The assumptions

For any cache implementation you need some statistics for your service to have an idea of what to cache, here I will write my assumptions, based on which I cache one thing or another.

1. I assume the system is very read heavy with a low number of updates. (By updates I consider not only insertion, update or deletion of products but also buying)
2. I assume a cloud based environment where reducing the size and number of requests to the database is a good idea not only for latency but also for cost considerations.
3. I assume a requirement for getting up-to-date information in GETrequests.

Implementation

I will use Flask-caching library that provides an easy to use decorators for flask view function to store the responses in redis, you can see example redis state below:

```
1) "flask_cache_/productsc49e76f84f1209dace582b7b97b9470f"
2) "flask_cache_/products/2bcd8b0c2eb1fce714eab6cef0d771acc"
3) "flask_cache_/products48b2ecdbf6e54fb65061b3af1a32d10f"
4) "SERVER_ONE"
5) "flask_cache_/products9c2a94cf9ec58f545b606e59f4041485"
6) "4"
7) "flask_cache_/products/4bcd8b0c2eb1fce714eab6cef0d771acc"
```

I will cache the products list endpoint and separate product endpoint, both of which are gets and I assume are accessed with magnitudes of order more often than any update endpoints.
I will also clear the cache when any update happens to assure up-to-date information. While the clearing of the cache is somewhat of a time consuming operation, with assumptions above it should be worth the tradeoff.
On most of the requests it will eliminate the need for sending requests to the database, which will improve the availability and performance of our database system, and reduce latency. As the cache is not bound to one server it will also allow different instances to share their results that are already “computed”.