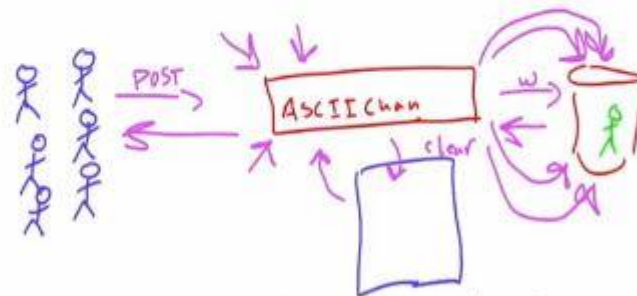


complex inserts + speed
vs
DB reads

the more accurate the
cache, the more
complex the code

Cache Stampede

Cache Stampede



→ when multiple cache misses
create too much load on
the database

Quiz

How can we avoid a cache stampede?

- ☐ replicate the db to handle more reads
- ☐ only allow one web request at a time
- ☐ only allow one db request at a time
- ☒ don't clear the cache, but instead overwrite with new data

Quiz

why do we separate our services?

- ☒ so they can be scaled independently
- ☒ to increase fault tolerance
- ☒ so two very different processes aren't competing for resources
- ☒ so they can be updated independently

Caching Techniques

Caching Techniques

<u>Approach</u>	<u>DB read / page view</u>	<u>DB read / submit</u>	<u>Bugs</u>
no caching	every	none	
naive caching	cache miss	none	yes
clear cache	cache miss	none	
refresh cache	(rarely)	1	

simple users shouldn't touch database

Quiz

What happens when you store more data in memcached than there is memory available?

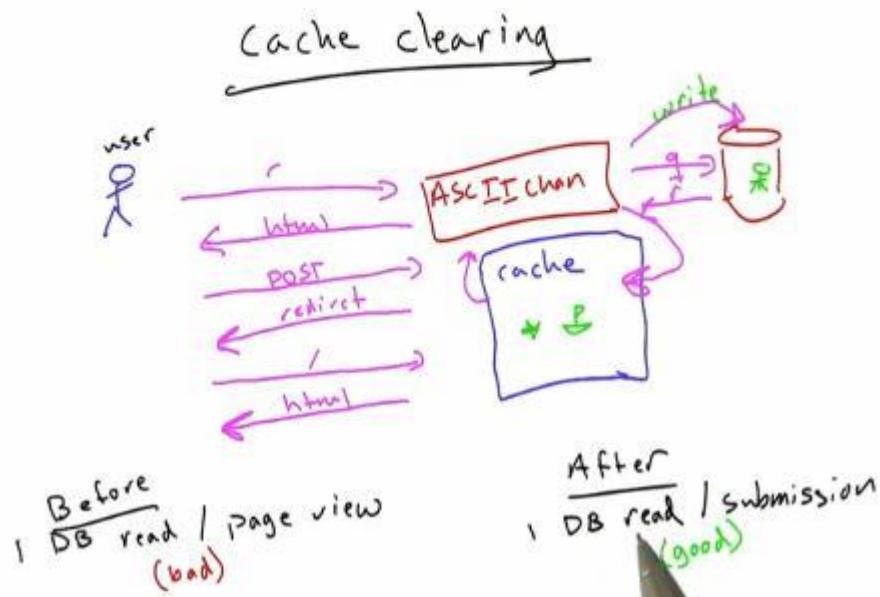
- ☐ error
- ☐ throw away data that is least frequently used **LFU**
- ☒ throw away data that is least recently used **LRU**
- ☐ write the extra data to disk

Quiz

How can we fix our stale cache problem?

- ☐ improve the cache to automatically expire after some time
- ☒ after submitting, clear the cache
- ☒ after submitting, update the cache
- ☐ don't cache, find a different approach

Cache Clearing



Caching

cache is a hashtable

```
if request in cache:
    return cache[request]
else:
    r = db-read() (100ms)
    cache[request] = r
    return r
```

cache hit

cache miss

why do we scale?

- ✓ so we can serve more requests concurrently
- ✓ so we can store more data
- ✓ so we're more resilient to failure
- ✓ so we can serve requests faster

Caching

Caching

caching refers to storing the result of an operation so that future requests return faster

when do we cache?

- computation is slow
 - computation will run multiple times
 - when the output is the same for a particular input
- you hosting provider charges for db access

```
5 # database, or a request to another web service.
6 def complex_computation(a, b):
7     time.sleep(5)
8     return a + b
9
10
11
12 # QUIZ - Improve the cached_computation() function below so that it caches
13 # results after computing them for the first time so future calls are faster
14 cache = {}
15 def cached_computation(a, b):
16     key = (a,b)
17     if key in cache:
18         r = cache[key]
19     else:
20         r = complex_computation(a, b)
21         cache[key] = r
22     return r
23
24 start_time = time.time()
25 print cached_computation(5, 3)
26 print "the first computation took %f seconds" % (time.time() - start_time)
27
28
29
```

RUN

```
0
the first computation took 0.500195 seconds
```


Broken Submissions

