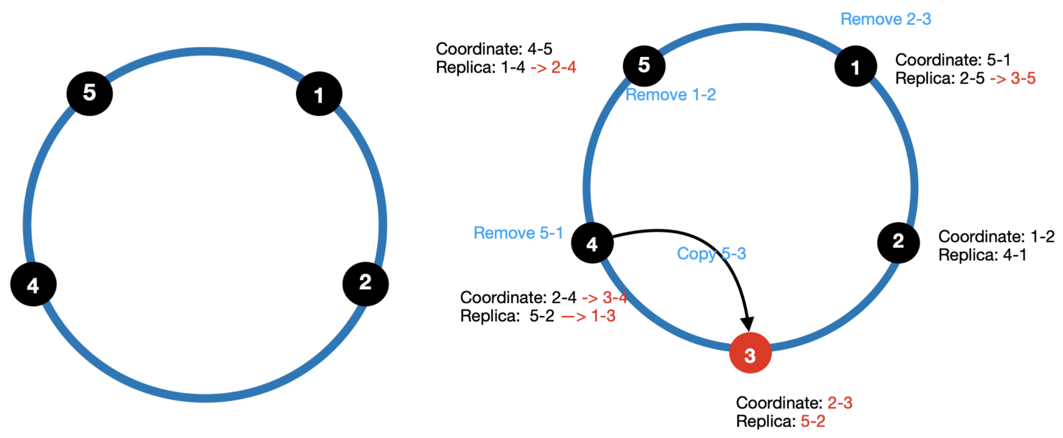
ECE419 Milestone3 Design Document

Bohao Wang 1002993442
Jingwei Zhang 1002896690
Yanyi Zhang 1003327517

1 Key Functionalities

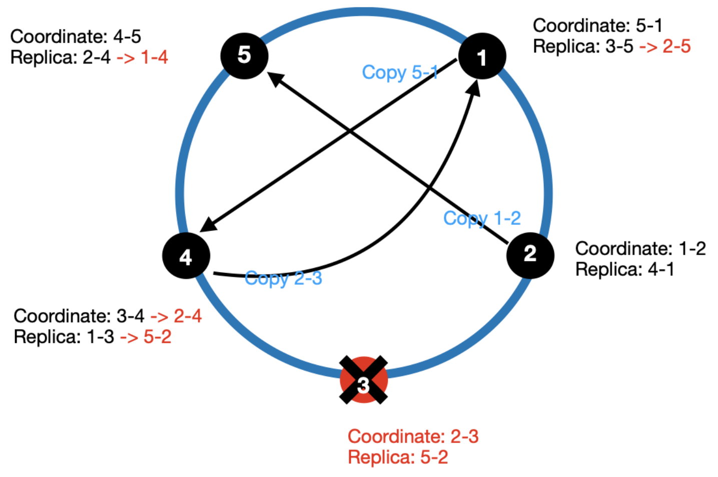
1.1 Adding Nodes



When adding a node to the hashring, each server's coordinating range and replica range should change. The following procedures will take place: 1 copy data command will be issued to move data from the new node's successor to the new node. And 3 remove data commands will be issued to the new node's 3 successor nodes. All commands will be issued by the ECS and will be executed by the corresponding servers.

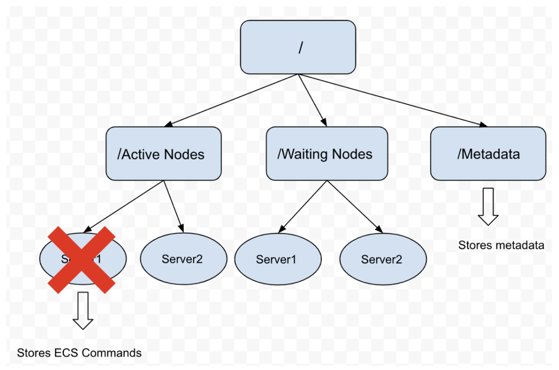
The copy command is for transferring the data to the new server and the remove command is for removing old coordinating data and replicas. Thus, all data will still have 1 coordinating server and exactly 2 replicas.

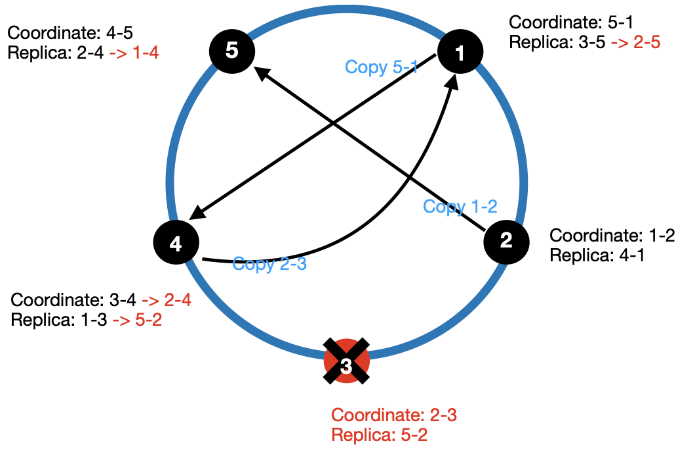
1.2 Removing Nodes



When removing a node from the hashring, 3 copy command is issued by the ECS to the removed node's 2 preceding nodes and 1 succeeding node. This will expand the coordinating range of its succeeding server and expand corresponding servers' replica range.
And we can see that after the data transfer process, all data still have exactly 2 replicas.

1.3 Node Crash and Restore





When a node crashes, the server will stop pinging the zoo keeper and the ephemeral node will disappear from the active nodes. This will trigger a ECS watch event. The ECS will treat the event as if the node has been removed and retain the data originally belonging to the crashed server. It will then randomly select another server from the configuration list and start it, after which the procedure for adding a node is triggered.
In an overview, the procedure for node crash and restore is equal to removing and adding another node.
After this process, no data will be lost and all data will still have exactly 2 replicas.

2. Component Change

ECS: We added logic to issue moveData commands to servers when node configuration event is triggered.

Clent: We added logic to prevent reconnection if the current connected server has the replica needed by PUT command.

Server: We added logic to handle moveData commands issued by the server and will update data to all replicas when receiving PUT/UPDATE commands.

3. Performance Evaluation

The performance evaluation is conducted in the same way as Milestone 2. We measured its throughput and latency by issuing 800 PUT/GET commands at the ration of 50%:50%. Generally performance for all types of hash strategy have been lowered by 10% on average. The performance loss might be caused by the replicas. The trends and patterns for different server/client configuration stays very similar to M2's performance.
Though overall performance is impacted by replicas, the performance of GET operations stays the same with slight improvements on some configurations. This is due to our connection logic. When our client is currently connected to a server with the replica, the client no longer needs to disconnect and reconnect to the coordination server.
And among these cache strategies, LFU seems to have slightly better performance, however overall the impact of cache strategy is minimal. Cache size plays a more important part in the performance side. As we increase cache size, an over 20% performance growth is witnessed during the test. Similar to M2, more servers means more performance, the throughput grows as we scale up our server group. This is due to we can better utilize the performance of all nodes when we have more servers online.

4. Test Description

Test Name	test01HashRingReplicaRange

Status	Pass
Description	Testing the functionality of calculating Replica range

Test Name	test01TestCorrectServer
Status	Pass
Description	Testing the functionality of calculating correct Server

Test Name	testPutReplicatedData
Status	Pass
Description	Testing the functionality of replicated Put request to corresponding replicas.

Test Name	testDeleteReplicatedData
Status	Pass
Description	Testing the functionality of replicated Delete request to corresponding replicas.

Test Name	testPreSendData
Status	Pass
Description	Testing the functionality of reading data to be sent from storage into KVMessage, as the preparation stage for sending data.

Test Name	testSendData
Status	Pass

Description	Testing the functionality of sending a given range of data to the targeted server.

Test Name	testDeleteData
Status	Pass
Description	Testing the functionality of deleting a given range of data from local storage.

Test Name	testNodeCrashHandling
Status	Pass
Description	Testing the detection and handling of a server crash. Make sure the crash is detected and there will be another server established to replace the crashed server.

Test Name	testAddNodeDataHandling
Status	Pass
Description	Testing the data handling during an add node event. Make sure that there are exactly 3 replicas of each data after adding the node.

Test Name	testRemoveNodeDataHandling
Status	Pass
Description	Testing the data handling during a remove node event. Make sure that there are exactly 3 replicas of each data after removing the node.

Appendix

FIFO Size 100	**5 Client/5 Server**	**5 Client 20 Server**	**5 Client 50 Server**	**20 Client/ 5 Server**	**20 Client/20 Server**	**20 Client/50 Serve**
Request/sec	3.36	56.4	144	2.97	114	127

Avg Latency (ms/request)	279	17.6	6.07	331.1	8.0	7.1
Avg Get Latency (ms/request)	251	12.7	4.27	338.6	8.8	7.9
Avg Put Latency (ms/request)	308	2.31	7.86	346	10.6	8.8

FIFO Size 500	**5 Client/5 Server**	**5 Client 20 Server**	**5 Client 50 Server**	**20 Client/ 5 Server**	**20 Client/20 Server**	**20 Client/50 Serve5**
Request/sec	5.04	67.11	2.44	3.9	133	138.9
Avg Latency (ms/request)	231.3	16.9	6.6	264.6	8.8	7.2
Avg Get Latency (ms/request)	198.2	14.9	4.1	174.4	7.5	5.1
Avg Put Latency (ms/request)	264.4	23.2	8.1	254.8	10	9.3

LRU Size 100	**5 Client/5 Server**	**5 Client 20 Server**	**5 Client 50 Server**	**20 Client/ 5 Server**	**20 Client/20 Server**	**20 Client/50 Serve**
Request/sec	3.38	47.62	147.1	3.19	119	147.0
Avg Latency (ms/request)	296	21	6.8	313.1	8.4	6.8
Avg Get Latency (ms/request)	271.5	15	4.3	300.5	6.9	5.3
Avg Put Latency (ms/request)	319.5	27	9.3	325.1	10	8.3

LRU Size 500	**5 Client/5 Server**	**5 Client 20 Server**	**5 Client 50 Server**	**20 Client/ 5 Server**	**20 Client/20 Server**	**20 Client/50 Serve**
Request/sec	4.52	61.4	185.1	4.2	120	140.84
Avg Latency (ms/request)	221.0	16.3	5.4	212.5	8.2	7.1
Avg Get Latency (ms/request)	187.1	10.0	4.1	225.2	6.2	5.6
Avg Put Latency (ms/request)	253.2	22.5	7.1	237.7	10.2	8.6

LFU Size 100	**5 Client/5 Server**	**5 Client 20 Server**	**5 Client 50 Server**	**20 Client/ 5 Server**	**20 Client/20 Server**	**20 Client/50 Serve**
Request/sec	3.31	58.82	176.05	3.47	119	153.8
Avg Latency (ms/request)	301.6	17	5.8	288.4	8.4	6.5
Avg Get Latency (ms/request)	274.0	10	4	276	97.4	4.7
Avg Put Latency (ms/request)	329.2	24	7.6	301.2	9.4	8.3

LFU Size 500	**5 Client/5 Server**	**5 Client 20 Server**	**5 Client 50 Server**	**20 Client/ 5 Server**	**20 Client/20 Server**	**20 Client/50 Serve**

Request/sec	4.38	76.92	202	4.45	156.3	172
Avg Latency (ms/request)	228.7	13	4.95	224.8	6.4	5.8
Avg Get Latency (ms/request)	194.6	9	3.4	208.6	5.9	5.2
Avg Put Latency (ms/request)	262.8	17	6.5	241	7.9	6.8

Table 1. Performance comparison with different cache sizes, strategies, number of servers and clients