Additional Materials for Iaso Paper Submission

(Anonymized)

1 Introduction

We make our datasets public in [1]. This document provides additional materials to our Iaso paper submission. We understand that the materials included in this document cannot be accounted for in the review process. However, we include this for interested readers/reviewers.

2 Iaso System

This section describes the cluster and machine configurations in our deployments and then uncovers the relationship between the number of fail-slow faults and these configurations.

Cluster Analysis: Table 1 gives an overview of the hardware configurations of 7 representative clusters. The machines come from the deployments at our customer environments, whose cluster size ranges from a minimum of 3 to a maximum of 56. Different cluster sizes exhibit different characteristics. First of all, the numbers of HDDs and SSDs are different due to the various cluster sizes (a roughly increasing trend). Secondly, the memory, CPU and NIC configurations are various in a certain range. Specially, the memory size is between 125GB and 512GB, the number of CPU is either 8 or 10 or 12. Likewise, the number of NIC is 2 or 4 or 6. Overall, this heterogeneity in our clusters is introduced for a couple of reasons:

- The first is system design, which needs to support heterogeneous applications which can result in nodes with different storage types (flash to storage heavy) or memory or cpu configurations.
- The second is system upgrade, which requires components replacement and upgrade. It's common that new components get added to clusters over a period of time.

To study the heterogeneity effects on the fail-slow faults, Figure **??** shows all six root causes (unknown, network, hardware, software, human and environment) among different cluster sizes. As we can see, 3-node cluster contributes to 31 fail-slow failures, which stands in the first place. One of the possible explanations is that cluster size 3 is the most common size in our deployments. Likewise, other small-scale clusters are dominant factors of fail-slow failures. Overall, sizes 3-8 take up 85% faults, but sizes 9-56 just have 15%. Meanwhile, fail-slow causes are various among these clusters. First of all, network and hardware are the most common causes for different cluster sizes. There are 8 and 9 clusters sizes involve network and hardware causes, respectively. Secondly, software failures only occur in small scale cluster (#N$<$10), one possibility is that large-scale clusters have more redundancies to tolerate software failures. Finally, human errors distribution is random, which just matches with its own random characteristics.

ID	#Node	#HDD	#SSD	MEM	#CPU	#NIC
1	3	6	6	125	8	6
2	4	14	8	-	8/12	2/4
3	5	0	30	384	12	4
4	6	36	12	512	8	6
5	16	64	28	-	8/12	4
6	19	66	27	256	8/10	4
7	26	104	52	256	8/12	4/6

Table 1: **Hardware configurations of a sample set of clusters.** *"-" means a combination of different memory sizes (unit GB, e.g, cluster 2 contains 128GB, 256GB, and 512GB memory; cluster 5 only contains 128GB and 256GB memory).*

Machine Analysis: In addition to above cluster characteristics, our machine nodes show different hardware characteristics. In our deployments, we notice some machine nodes with similar configurations, which are possible to be in the same cluster or different clusters. Regarding this, we introduce a new term "node model family" to represent them. Table 2 shows the different node model family configurations, where A-H have different processor, memory, HDD, SSD and network adaptor configurations. In our experience, one node model family typically contains 3 to 8 nodes. This small scale difference motivates us to further investigate the node family's hardware configuration effects on the fail-slow failures. Figure **??** shows the failure distribution for different node families, where family B has the most failures and family D vice versa. The interesting thing is family B represents the most storage heavy family but family D is opposite. From this point, we can figure out the correlated relationship between the number of failures and node families. In addition, when it comes to the failure causes, all of them are common. This might be a clue of no significant correlation

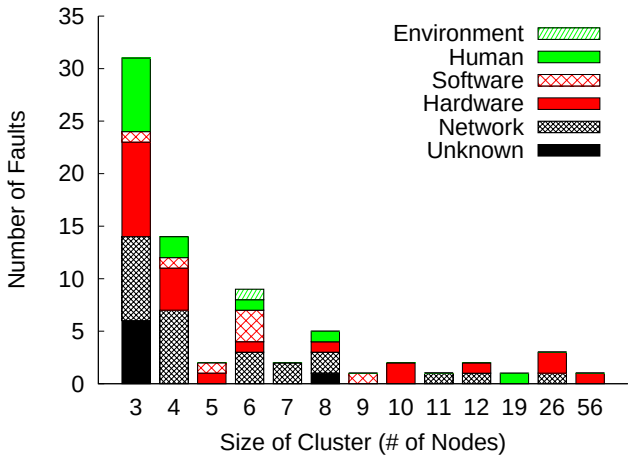

Figure 1: **fig-faultCluster: Fail-stutter faults across different cluster sizes.**

between the fail-slow failures causes and node family models.

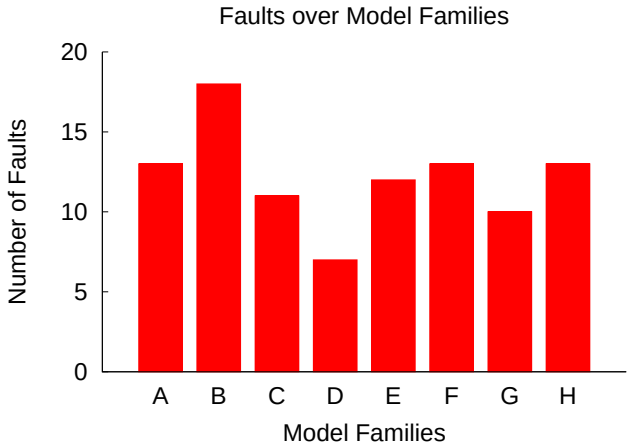

Figure 2: **fig-faultModels: Fail-stutter faults across model families.**

3 False Positives and Negatives

3.1 False Positives

False positive is to mark a good/down node as fail-slow. This section will introduce two false positive stories.

Case1 In our Zookeeper score reporting system, one node randomly picks up about 7 peer nodes to monitor for every 2 minutes. The node will send RPC requests to these peer nodes and measure the RPC round trip time, which will be assigned as a score. Once the RPC request times out, the node will get a default score 2000ms, which is the default timeout value. If the peer node L couldn't respond in 2000ms, the node continues keeping a sliding window of the last 10 scores for the peer node L, which looks like {2000, 2000, ... , 2000}. In the ScoreDB, if 70 percentile scores for peer node L are 2000ms, our score detection system will mark the node as down instead of fail-slow. Even if a potential fail-slow is alive, the score analysis system waits for the peer node to be up for a certain time (e.g., 4mins in our system) before marking it as fail-slow. This ensures that if the node received bad scores due to the node is down or just rebooted, we don't label it as fail-slow. In this case, our score analysis system waits for the peer node L to receive good scores in the next 4 minutes. However, if none of the nodes choose the peer node L to monitor (randomly picks up 7 nodes), then our score analysis system will never get any good score for the peer node L and incorrectly mark it as fail-slow.

Case 2 In our score reporting system, the Zookeeper instance on one node is unreachable due to the degraded NIC. Then, many services will restart if they lose the Zookeeper sessions. In this case, other peers of the degraded node restarted, which triggers the lose of their previous state of the number of timeouts and responses. Then the node for which this node was replica for, started to timeout sporadically because it could not reach the degraded node, leading the healthy node to receive higher scores and eventually being marked as degraded.

3.2 False Negatives

False negative is not to mark a degraded node as fail-slow. This section will describe three false negatives cases.

Case 1 Our score reporting and analysis systems highly reply on the ScoreDB. The former stores the reported peer scores in the ScoreDB and the latter analyzes these scores to detect the outlier. In our Zookeeper score reporting system, if the Zookeeper leader is a fail-slow node (e.g., CPU soft lockup), it leads to the write pipeline blocked. However, Zookeeper's internal ping mechanism bypasses the write pipeline, which makes the followers not detect that they have to detach from the fail-slow leader. In this case, Zookeeper is not available and then the score reporting and analysis systems could not work well.

Case 2 In our Cassandra score reporting system, there is a node with network packet loss issues due to a faulty NIC, which will cause the Cassandra instance on that node to timeout most requests from its peers. And this degraded node has not provided even a single successful response. Hence this node was not part of the set of nodes for which the peers reported scores for. In this case, it seriously leads to the cluster

Model Family	Processor	Memory	HDD	SSD	Network Adaptor
A	2.40GHz 6-cr Haswell E5-2620 v3 15M Cache	96	4TB 3.5”	800GB 3.5-C”	10GbE Dual SFP+
B	2.4GHz 10-cr Brdwl E5-2640 v4 25M Cache	512	6TB 3.5”	1.6TB 3.5”	1GbE Dual Base-T
C	1.7GHz 8-cr Brdwl E5-2609 v4 20M Cache	64	2TB 3.5”	480GB 3.5”	10GbE Dual SFP+
D	2.1GHz 8-cr Brdwl E5-2620 v4 20M Cache	64	2TB 3.5”	480GB 3.5”	10GbE Dual SFP+
E	2.60GHz 10-cr Haswell E5-2660 v3 25M Cache	384	2TB 2.5”	1200GB 2.5-C”	10GbE Dual SFP+
F	2.4GHz 14-cr Brdwl E5-2680 v4 35M Cache	512	2TB 3.5”	1.2TB 2.5”	10GbE Dual SFP+
G	3.50GHz 4-cr Haswell E5-2637v3 15M Cache	512	6TB 3.5”	1600GB 3.5-C”	10GbE Dual SFP+
H	2.4GHz 10-cr Brdwl E5-2640 v4 25M Cache	256	6TB 3.5”	1.2TB 2.5”	10GbE Dual SFP+

Table 2: **tab-family: Hardware configurations of a sample set of node model families.**

outage because the actual degraded node was not quarantined.

Case 3 The Cassandra score reporting system is dependent on the uniformity of requests to neighbors based on its DHT nature. In the case of idle cluster, if the number of requests and responses is not sufficient high, the degraded node may receive less or even zero IO compared to other good nodes. This leads to other nodes to slow down and falsely marks one of them as degraded.

4 Versions

Our versions v1 to v5 merely correspond to our software release cycles. Based on the current release, we stages our changes and fixes. The following is a short summary of the description of the versions and the justification on why it is needed to improve the previous version.

- **V1 (4.5)** This version is mainly for evaluation. It contains score reporting and score analysis, but the actions to quarantine a degraded node are disabled. We are able to collect the degraded node scores from customer clusters and check whether the observed high scores (if any) were indicative of real problems.
- **V2 (4.6)** This version fixes some implementation related issues, which mostly exist in the score analysis algorithm.
- **V3 (4.7)** This version acts as four roles: (1) it prevents Cassandra from performing node addition if the node was marked degraded; (2) it aligns the interval to measure local disk write latency in Zookeeper with the peer rpc interval; (3) it does not mark a node as degraded if our fault tolerance (FT) is already 0 in a cluster which supports the FT of 1; (4) it fixes an implementation bug to discontinue isolation actions if the degraded flag has been unset.
- **V4 (5.0)** This version moves degraded Cassandra node to a forwarding mode (but disable via flag).

Year	All Outages	Fail-slow Outages	With Iaso
2014	8	2	-
2015	11	2	-
2016	19	12	-
2017	23	8	7
2018	12	1	38

Table 3: **Comparisons of outages (§5).** *The table shows the number of outages before and after the deployment of Iaso (columns 2 and 4) and the number of outages caused by fail-slow failures (column 3)*

- **V5 (5.1)** This version solves three : (1) it fixes a bug in C* score reporting that is exposed when a node has no degraded scores if it starts up in such an unhealthy state that it has 0 healthy response; (2) it enables degraded node detection and takes degraded node action by default; (3) it modifies Zookeeper’s selection logic to add weights while selecting peers to monitor. Zookeeper doesn’t select all other peers to monitor, it is possible that some peers never got selected by anyone in a particular window, which leads to a false positive.

5 Past Outages

The outages in our clusters occur now and then. Table 3 shows the outage frequencies in the past 5 years. As we can see, the number of outages is increasing every year (e.g., from 8 to 23). One of the reasons is that there are more nodes/clusters deployed, which increases the possibility to have more outages. However, we have deployed Iaso for more than 1.5 years in our customers sites (from 2017), which makes the number of outages decrease to some extend in 2017 and 2018.

Once outages are reported, our site-reliability engineers (SREs) perform the full diagnosis. Some of the outages are due to fail-slow incidents but some are not. For example, in 2014, there are 8 full outages (IOPS went to zero), just 2 of which are caused by fail-slow.

With Iaso deployment, the number of fail-slow outages

PID	MEM	CPU
1	23MB	0.1
2	55MB	0.2

Table 4: **Iaso Overhead (§6).** *The table shows the memory and cpu consumption before and after the Iaso deployment*

started to decrease. Since Iaso was deployed in 2017, the number of outages in 2016 and 2017 would be representative to illustrate the Iaso effectiveness. As we see, there are 12 fail-slow outages in 2016 but just 8 fail-slow outages in 2017. Meanwhile, with the development of Iaso, we just have one fail-slow outage in 2018.

6 Overhead

The overhead of our score reporters is mostly negligible as they piggyback on existing score collection metrics used by the application. We will illustrate the overhead regarding to the memory and cpu consumptions in Table 4, where PID 1 is a Zookeeper monitor process that doesnt run the score analysis algorithm (we run it on multiple nodes but not all nodes, only the ones that are Zookeeper nodes). On the contrary, PID 2 is a Zookeeper monitor process that does the score analysis (as part of the Zookeeper monitor process). Regarding to the resident memory consumed (RES), PID 1 is around 23MB, but PID 2 is 55MB. The thing to note here is that the nodes on which we do score analysis as part of the Zookeeper monitor process also do a couple of other unrelated things. Therefore, the 55MB is still like an upper bound. When it comes to CPU consumption, we can see the CPU used for PID 1 is 0.1 and for PID 2 is 0.2, which are both negligible.

7 Future Work

In our paper, we have described a generic design to build a heuristic-based fail-slow fault tolerant distributed application. As with majority of such systems, we have certain areas that can be improved. A few of those are as follows:

- **Tuning fail-slow node detection interval:** Currently we have conservatively chosen the fail-slow node detection time interval to be 10 minutes based on internal testing, mainly because of the of the high impact of false positives. However, we could safely determine the presence of a fail-slow fault in less than 10 minutes. Given that we now have the feature deployed in the field for a large number of clusters, we intend to mine the peer scores reported in the field and internally run our score processing engine on them with various fail-slow node detection intervals. Once we obtain the false positive rate for various fail-slow node detection intervals, we will be able to pick a more aggressive value as our fault detection interval, and detect and mitigate the effect of a fail-slow node quickly.

- **Enable score reporting for other services:** Currently only the Metadata Service and the Cluster Configuration Manager report scores in our system. Since the causes for a fail-slow fault can be diverse and varied, we intend to enable score reporting for other critical services such as our cluster block storage system and the background cluster management service.

- **Automatically detect fail-slow node recovery:** Currently we depend on the user to mark that the fail-slow fault has been resolved in the cluster configuration. Instead, we intend to allow the peers of the fail-slow node to continue reporting scores for the faulty node, and use them to determine whether the node has recovered. This will help if the fail-slow fault was a transient network failure. In such cases, being able to automatically detect the recovery of a fail-slow node will be highly useful.

References

[1] Iaso Datasets. https://tinyurl.com/Iaso-datasets, 2019.