Caching

Overview

Caching involves copying or storing data in fast storage close to the application that will consume it (or at least closer than the original data store). Caches and load balancers facilitate horizontal scaling. Caching can provide the following benefits. Web browsers provide one simple means of caching. A web client caches data using the URI of resources as a key. A web application can then force the client (browser/proxy) to fetch the latest version.

Benefits of Caching

- Improved performance and scalability
- Improved availability
- Avoidance of repeated calculations

The following factors impact the effectiveness of caching strategies.

Drivers that impact decision to use Caching.

- Data is read more than written.
- Data is relatively static.
- DataSource is far from the consumer such that latency is a factor.
- DataSource is slower than the cache.
- DataSource is subject to a lot of contention.

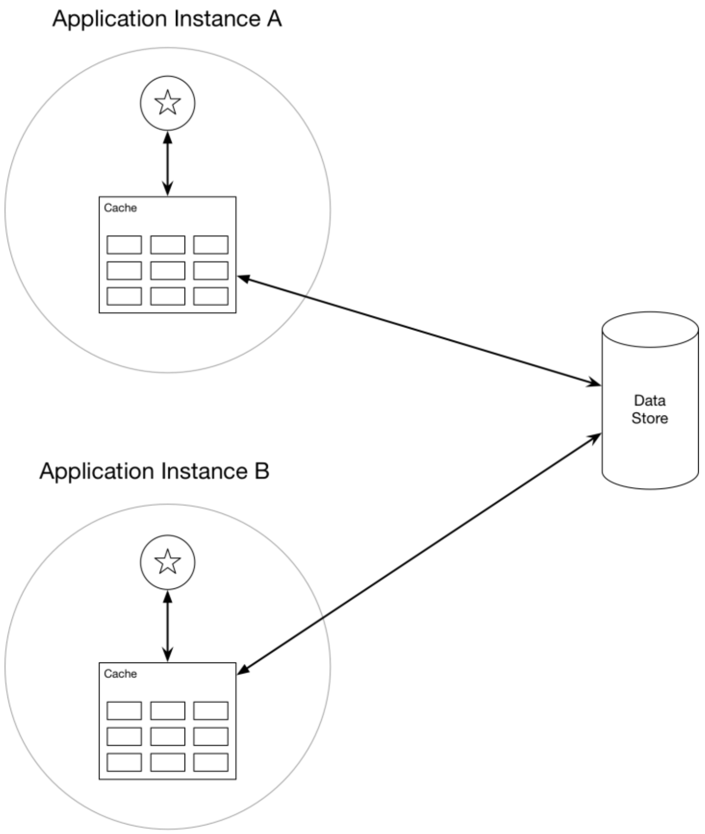
Private and Public Caches

In a distributed application caching can be either

1. Private cache – data held on same machine as application accessing it.
2. Shared cache – common resource accessed by multiple processes and machines.

Private Cache

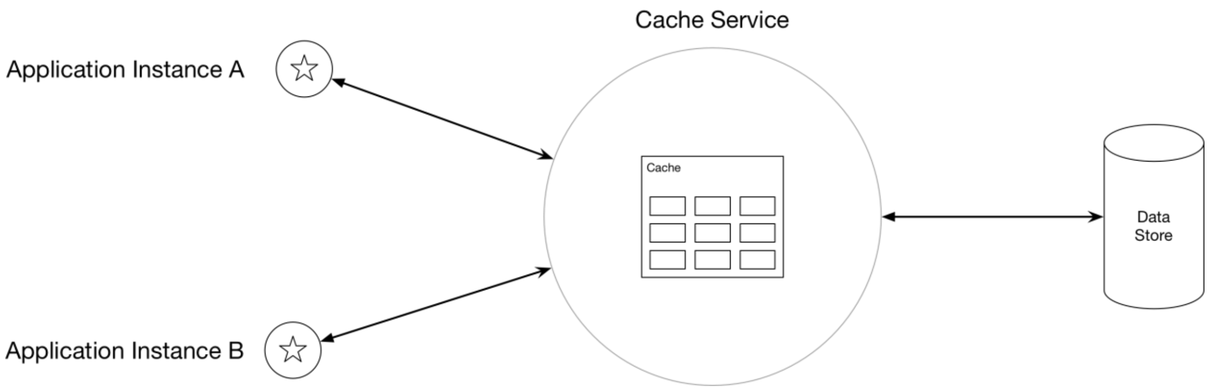
The simplest type of private cache is an in-memory store. It is very fast, however the amount of data that can be cached is constrained by the amount of memory on the machine. If we need to store more than can be accommodated in-memory we can use the local disk. Retrieving data from such a cache will be slower than from an in-memory cache but hopefully faster than accessing a remote data store across a network.



If multiple machines have their own private cache it is likely at some point different app instances will have different versions of a given piece of data in their caches. Shared caching can help with this issue.

Shared Cache

A shared cache as a service can help prevent the problem of multiple private caches having different versions of the data. Such a service typically uses a cluster of servers onto which it distributes the data. This enables the cache to be scaled by adding more servers.



Stale Data

When the original store changes after the cache has read from it the cache data becomes stale. A caching system supports expire dates such that the period during which data is out of date is minimised. As cached data expired it is removed from the cache and the application must retrieve the data from the original data store. The expiration period can be specified as an absolute time or as a sliding window. Deciding on the best expiration period is a trade-off. If one sets it too short, then the benefit of the cache will be reduced. Too long and the data will be stale more often.

If a cache becomes full the cache must evict data. Typically, it will follow a strategy to decide which data to evict. The default strategy is often LRU, however there exist alternatives.

Eviction Strategies

- MRU – assuming it won’t be needed again soon
- FIFO
- Explicit removal

Concurrency

Caches that are shared and updated by multiple application instances have the same concurrency issues as any data store. There are two approaches we can take to concurrency.

- Optimistic concurrency
- Pessimistic concurrency

With optimistic locking the updating application checks to see if the data has changed since it was read. If it has not been changed then it is updated. If the data has been updated since it was read it is up to the application to decide what to do. The optimistic strategy works well when updates are infrequent and there is low risk of collisions.

In pessimistic locking the cache is locked on retrieval and the lock is held until the update is down. Typically, it is useful for short lived operations where collisions are likely. The major downside is that it can limit concurrency.

Miscellaneous

Populating the cache

We can either cache at application start up (seeding) or on demand the first-time data is retrieved. Seeding can impose a large load on servers when the application starts up. In order to determine the best time to cache it is necessary to carry out performance testing and usage analysis.

Cache Down

If the cache goes down there are several strategies, we can use. We can fall back to the original store. This can generate a huge amount of traffic on the original store. One alternative is to use a combination of local private cache and remote public cache. If we are using a cache as a service, then the service can carry out automatic failover.

Security

We can protect our data by using SSL to protect data over the wire and use authentication and authorisation services on a cache as service.

Immutable Data

Immutable data or data that changes infrequently lends itself well to caching. Examples would be reference data or product data.

Questions – Caching

Introduction

What is caching?

Copying or storing data in fast storage close to application

What is the intention of caching?

Improve performance and scalability and availability

Reduce latency and contention associated with large volumes of concurrent reads

Avoid repeated calculations

For what kind of data is caching most effective?

Client repeatedly reads the same data

Data is read more than written

Data is relatively static.

What properties of a data store make caching a good option?

Far from the application such that latency is an issue

Slower than the cache

Subject to a lot of contention

Give example of where cache is particularly useful at improving performance?

DB supports limited number of concurrent connections

Even if this number is exhausted clients can still access data via the cache

Give example of where cache is particularly useful at improving availability?

DB goes down and clients can access data via a cache

Caching Strategies

What are the two caching strategies?

Private – cache held on machine that is running an application

Private - cache shared by multiple machines and processes.

What is the simplest form of private cache?

In memory store

What is the disadvantage of in memory cache?

The amount of data that can be cached is constrained by the machine's memory

How can a private cache overcome this limitation?

By writing to disk

What are the benefits of private cache?

Fast

What are the disadvantages of private caches?

Multiple applications have their own private cache and different versions of the same data

What can be used to overcome this problem?

Use a shared/public cache

Describe the features of cache as a service

Utilises a cluster of servers onto which software distributes the data.

What does this enable?

The cache can be scaled by adding more servers.

What are the disadvantages of shared caches?

Slower than private caches

Increased complexity

What is key to good caching?

Determining what and when to cache.

Stale Data

Describe a general problem with caching?

Data can become stale if the original store changes after cache is updated.

How do caching systems deal with this?

By supporting expire dates

As cached data expires it is removed from the cache.

How can the expiration period be specified?

As an absolute time or as a sliding window.

What are the trade-offs when setting cache expiration period?

To short and objects expire to quick reducing benefit of cache

Too long and data becomes stale more often

What happens if a cache becomes full?

Caches evict data on a LRU basis by default

What are the alternatives to LRU?

MRU – assuming wont be needed again soon

FIFO

Explicit removal such as data being modified

Web Clients

How is client side caching done on a web client?

The URI of a resource is used as a key

The web app can force the client (browser/proxy) to fetch the latest

Concurrency

What are the two forms of concurrency strategy?

Optimistic

Pessimistic

What is optimistic?

Before updating the data app checks if cache has changes since it was read. If not update is made, otherwise app need to decide what to do

When is optimistic useful

Infrequent updates

Low change of collisions

What is pessimistic?

Data is locked in cache on retrieval and cannot be accessed until the update is done.

When should pessimistic be used?

Short lived operations

Collisions are likely

Need to consistently apply multiple updates

What is the downside of pessimistic?

Limit concurrency.

Cache Down

What are options when cache goes down?

Fall back to original data store

Why is this bad?

Huge load on data store

Give alternative

Use combination of local private cache and remote public cache.

Give another alternative

Cache service provides automatic failover

Shared caches distribute data over nodes and rebalance the data.

Security

How can one protect cached data?

Use an SSL connection to protected data over TCP/HTTP

Use an authentication and authorisation model on cache service

When measuring the performance on an application that communicates with other end points over a network, we will consider three important terms.

	Description
Bandwidth	Maximum amount of data per unit of time e.g. gigabytes per second
Throughput	Actual amount of data per unit of time
Latency	The delay between the sender sending and the receiver reiving information.

We can think of bandwidth as the width of the pipe and latency as the delay incurred by packets traveling from one end of the pipe to the other. If I use a high bandwidth, high latency system I would expect to wait a while for my file to start downloading but from the point I see the download start it would be fast. If I were using a low latency, low bandwidth channel the file would start downloading immediately but the download would take a while.

> ### Acceptable Delay
>
> Most people notice a delay of 100-200ms and any delay greater than 300ms is considered sluggish. 1second starts to be a bad delay. To provide a great use experience we need to aim for response times of 100-200ms. This is one of the driving factors behind the adoption of content delivery networks to locate servers close to users. See the following link

We will now look at the three elements of networking performance.

Latency

Latency describes the time taken by a packet to travel from sender to receiver. Latency is impacted by the following factors.

	Description
Propagation Delay	Length of channel dividend by the speed of the signal travelling down it. Typically, the speed of the signal is close to the speed of light.
Transmission Delay	Time required to push a packet's bits onto the wire. Simply a function of the number of bits in the packet and the rate of transmission in bits per second. If we have a 100Mb file then it will take 100 seconds to push it onto the wire using a 1Mb/s link and 1 second to push it onto the wire using a 100Mb/s link
Processing Delay	The delay incurred by a router processing the packet header, handing errors and deciding on the next destination
Queuing Delay	If incoming packets arrive faster than a router can deal with them, they get queued inside a buffer

Typically, a packet traveling across a network will encounter many intermediate routers and will hence be impacted by multiple processing and transmission delays. Where load is heavy, we have increased risk of incurring queueing delays.

Very roughly we can assume that a signal in fibre travels at about 200,000,000 metres per second. The distance between London and New York is 5500km so the round trip in seconds is 0.055 seconds or 55ms. The world circumference is 40075 km, so the time taken to circumnavigate the globe is then 0.2 seconds or 200ms. In practice the route is not the minimum distance and there will be multiple transmission, processing and queueing delays.

Another problem is that there is a delay between the home and the ISP of between 10ms and 70ms. This is known as the last mile latency.

These might add 100% giving us 400ms to circumvent the globe and 110ms for round trip between New York and London. This latency is affected by the ISP, technology used and time of day.

Bandwidth

A single fibre optic link support multiple channels via wavelength multiplexing. The capacity of a fibre is 171Gb/s so across all channels we get 70Tb/s. A single cable often has four strands of fibre giving a bandwidth of 280Tb/s. The ends of these fibres, however, are connected to much lower bandwidth technologies such as DSL, cable and wireless technologies. The available bandwidth is driven by the weakest link which is typically these low-capacity technologies. In 2018 the worlds average download speed was 46Mbs and upload speed was 22Mbs. My own sky gave figures of 16Mbs and upload 0.9Mbs.

The following online resource provides details on networking.

https://hpbn.co