MP2 Report

Linxi Li(linxili2), Gan Yao(ganyao2)

System design

The system was designed on a dynamic ring basis, for initialization, we set VM1 to be the introducer. We have to initialize VM1 first can other VMs join later. Whenever a new VM request a join, it would first Ping the introducer with join request, after introducer receives this message, it would then send the new joined VM's predecessor IP address and related information to the new VM. Then the new VM would PING its predecessor to get a full membership list, and identify its predecessor and successor, then start to work. To maintain the member of the membership list, each VM will constantly Ping its two successors and its one predecessor with period of 1 second. Once the predecessor/successor receives its ping message, it would then return a pong message to the original VM. If the VM receives the Pong within 4 seconds, then the VM would then consider its predecessor/successor to be active, and will then keep it in the membership list, this ensures a 5-second completeness since 4 seconds < 5 seconds. Otherwise, it would then remove the non-responsive VM in the membership list, and then take the next VM on the ring of its predecessor/successor to be its new predecessor/successor. All the join/fail process would update the list of membership, each update will be recorded in a list UPDATE_msgs, include those updates that are communicated from other VMs. Every time before the PING is sent out, the UPDATE_msgs will be checked, if it is not empty, these update messages will be piggybacked on PING messages.

This design would ensure that when 3 VMs fail simultaneously, even in worst case, all of the three machines of one VM listens to fails simultaneously, the VM can detect their failure. However, when 4 consecutive VMs fail simultaneously, the third machine in this "failure sequence" would not be detected because all the machines that listens to it's ping-back have already failed. When N is large, the completeness can still be ensured since one VM only needs to listen to its two successors and one predecessor. For each message, we first construct a string that follows the following format: "MessageType:MachineNumber_MachineIP_Timestamp" where one underscore represents one space, and then transform this string to bytes. With this format, we can then easily distinguish different message types, read the information of machines, either in pong and ping process, and respond according to the message type.

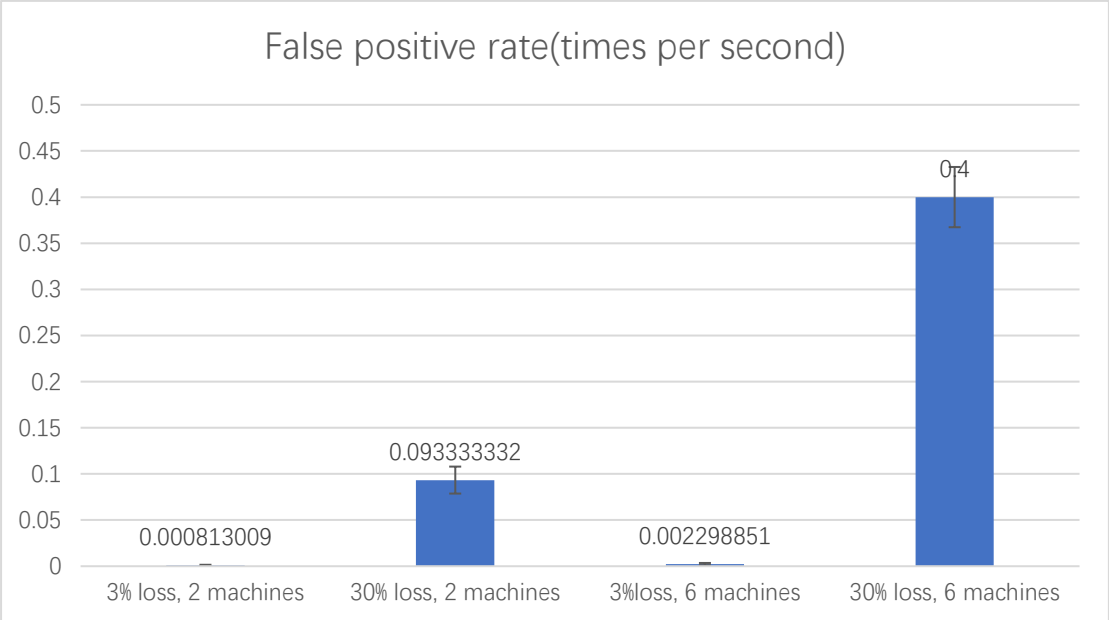
We used MP1 to debug in later phase of our development, since it can help us track all the update events in all the log files distributed on each machine.

Bandwidth usage

Type/Bandwidth(Bps)	Total Regular Bandwidth	Average Bandwidth (Leave&fail)	Average Bandwidth (Join)
Out	1152	247.5	242
In	1656	309	333
Sum	2808	556.5	575

The bandwidth usage of our system is shown in the above chart. Because we use python's bytes object to transport messages, messages sent between machines have relatively large overhead, hence bandwidth consumption is significant. The first column, 'Total Regular Bandwidth', shows the total bandwidth usage of a cluster of 6 VMs assuming no update events on membership list. The second and the third column show the average bandwidth usage of each machine(total bandwidth divided by number of machines) under the corresponding event. Note that due to our design, the 'leave' and 'fail' events are basically equivalent in our system, so they are listed together. We used linux software *iftop* to measure the bandwidth usage. Due to the limitation of the interface of this software, the measurement may not be completely accurate.

False positive rate



Our tested false positive rate is as shown in the above bar plot. For each reading, we record the number of false positive that happens in a period of 15 seconds. Each data point consists of at least 5 readings. The error line in plot represents one standard error.

The plot generally meets our expectation. When the packet loss rate is relatively low(e.g. 3%), it's hard for a machine to be falsely detected as failed. Because our timeout period for a machine to be marked as failed is four times the PING period, which means even if one pong message is dropped, there are still 3 chances left for the listened machine to tell the listener its still alive. When packet loss rate is low, it's very unlikely that the listened machine will fail to respond even once in one timeout period. But of course, when the pack loss rate increases, the false positive rate will get worse.

Also, the false positive rate increases as the cluster size grows. Assume the same packet loss rate, in a large cluster, the probability of no machine fails get smaller at exponential speed. Therefore, false positives are more likely to happen in a large cluster than in a small one.