Number of users - 10 million
Each user buys an item daily, assuming those action are distributed evenly during the day we need to handle 10000000/(60*60*24) ~ 115 buy requests per second on average.
As reads traditionally occur orders of magnitude more often, we will assume 10000 reads per second. As the number of products is not mentioned and in my initial design the store offers very few products (each of which can be sold thousands of times), I will keep that assumption to avoid making changes in cache implementation.

We will do our planning in four steps: Nginx, Flask servers and Redis cache and PostgreSQL database.

Nginx
According the the Nginx plus size guide (https://www.nginx.com/resources/datasheets/nginx-plus-sizing-guide/) we would have more than enough throughput deploying nginx on a 4-core 8GB RAM instance with very limited storage, as we will use nginx caching only for serving the front end artifacts. Note that the size of our requests is relatively small: 292 bytes for update (buy) requests with response of just 4 bytes . The largest response size being the response of product list endpoint: 3480 bytes assuming 20 products. So assuming 10K reads per second, and all of them on the heaviest endpoint with no pagination limits and 200 update requests of various types we get approximately 40 MG of network traffic per second, well within the abilities of Nginx to handle given the instance. The same amount of traffic is not an issue with the other parts of the system.

Flask servers and Redis cache
After switching to async Gunicorn workers (which is the go-to choice for IO bound tasks), each of our backend instances will be able to handle 5000 requests concurrently, when deployed on an instance with at least 2 CPU cores and 4 GB of RAM. We will scale the Flask backend horizontally to reach even more capacity, in our case we are good to go with 2 instances, assuming that the cache and database networking is done in the same cloud VPC.
As for Redis, we will be storing a relatively small amount of data, at best, 20 separate product data and product list data, the network limitations mentioned above are not relevant to Redis, as a single instance with 2 CPU cores and 4GB of RAM will be able to handle millions of requests per second.

The database
For data of 10M users, and average estimated size of user data being 5000 bytes, which is a rather large number to assume, we would need 50 GB of persistent storage + a negligible amount of storage for products. It should be no issue for a single instance of PostreSQL to handle more than 10K transactions per second, while we only need around 200. Assuming that we would want to have a replica of our main instance. We need two 4 CPU, 16GB RAM, 55 GB storage instances. (The size is a bit larger than what we need at minimum, but PostgreSQL makes a good use of parallelism with 4 CPU cores and will use 16GM of ram for internal cache.)

You can check the code updates at https://github.com/SayatP/CS-392-SD.