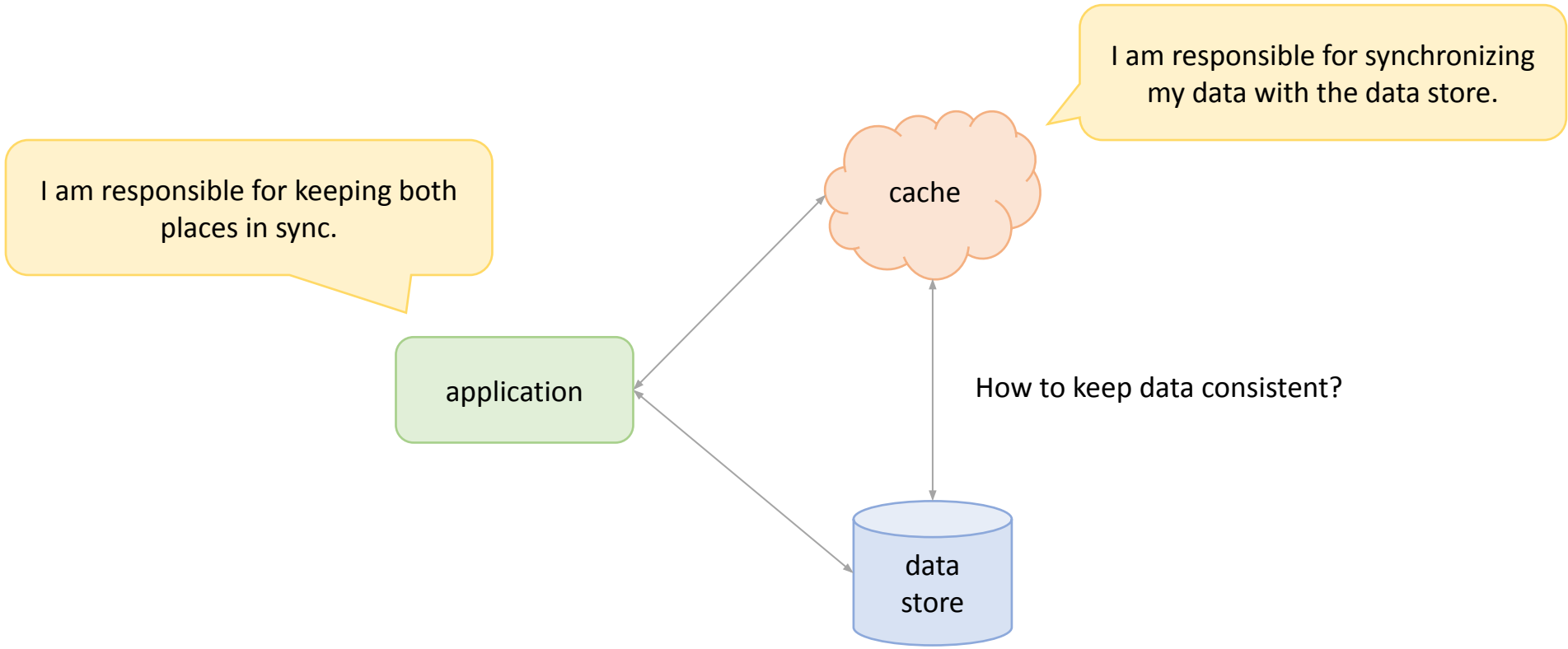
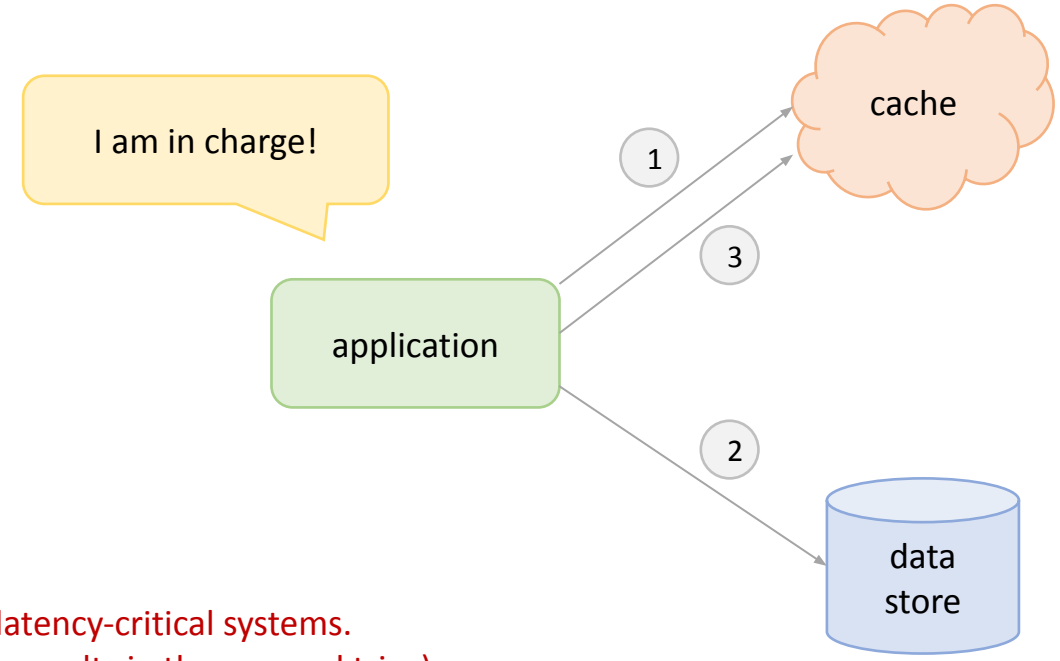
Metadata cache



Metadata cache

cache-aside pattern



1. application looks for an entry in the cache
2. application loads data from the data store
3. application adds a new entry to the cache

cons

- Less suitable for latency-critical systems.
  (every cache miss results in three round trips)
- Stale data when data changes frequently in the data store.
  (mitigated by setting expiration time on cache entries)
- Prone to cache stampede behavior.
  (multiple threads simultaneously query the same entry from the data store)

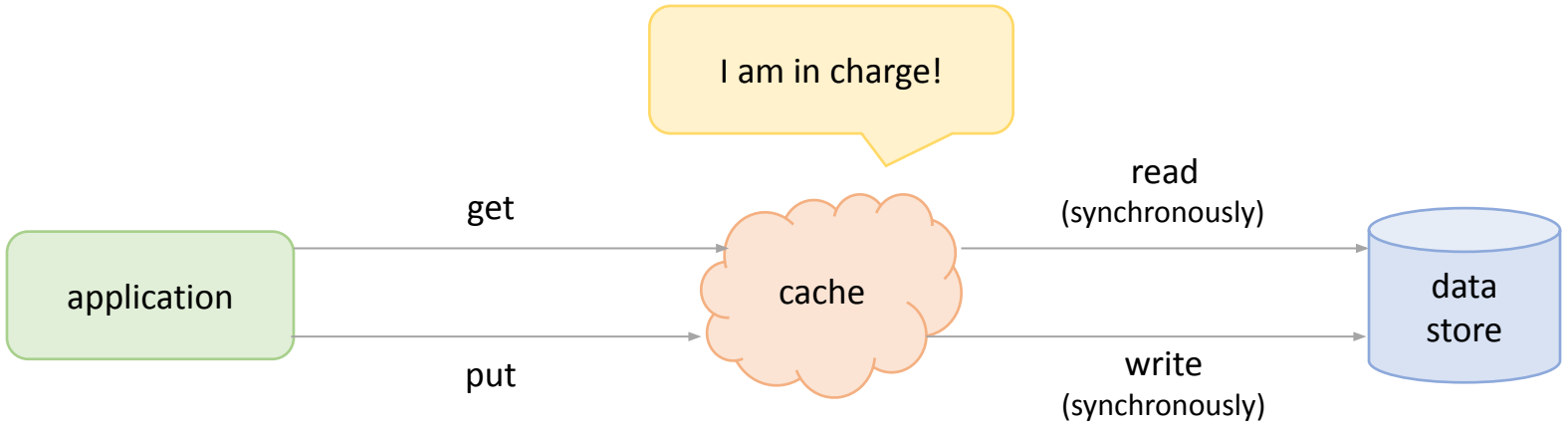
pros

- Widely used pattern.

Metadata cache

read-through pattern

write-through pattern

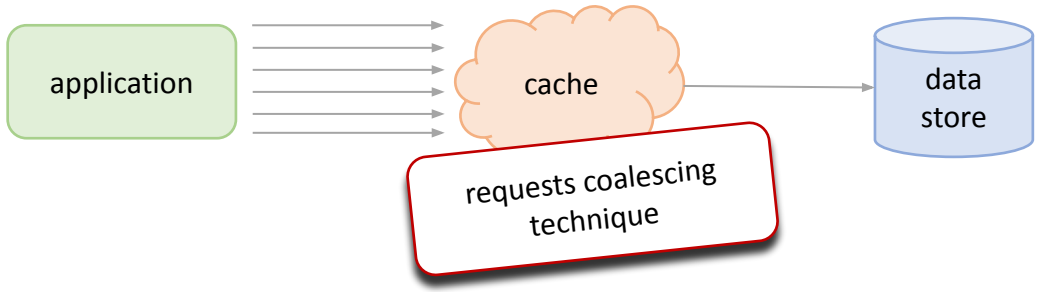


cons

- Cache may contain a lot of rarely used data.
  (a problem for small caches)
- Cache becomes a critical component for the system.

pros

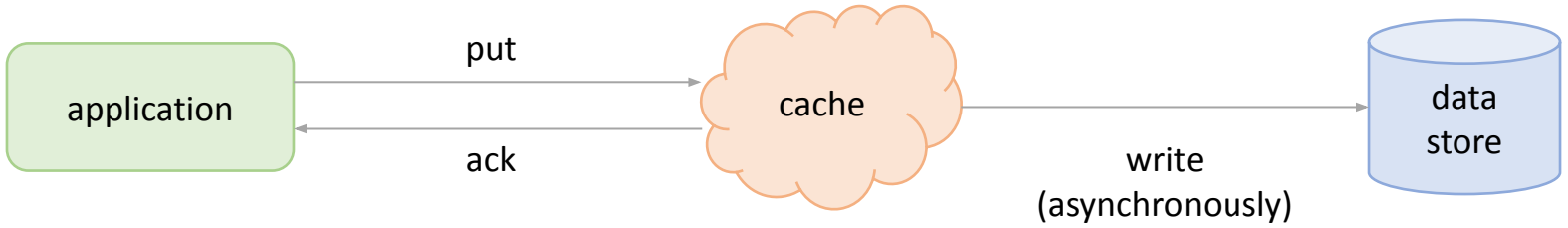
- (both patterns) Simplify data access code on the application side.
- (read-through) Helps to mitigate the cache stampede problem.



Metadata cache

write-behind pattern

(aka write-back)



cons

- Data can be lost.
  (mitigated by data replication in cache)

pros

- Better write performance (higher throughput, lower latency).

Metadata cache