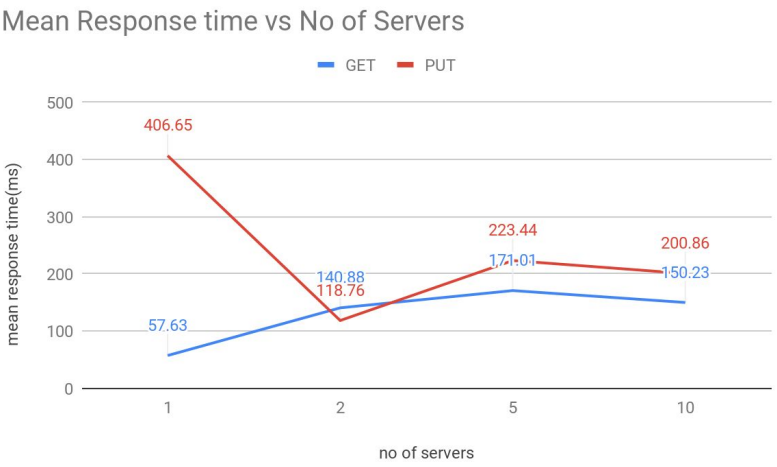
Performance Test Report

For the different test scenarios we used the below configurations. We used 3 Raspberry Pis to run our KVServers which have below configuration. All Raspberry Pis were connected to the same local network over a LAN 100MB switch. They were used to run clients and servers.

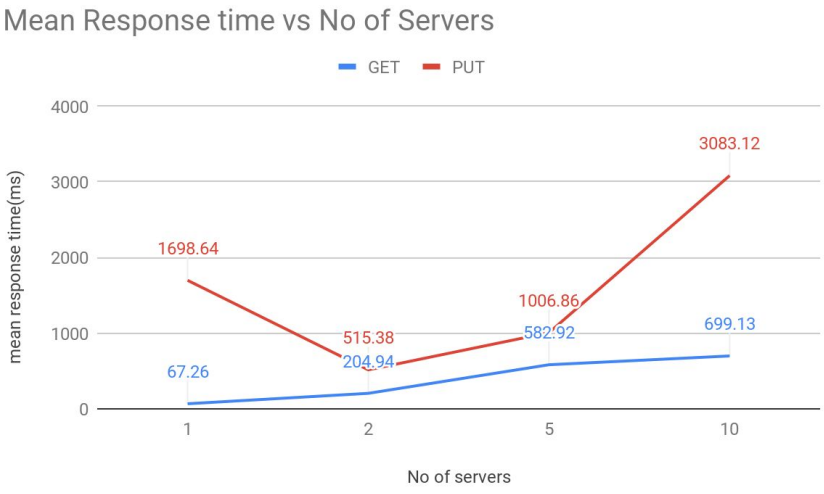
- **Pi 1** - 700 MHz single core, 512 MB RAM
- **Pi 3** - 1.2 GHz quad core, 1 GB RAM
- **Pi 3B** - 1.2 GHz quad core, 1 GB RAM

The data set used for the evaluation was enron data set available with the link. During all test runs the respective number of clients was run evenly distributed on all servers and in parallel. They all issued 80% Get and 20% Put requests.

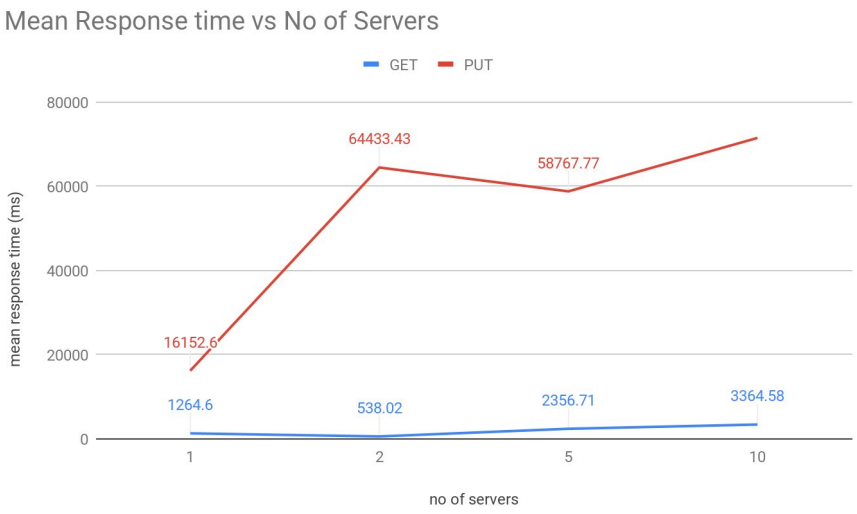
***Scenario 1** : Running 1 KVClient with on 1, 2, 5, 10 KVServers with cache LFU and cache size limited to 100.*



***Scenario 2** : Running 10 KVClients with on 1, 2, 5, 10 KVServers with cache LFU and cache size limited to 100.*



***Scenario 3** : Running 100 KVClient with on 1, 2, 5, 10 KVServers with cache LFU and cache size limited to 100.*



Explanation for above scenarios

Expected behaviour

With increased no of servers we expected reduced response time for GET requests. GET requests should be faster because

1. Increasing servers add the ability to handle more concurrent GET requests
2. Each server has its own cache. So the cache hit rate should be increased with increasing no of servers running.
3. The size of the database gets smaller with increasing no of servers. So I/O operations should be faster

Expected behaviour

We expected a reduced response time for PUT while increasing servers running because

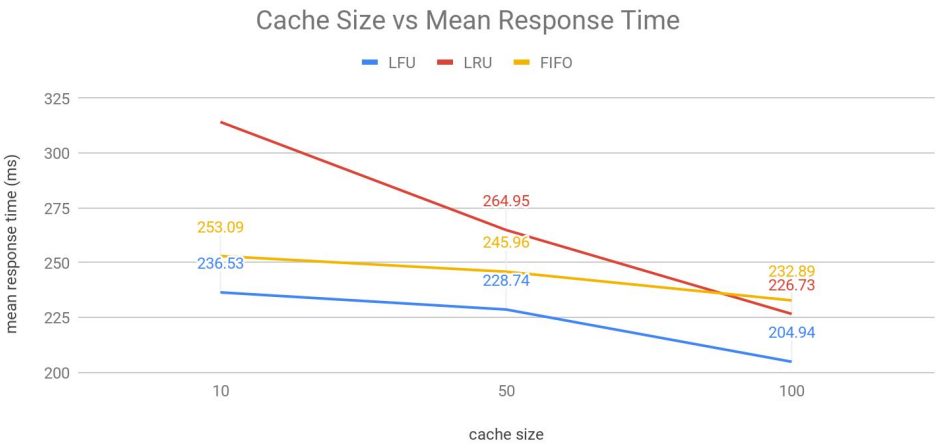
1. Increasing servers add the ability to handle more concurrent PUT requests
2. The size of the database gets smaller with increasing no of servers. So I/O operations should be faster

WIth increased no of clients should experience in slowness for GET and PUT because no of parallel requests to the servers are increasing.

Observed behaviour

The behaviour of performance values are irregular sometimes as per our expected behaviour as mentioned above. We observed that the distribution of keys over servers was not uniform. Also our servers were not all with equal specifications. Slower servers had more keys than faster servers.

Scenario 4 *: Varying caching policy (FIFO, LFU, LRU) for cache sizes 10, 50, 100 for 10 KVClients with 2 KVServers*



Explanation: LFU performs better since the probability is higher for a certain key to be in cache compared to LRU or FIFO. This is the expected behaviour also. SInce the no of servers didn't vary across test scenarios we hadn't any irregular results as in scenario 1, 2 and 3.