**Question**

Imagine a web server for a simplified search engine. This system has 100 machines to respond to search queries, which may then call out using *processSearch (String query)* to another cluster of machines to actually get the result. The machine which responds to a given query is chosen at random, so you cannot guarantee that the same machine will always respond to the same request. The method processSearch is very expensive. Design a caching mechanism for the most recent queries. Be sure to explain how you would update the cache when data changes.

**Solution**

**Web Server**

**Helper Cluster**

1

2

100

**Search Cluster**

1

2

100

**Cache**

**Users**

1

2

100000000

**P**

**P**

ProcessSearch(String query) method

**Request**

**Response**

**Cache Map**

**3**

**2**

**1**

**1000**

**Queue <Request>**

1. When a request is made first we need to check the cache for the response using the request as the key. If it is not available then the *ProcessSearch (String query)* method is invoked and the response is saved in the Cache Map <Request, Response> and the queue is added with the Request.
2. Now before saving in the Cache we have to check if the queue is full. If full then we take out the 1st element in the queue and remove it from the Cache Map and then add the new entry and add to the queue as well.
3. Queue should be implemented with the LinkedList since the removal of the first element and the addition of the last element can be made in big (1) constant time.
4. We are assuming here that 1000 entries will suffice. If there is a need to increase the cache size it can be made by increasing the Map and Queue.

**Modification (Optimal Solution)**

It was assumed that the cache will be common to all the machine, which is actually not possible. So we can use 3 scenario

1. Each machine has its own cache – What if the particular query is new to the machine, it will not be able to identify the machine and so it will start calling the ProcessSearch which is expensive
2. Each machine shares a copy of cache – Duplicating the cache will render it difficult to update the cache when there is a change
3. Each machine has a part of the cache – Hashing the query and saving it across the different machines will ensure that the data is split across the hash table.

As for the most recent queries alone should stay in the cache, we can a set a time-limit or the priority-limit and update the priority again when the query is made again.