Cache Problems and Solution

1. Cache Penetration
Refers to a situation when a potential hacker flood server with numerous request containing random keys and subsequently absent from the primary database, causing primary DB become overwhelmed, causing failure.
Techniques to Mitigate
1. Caching Empty Result
1. Technique to address cache penetration is to cache empty results
2. Problem with this technique: wasteful utilisation of valuable in-memory storage in case of heavy request going miss from primary DB.
1. Resolution : Evict empty memory space after specific time period.
2. Resolution : Strong eviction policy, making eviction decisions based upon various factors like cache usage pattern, access frequency.
2. Bloom Filter
1. Serves as a protective mechanism, implementing either an allow-list or block-list to filter request before they reach the primary database.

2. Cache BreakDown
A situation where hot key got expired or gets evicted from cache unexpectedly.
When this happens multiple requests simultaneously query the same data, overwhelming the database due to high concurrency.
Techniques to Mitigate.
1. Refresh Ahead Caching Strategy.
proactive approach by refreshing the cache before the cached key expires, keeping data hot until it's no longer in high demand.
Start by choosing refresh ahead factor say 50% for 100sec
So for first 50th second requests will be server normally and after 50th second the cached data will still be returned but background worker will trigger data refresh.

3. Cache Avalanche
Situation where significant number of cached keys cannot be located in cache layer simultaneously, leading to sudden surge of queries overwhelming the database.
Two primary reasons behind cache avalanches
Many cached keys expires simultaneously.
cache layer experience a crash or become inaccessible for a period of time.

Technique to Mitigate
1. Expiration Time Jitter
introduce random time jitter to each cached key, ensuring cached keys don't all expire simultaneously.
2. Refresh-Ahead Caching Strategy
Same as above.
3. High-Availability
Make cache layer highly available.

****
https://medium.com/bumble-tech/working-with-a-cache-problems-and-solutions-php-36ed76451ac

If multiple thread tries to access data from cache and data is not available, each try to get the data from primary source. This situation is referred to as "hit-miss storm", "dog-pile effect" or a "cache stampede"

Their are several ways to resolve this:
1. Blocking the re-computation/load data operation before it starts
1. The idea is to block the process that want to load it and effectively also prevent other processes running in parallel from doing the same thing.
2. In this case data get updated in one process only, but you do need to decide what to do with the processes that encounter a missing cache but which couldn't be locked to prevernt data loading. These might return an error or default value or perhaps wait for a certain time, before attempting to obtain data again.
3. Blocking time need to be decided carefully and long enough to load data from source and save to cache.
2. Moving updates to background
1. Updates will be done by the background process/script whose sole responsibility is to update the cache and setting up the time of next launch.
3. Probabilistic updating methods
1. Main idea is that data in the cache get updated not only when data is absent from cache, but also when there is certain level of probability that data is present in the cache. This allow us to update the data before the cache data expires. And get requested simultaneously by all process.
4. Cold Start and warming up the cache.
1. Request for particular key or large number of simultaneous updates of different keys.
2. Techniques to resolve:
1. Switch on new functionality graduallyXopening feature to small number of user and gradually expand to big set of audience) this allow cache to warm up this reduce surge of request.
2. Setting different lifetimes for different elements in the data
5. Hot Keys: Writing data not on single server but to several server at the same time. We reduce sever fold the number of time key is read. Other approach is to write data to all the available server to avoid load on single server.
6. Downtime handling.