**Question**

Stock Data:

Imagine you are building some sort of service that will be called by up to 1000 client applications to get simple end-of-day stock price information (Open, Close, High and Low). You may assume that you already have the data, and you can store it in any format you wish. How would you design the client-facing service that provides the information to client applications? You are responsible for the Development, Rollout and ongoing Monitoring and Maintenance of the feed. Describe the different methods you considered and why you would recommend you approach. Your service can use any technologies you wish and can distribute the information to the client applications in any mechanism you choose.

**Initial Queries**

1. Are we dealing with Stock price of a single company? If that is the case it will be strait forward. i.e., Storing the data and service to as many client as needed. But assuming that it is not the case we proceed. **Assumption made here**: We are going to have many companies
2. Now since we are going to have a lot of companies, again we make an assumption that the list is very high running in millions. So we can use a hashing technique to split the companies based on company name and storing it in different servers.
   1. Here there can be two assumptions, First is that new companies are going to be added and they can be added after a certain period of time (let’s say each quarter). Now as we are using hashing technique it will be expensive to rehash each and every time when additions occur and their frequency is very high. In our assumption we made that it happens each quarter then it won’t be that much considering the easy access of each company and the no. of request which can be processed per time interval.
   2. If no new companies are going to be added. Then also we can use the hashing technique and it is better since we need not rehash after a certain time.
   3. If new companies needs to be added frequently, then is there a possibility that clients are searching based on the Company ID, so that we can add the company in order of insertion and use the same hashing to find the DB where is will reside. If new DB is required we can add (Horizontal Scaling). **Assumption made here**: is the first one (assumption a)
3. It has been mentioned as End of Day stock and so there is no frequent updating of the request. This paves way for utilizing the **Cache** concept to reduce the frequency of request to server each time the same details are required. We need to make the cache to be alive for a day. **Note: This concept will explained further below.**
4. We can use the NoSQL for storing the data of the company, since there is a potential for adding additional company details in the future and the traditional SQL might be problematic. **Note: Assumption is that NoSQL has features as expected, since I am not much into it have to dwell a little deeper.**
5. How the data is feed into the DB before it is being served each day?
   1. We can create a separate portal where users with access can be feeding the data (or) it can be automated to gather the details from some stock exchange portal, since there is possibility of wrong info being entered if fed manually.

**Technologies**

1. **Multiple servers** to hold the data. i.e., the company names will be used as the key for **hashing** and the specific range of companies will be present in a particular server. So when requested we can look up and find the Server where the details will be present and access particularly.
2. **NoSQL** – In future if we want to add more information to the DB then it will require the DB Schema and other Query design changes. NoSQL might overcome these issues. If not, then use either one.
3. Web Service – Any of the below can be used. The Request should be in the format [User Details], [List of Companies for which the stock price is needed]. It could be JSON, XML, etc.
   1. SOAP WSDL
   2. RESTful
   3. Microservices
4. Cache

**Cache**

**Client**

**Web Service**

**Hashed DB**

1

2

1

2

When the initial request was not present in the browser cache it requested to the web service to provide the info. It stored upon receiving it.

When the same request is being made for the second time then the cache will directly provide back the info

**Flow**

The DB will be hashed based on the Company names, and the details will be saved. **Web service** will expect a request in a certain format as discussed earlier and then perform the fetching operation from the corresponding server and build the response and send it back. Since the hashing has been made it will be easier to find the location of a specific company.

In the client application we can implement the **EhCache** mechanism to store a data for a day and expire EOD