Item 286
git_comments:

git_commits:

1. **summary:** Log Collation - Memory leak when all hosts are down.
   message: Log Collation - Memory leak when all hosts are down. This patch is intended to address issue #2416. When all hosts are down, the log output is supposed to be written to an orphan file. The LogHost::orphan_write_and_try_delete() is supposed to consume a LogBuffer reference count, but it only does so in one branch of the if/then/else instead of both. This patch fixes this issue.
   label: code-design

github_issues:

1. **title:** memory leak - unreachable log collation host
   body: This memory leak is the result of configuring a log collation host in “logs_xml.config” and then having the host be down during the traffic run. We verified that this occurs with vanilla 6.2.1. We also noticed that there was a memory leak fix in this function patched in TS-4872, which is present in 6.2.x, 7.1.x, and master. So we tried the latest 6.2.x (current) from yesterday, but the memory leak still occurred. We have also seen that bringing up the log host during the test run stabilizes the memory allocation so we’re pretty sure that this is the trigger for the memory leak. Some heap-check output from ATS compiled with tcmalloc -- Total: 3014.2 MB 2902.3 96.3% 96.3% 2902.3 96.3% ats_malloc 76.0 2.5% 98.8% 76.0 2.5% ats_memalign 35.8 1.2% 100.0% 2938.1 97.5% LogObject::_checkout_write 0.1 0.0% 100.0% 0.1 0.0% BaseLogFile::open_file 0.0 0.0% 100.0% 0.0 0.0% ResourceTracker::lookup (inline)
   label: code-design
2. **title:** memory leak - unreachable log collation host
   body: This memory leak is the result of configuring a log collation host in “logs_xml.config” and then having the host be down during the traffic run. We verified that this occurs with vanilla 6.2.1. We also noticed that there was a memory leak fix in this function patched in TS-4872, which is present in 6.2.x, 7.1.x, and master. So we tried the latest 6.2.x (current) from yesterday, but the memory leak still occurred. We have also seen that bringing up the log host during the test run stabilizes the memory allocation so we’re pretty sure that this is the trigger for the memory leak. Some heap-check output from ATS compiled with tcmalloc -- Total: 3014.2 MB 2902.3 96.3% 96.3% 2902.3 96.3% ats_malloc 76.0 2.5% 98.8% 76.0 2.5% ats_memalign 35.8 1.2% 100.0% 2938.1 97.5% LogObject::_checkout_write 0.1 0.0% 100.0% 0.1 0.0% BaseLogFile::open_file 0.0 0.0% 100.0% 0.0 0.0% ResourceTracker::lookup (inline)
3. **title:** memory leak - unreachable log collation host
   body: This memory leak is the result of configuring a log collation host in “logs_xml.config” and then having the host be down during the traffic run. We verified that this occurs with vanilla 6.2.1. We also noticed that there was a memory leak fix in this function patched in TS-4872, which is present in 6.2.x, 7.1.x, and master. So we tried the latest 6.2.x (current) from yesterday, but the memory leak still occurred. We have also seen that bringing up the log host during the test run stabilizes the memory allocation so we’re pretty sure that this is the trigger for the memory leak. Some heap-check output from ATS compiled with tcmalloc -- Total: 3014.2 MB 2902.3 96.3% 96.3% 2902.3 96.3% ats_malloc 76.0 2.5% 98.8% 76.0 2.5% ats_memalign 35.8 1.2% 100.0% 2938.1 97.5% LogObject::_checkout_write 0.1 0.0% 100.0% 0.1 0.0% BaseLogFile::open_file 0.0 0.0% 100.0% 0.0 0.0% ResourceTracker::lookup (inline)
   label: code-design
4. **title:** memory leak - unreachable log collation host
   body: This memory leak is the result of configuring a log collation host in “logs_xml.config” and then having the host be down during the traffic run. We verified that this occurs with vanilla 6.2.1. We also noticed that there was a memory leak fix in this function patched in TS-4872, which is present in 6.2.x, 7.1.x, and master. So we tried the latest 6.2.x (current) from yesterday, but the memory leak still occurred. We have also seen that bringing up the log host during the test run stabilizes the memory allocation so we’re pretty sure that this is the trigger for the memory leak. Some heap-check output from ATS compiled with tcmalloc -- Total: 3014.2 MB 2902.3 96.3% 96.3% 2902.3 96.3% ats_malloc 76.0 2.5% 98.8% 76.0 2.5% ats_memalign 35.8 1.2% 100.0% 2938.1 97.5% LogObject::_checkout_write 0.1 0.0% 100.0% 0.1 0.0% BaseLogFile::open_file 0.0 0.0% 100.0% 0.0 0.0% ResourceTracker::lookup (inline)
   label: code-design

github_issues_comments:

1. **body:** Confirmed this memory leak still occurs with 7.1.3.
   label: code-design
2. Collation is gone, so closing this.

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** clang-analyzer: Memory leak LogObject.cc _checkout_write
   description: We could not quite figure out why it reports this (smells almost like a bug in clang-analyzer), but [~jpeach@apache.org] has a refactoring that avoids the warning, and cleans up the code overall.

jira_issues_comments: