Evaluation of Storage Service implementation for Milestone 3

As part of Milestone 3 we evaluated our distributed key-value store implementation in various scenarios. We measured the latency between the client send request and the response received for the request. Besides, we measured the throughput on the server. In the lack of time, we just measured the values for the PUT operation, not for the GET.

The data set which we used was the Enron mail dataset. First, we transformed the dataset into several CSV files. The CSV files contained two columns. The first column was a key which was randomly generated and its length was 20 bytes in every case. The second column was a value, which was the content of the respective file from which it was created. Its size was at most 200 kilobytes. In case the content had been larger than that size, it would have been cut into pieces with size at most 200 kilobytes. However, in case of the dataset it never occurred.

Second, we developed a tool which uses the same KVStore interface as the KVClient does. In this way we could read the CSV files and PUT the read key-value pairs automatically to the respective KVServer.

Third, with the ECS we allocated several KVServers with the initialize server command, and after that we gave one of these server's IP address + port to the aforementioned tool as a bootstrap KVServer connection info. As soon as the tool started pushing key-value pairs to the storage service it received the metadata structure so that all key-value pairs were transferred to the responsible server.

Regarding the scenarios, we measured latency and throughput for one client, and two clients with either three servers or six servers. The number of servers were not changed after the ring was initialized (lack of time to do experiments for that). So in this way we had 4 different configurations: 1 client – 3 server, 2 clients – 3servers, 1 client – 6 servers, 2 clients – 6 servers.

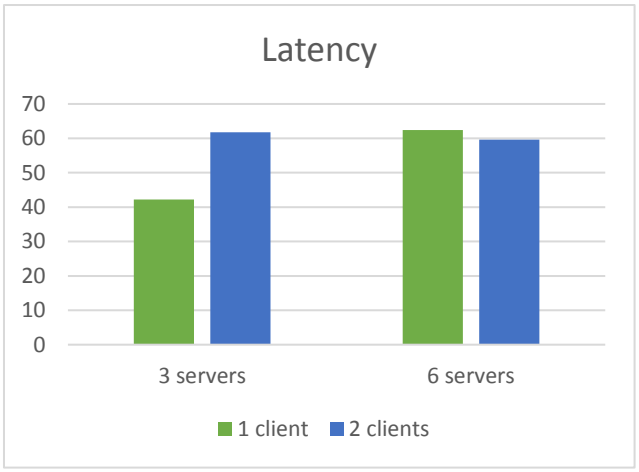
Latency was measured in milliseconds, while throughput was measured by number of PUT requests processed per seconds. Every experiment was run for two minutes. In the followings we show several figures about the experiments.

The servers were deployed on DigitalOcean a virtual machine provider where the VMs are SSD based. The datacentre was in Frankfurt.

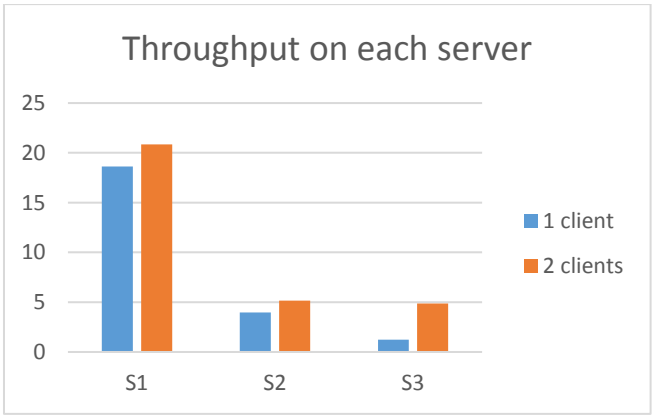
On Figure 1, the latency between the PUT request sent from the client and response received to that is shown in milliseconds. The average latency was 56 milliseconds.

On Figure 2, the throughput per server is shown in case of three servers. It can be seen that most of the data was handled by the first server (S1) in the ring.

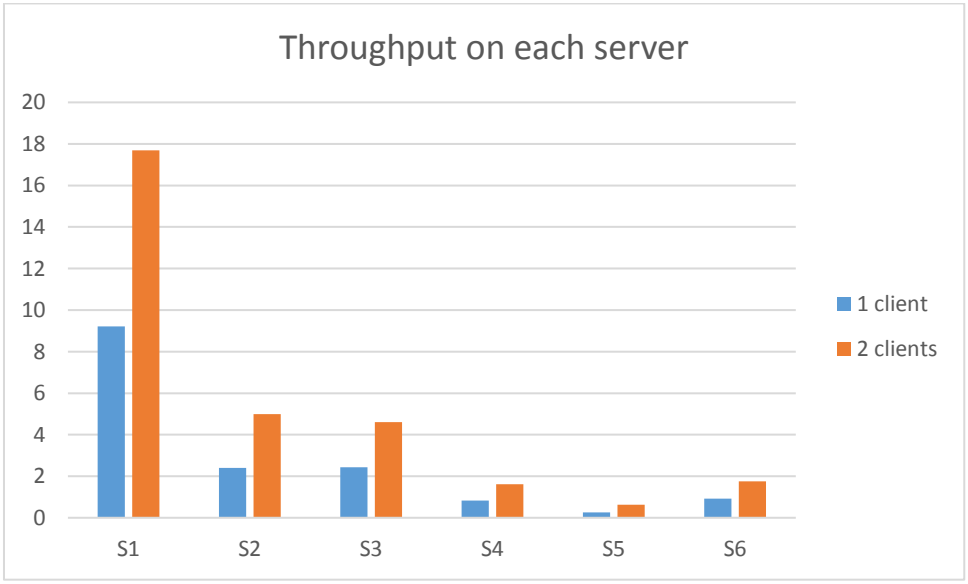
On Figure 3, the throughput per server is shown in case of six servers. Compared to the previous scenario, in this case the ring was more balanced among those five servers which were in latter positions. However, the first server in this case handled most of the hash values.



1. Figure Latency of PUT operation measured from the client side



2. Figure Througput per server (3 servers)



3. Figure Throughput per server (6 servers)