1. Application server cache

Placing a cache directly on a request layer node enables the local storage of response data. Each time a request is made to the service, the node will quickly return local, cached data if it exists. If it is not in the cache, the requesting node will query the data from disk. The cache on one request layer node could also be located both in memory (which is very fast) and on the node’s local disk (faster than going to network storage).

What happens when you expand this to many nodes? If the request layer is expanded to multiple nodes, it’s still quite possible to have each node host its own cache. However, if your load balancer randomly distributes requests across the nodes, the same request will go to different nodes, thus increasing cache misses. Two choices for overcoming this hurdle are global caches and distributed caches.

1. Distributed cache

In a distributed cache, each of its nodes own part of the cached data. Typically, the cache is divided up using a consistent hashing function, such that if a request node is looking for a certain piece of data, it can quickly know where to look within the distributed cache to determine if that data is available. In this case, each node has a small piece of the cache, and will then send a request to another node for the data before going to the origin. Therefore, one of the advantages of a distributed cache is the ease by which we can increase the cache space, which can be achieved just by adding nodes to the request pool.

A disadvantage of distributed caching is resolving a missing node. Some distributed caches get around this by storing multiple copies of the data on different nodes; however, you can imagine how this logic can get complicated quickly, especially when you add or remove nodes from the request layer. Although even if a node disappears and part of the cache is lost, the requests will just pull from the origin—so it isn’t necessarily catastrophic!

1. Global Cache

A global cache is just as it sounds: all the nodes use the same single cache space. This involves adding a server, or file store of some sort, faster than your original store and accessible by all the request layer nodes. Each of the request nodes queries the cache in the same way it would a local one. This kind of caching scheme can get a bit complicated because it is very easy to overwhelm a single cache as the number of clients and requests increase, but is very effective in some architectures (particularly ones with specialized hardware that make this global cache very fast, or that have a fixed dataset that needs to be cached).