

# Near-Far Cache Architecture





Near-far cache architecture is a robust architecture for deploying cache tier for web applications with following features.

1. Improve Performance
2. Protect data tier and do it reliably
3. Cache consistency

Deploying memcached on few nodes and keeping round trip time (RTT) from web / app servers to memcached servers  $\Theta(\text{sub millisecond})$  can easily improve the performance of web apps. So, without much work, we can easily improve the performance and meet our first goal. These nodes make the Far Cache.

Will far cache reliably protect data tier? Perhaps not. A digged, slashdotted, techcrunched blog post may create hot-spots in certain memcached nodes. Memcached response slowdown may cause requests to backlog and some memcached nodes are likely to fall and network switches may saturate. Hot memcached node failures may expose data tier and there is never one cockroach.

To solve hot-spot problem, we introduce near caches. A near cache is a memcached node running on the web server or app server itself. We also cache a key in near cache and if we get a near cache miss, we fetch it from far cache and put into near cache. It prevents hot-spots from forming in the far cache. But wait, we just introduced another problem. The moment we start putting data into near cache, we run into cache consistency issues and user may observe the inconsistency. With near cache, inconsistency can't be eliminated without sacrificing scale out, but we can provide user observed consistency. This can be done by near caching only for few seconds — it works just the way alternating current works. If near cache timeout is smaller than user page round trip time plus user thinking time, then a user will not observe inconsistency introduced by near cache. 3 seconds near cache timeout is a sweet spot for us and guarantees that far cache will get 20 hits per minutes per cache key per web or app server and only user visiting next link within 3 seconds may see inconsistent data in some cases when far cache is being refreshed. We may not have impressive cache hit ratio for near cache but it protects far cache and network and provides soft user observed consistency. With some domain specific tweaking of cache keys and dynamic near cache timeout, we get 85% near cache hits.

Host Status Diagrams	
Cache Usage	Hits & Misses
 Free: 99.7 MB (9.74%)	 Hits: 2.5 B (85.69%)
 Used: 924.3 MB (90.26%)	 Misses: 415.9 M (14.31%)

Cache Information	
Current Items(total)	477.1 K (972.9 M)
Hits	2.5 B
Misses	415.9 M
Request Rate (hits, misses)	164.24 cache requests/second
Hit Rate	140.74 cache requests/second
Miss Rate	23.50 cache requests/second
Set Rate	54.97 cache requests/second

What is the timeout for far cache? Obviously, it should be grater than near cache timeout and if the data tier can’t handle troff traffic, then it must be greater than wavelength of the traffic cycle which is typically 24 hours. If you don’t want to charge users for far cache miss as much as possible, then far cache data for a long time (a week or so) and have an application level timeout by wrapping cached values.

```
class CacheValue {
Date last_updated;
Object value;
}
```

If the far cache value expires according to application level timeout, then we still serve the data from far cache but issue a background update. Keep application level timeout  $\Theta(\text{minutes})$  and Far cache timeout  $\Theta(\text{weeks})$ .

12 YEARS AGO      SHORT URL      COMMENTS

uncategorized

architecture / memcached

[PREVIOUS POST](#)   [NEXT POST](#)