CS739 p1 kvstore

First, you will describe/argue why your store is strongly durable (what is your write protocol and what you do upon recovery etc).

- sync.map
    - we update the log file per set
    - we build the in memory database by the log file
    - mutli-version (read map / write map) and usually only lock write, we lock read when read map has no data
- bptree
    - we have a log.txt and a db.txt
    - we build the in memory database on both log.txt and db.txt, meanwhile, we clear the log.txt and update the db.txt
    - we update the db.txt by both in memory database and log.txt per minute
    - atomic swap and delta chain

Second, you will show some graphs from your measurements and explain the trends. High performance is not a goal for this project; instead, you should be able to understand and explain why your system has a particular performance behavior.

Finally, you should say what's one thing that's similar and different about the RPC package you used compared to what's described in the Xerox RPC paper.

- difference
    - Xerox choses to have no time-out mechanism limiting the duration of a remote call (in the absence of machine or communication failures), whereas most communication packages consider this a worthwhile feature.Our argument is that local procedure calls have no time-out mechanism.
    - Xerox believes the request-response nature of communication with RPC is sufficiently unlike the large data transfers for which bytes streams are usually employed
    - grpc has load balance feature and supports stream
- similar
    - both support package
    - grpc uses protoc buffer, which has the same goal as Xerox to improve remote procedure call

Machine specs

CPU: 32 cores
RAM: 126GB
LAN:
IP: 10.10.1.2,
average latency: 0.133ms
bandwidth: 9.32 Gbits/sec

Exp1

1GB data, run for 3mins

value size	num keys	read-only		50% read 50% write		
		latency (ms)	throughput (ops/s)	read latency (ms)	write latency (ms)	throughput (ops/s)
512	2097152	0.277	3603	0.380	15.4	127
4096	262144	0.3210	3116	0.3985	15.72	125.4
524288	2048	4.117	242.9	4.482	25.21	67.10
1048576	1024	8.043	124.3	7.453	33.61	48.64
4194304	256	30.25	33.06	29.07	92.31	16.54

- 100% read
  - As valus size increases from 512 to 4194304, the latency grows and the throughput drops, the reason is that a server needs to package a larger string into a package, and thus increases the latency.
  - latency is reciprocal to throughput.
  - internet bandwidth is 1.165GB/s, which is not the bottleneck for this experiment.
  - We check the latency every 10 seconds, and the latency doesn't grow as time passes, so a client takes more time than a server for each operation.
- 50% 50%
  - latency of 50% is larger than 100%, the reason is from our database structure, we use sync.map for the experiment.
  - sync.map actually has two map so it can achieve an approximate multi-version, so read is usually free of lock, but sometime, we still need locks for read, when we can find the record in "read map", then we need go find the record in "dirty map", thus lock is needed.
  - write operations conflict with other operations, and the number of conflict grows as the number of key drops.

Exp 2

1M keys
4KB values

cold start latency: 40.47sec

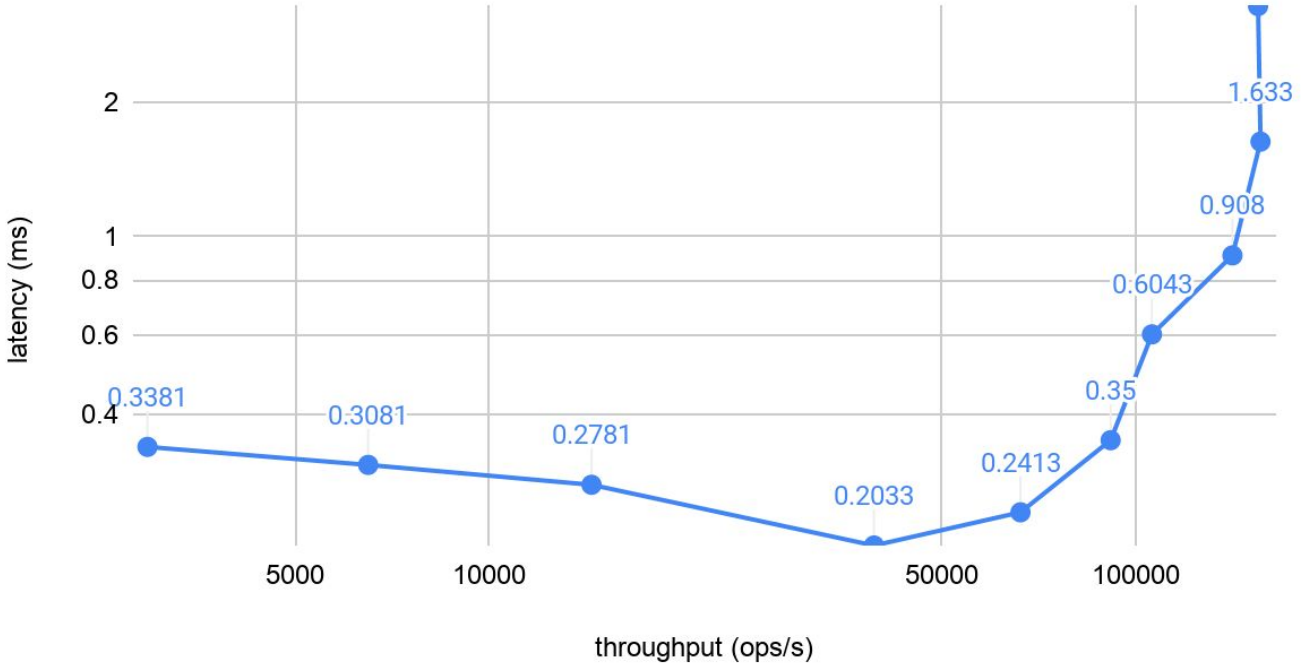
Exp3

1M keys
4KB values

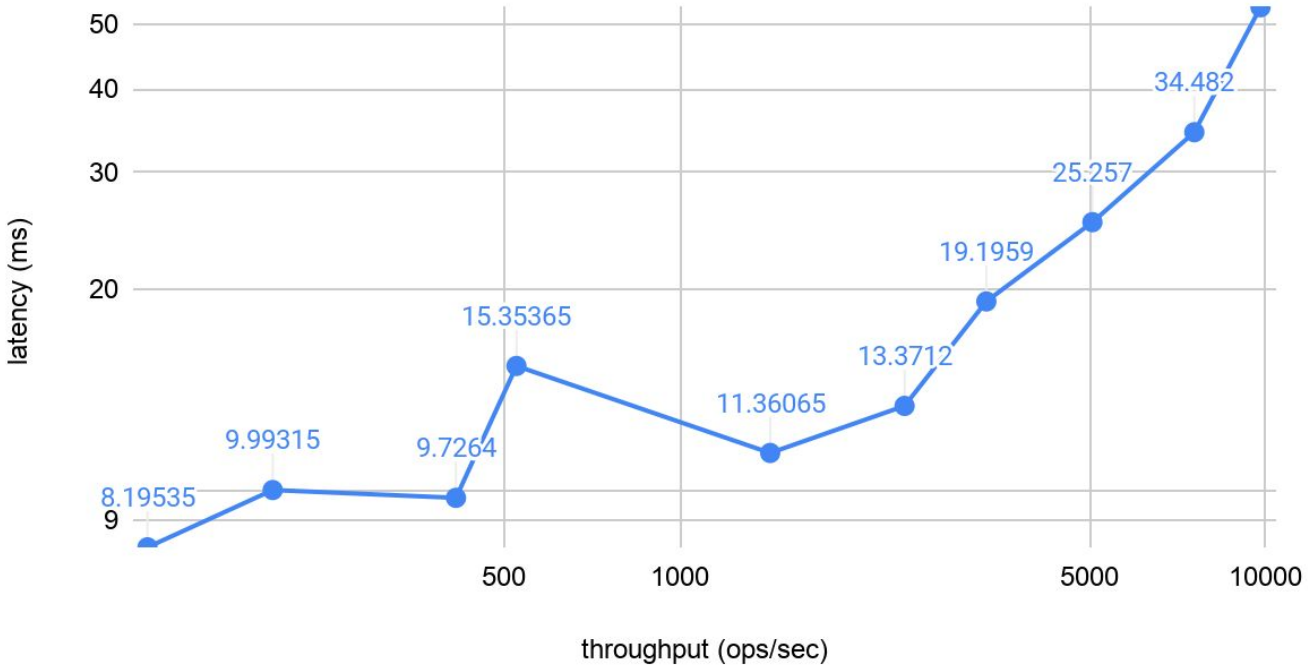
clients	read-only		50% read 50% write		
	latency (ms)	throughput (ops/s)	read latency (ms)	write latency (ms)	throughput (ops/s)
1	0.3381	2957	0.4107	15.98	123.2
2	0.3081	6491	0.4163	19.57	201.6
4	0.2781	14385	0.4128	19.04	414.3
8	0.2033	39350	0.4273	30.28	525.5
16	0.2413	66314	0.5013	22.22	1425
32	0.3500	91428	0.5824	26.16	2417
64	0.6043	105923	0.9118	37.48	3333
128	0.9080	141099	1.604	48.91	5058
256	1.633	155933	2.024	66.94	7550
512	3.286	154565	2.782	103.5	9794

- 100% read
  - The maximum throughput of the server is approximately 156000 ops/s.
  - The throughput of a client is approximately 3200 ops/s.
  - The latency grows as time passes, this reflects the fact that the overwhelming requests are queued at the server.
  - the bottleneck of server is I/O, since the CPU% of server is 65%
  - client should not be the bottleneck, since the latency grows when clients grows, if the client is the bottleneck, latency shouldn't grows
- 50 % read
  - Again, read latency grows, but doesn't grow much, because of sync.map.
  - When the total requests are larger than the server's capability, the latency grows.
  - The throughput of a client is approximately 125 ops/s.

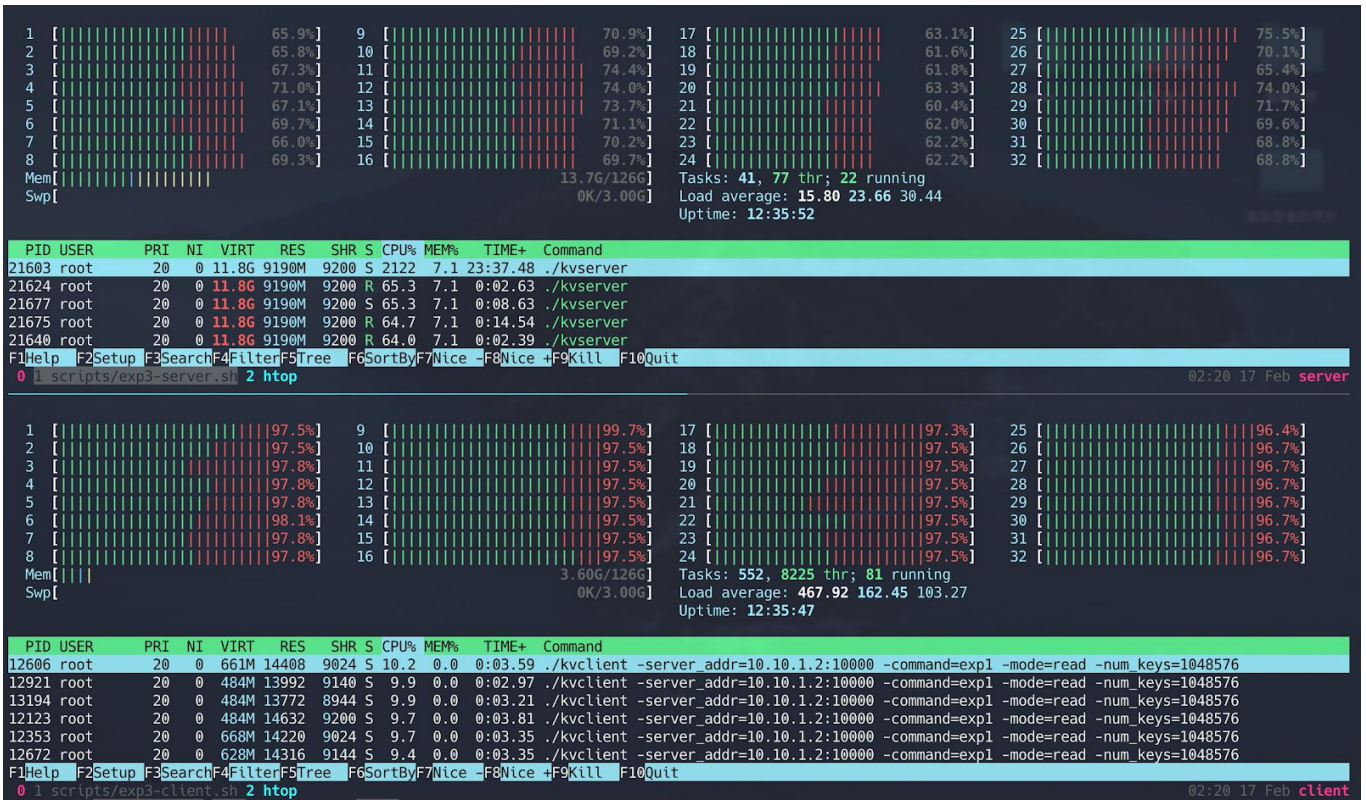
read-only latency vs. throughput



50% read 50% write latency vs. throughput



CPU usage for server and 512 clients (top: server, bottom: client):



iperf3 test link bandwidth (top server, bottom client):