**Cache:** 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:**

1

2

3

4

N

**Clients**

1

2

3

100

**100 Machines**

**Load Balancer 1**

**Load Balancer 2**

**Active – Passive Load Balancer**

**1**

**2**

**Load Balancer 1**

**Load Balancer 2**

**In-Memory Cache**

1

2

3

N

**Cluster of Machines**

**processSearch**

**3**

**5**

**Read Replicas**

**Master – Master**

**4**

**6**

**Cache Invalidation when data changes**

1 – Fetch the query data from the cache

2 and 3 – If not present then invoke processSearch from the cluster of machines

4 and 5 – Cluster will compute the result and return it to the load balancer

6 – Cache the query and then return to the client

**Cache Invalidation**

The Cache has to be invalidated when the data change occurs in the master-master

The read replica will handle the read functionality and the master will concentrate on saving the crawled websites to index the search for faster retrieval

**Cache Expiry**

Since the large query will be cached we need to remove the less popular query based on the number of hits. Linked list can be used to achieve easy removal of the less used cache query.

The cache has to be efficiently retrieved by using a HashMap<Key, Value>

Combine the both qualities LinkedHashMap<Key, Value>. When a hit occurs move the Key up the rank, and expire the cache after a certain interval of time even though they get huge hit since this will make up space for other complex queries as well, so that they are not starved of a cache space.