DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Milestone 3 Evaluation

Group 7

Tuan Anh Ma

Fedor Fedoseev

Florian Alkofer

Test Setup

The benchmark is conducted across three machines with the following specifications:

	Machine1	**Machine2**	**Machine3**
Processor	i5-3210M 2.50GHz	i5-2410M 2.30GHz	i5-2520M 2.50GHz
RAM	8GB DDR3 1333 MHz	8GB DDR3 1333 MHz	4GB DDR3 1333 MHz
Storage	Samsung SSD 850 256GB	Samsung SSD 850 PRO 128GB	Seagate 320 GB, 7200 RPM, 16 MB Cache

Server specifications

Machine1 runs all of the clients while Machine2 and Machine3 share an equal number of server threads. Each client will send a set of 1000 random emails read from the Enron dataset to the servers
We set a stopwatch in every client thread and measure how much time a client needs to send all 1000 operations. Having the run-time, the throughput is then calculated by $\frac{1000}{run\_time}$ (ops/s) and the latency by $\frac{1}{throughput}$ (s/ops). The overall throughput and latency of a particular setting is determined by averaging these quantifies over all clients. Every setting is tested three times to minimize the effects of randomness and the corresponding outcomes are averaged over the averages of each setting.
The benchmarks of different cache sizes and displacement strategies in Table 2 are performed with a single server (on Machine2) and 5 clients. On the other hand, we fix the cache size and displacement strategy for the tests of multiple servers and clients in Table 1 with a cache size of 1000 and FIFO strategy.

Multiple clients and servers

In general, the performance drops as the number of clients and servers increases. There are two reasons for this behavior. On one hand, the machines and server threads have to carry more load, on the other hand, the storage ring is divided into more and smaller ranges as the number of servers increases, therefore, the clients have to redirect its TCP connection to the responsible server too frequently. One of the suggested solutions is to implement client-server interaction over UDP in upcoming versions to reduce the cost of building up and tearing down the stateful connections.

Different cache sizes and displacement strategies

The test in Table 2 shows that a larger cache size generally increases performance. From our benchmark, there is no clearly discernible best displacement strategy for our service.

System scaling

With the available data of 10.000 key-value pairs in the system across 5 servers, the service requires 15.56 seconds to scale down and 17.78 seconds to scale up by one server on average.

		Servers		
		2	**8**	**16**
5 Clients	**PUT**	133.55	64.23	58.53
	GET	756.42	464.75	422.15
12 Clients	**PUT**	32.43	12.29	10.775
	GET	283.17	47.57	204.33

Table 1: Throughput (ops/second) with multiple clients and servers

		Cache Size		
		100	**1000**	**5000**
PUT	**FIFO**	139.87	152.18	149.94
	LRU	146.61	159.54	149.77
	LFU	167.16	145.87	149.71
GET	FIFO	755.22	809.51	890.33
	LRU	888.74	848.37	893.79
	LFU	797.31	894.59	904.04

Table 2: Throughput (ops/second) with different cache sizes and displacement strategies