Item 70
git_comments:

1. keep this runner alive
2. remove it from the list of running things unless we are the last runner and the queue is full... in which case, the next queue.put() would block and there would be no runners to handle it.

git_commits:

1. **summary:** SOLR-1711: fix hang when queue is full but there are no runners
   message: SOLR-1711: fix hang when queue is full but there are no runners git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/trunk@1063869 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Race condition in org/apache/solr/client/solrj/impl/StreamingUpdateSolrServer.java
   description: While inserting a large pile of documents using StreamingUpdateSolrServer there is a race condition as all Runner instances stop processing while the blocking queue is full. With a high performance client this could happen quite often, there is no way to recover from it at the client side. In StreamingUpdateSolrServer there is a BlockingQueue called queue to store UpdateRequests, there are up to threadCount number of workers threads from StreamingUpdateSolrServer.Runner to read that queue and push requests to a Solr instance. If at one point the BlockingQueue is empty all workers stop processing it and pushing the collected content to Solr which could be a time consuming process, sometimes all worker threads are waiting for Solr. If at this time the client fills the BlockingQueue to full all worker threads will quit without processing any further and the main thread will block forever. There is a simple, well tested patch attached to handle this situation.

jira_issues_comments:

1. **body:** Patch 1, 2: Inside the Runner.run method I've added a do while loop to prevent the Runner to quit while there are new requests, this handles the problem of new requests added while Runner is sending the previous batch. Patch 3 Validity check of method variable is not strictly necessary, just a code clean up. Patch 4 The last part of the patch is to move synchronized outside of conditional to avoid a situation where runners change while evaluating it. To minify the patch all indentation has been removed.
   label: code-design
2. Thanks Attila! I just committed this.
3. This is a very serious problem for us. We have multiple threads adding to the StreamingUpdateSolrServer's BlockingQueue, and if I bump the thread count high enough (around 10 for my process) I can reproduce this problem every time. I'd say this bug is critical enough to warrant a SOLR bug-fix release.
4. Correcting Fix Version based on CHANGES.txt, see this thread for more details... http://mail-archives.apache.org/mod_mbox/lucene-dev/201005.mbox/%3Calpine.DEB.1.10.1005251052040.24672@radix.cryptio.net%3E
5. Committed revision 949473. merged to branch-1.4 for 1.4.1
6. We are still seeing the same issue with Solr1.4.1 We get into this situation when all the runner threads die due to a broken pipe, while the BlockingQueue is still full. All of the producer threads are all blocked on the BlockingQueue.put() method. Since the runners are spawned by the producers, which are all blocked, runner threads never get created to drain the queue. Here's a potential fix. In the runner code, replace these lines: // remove it from the list of running things... synchronized (runners) { runners.remove(this); } with these lines: // remove it from the list of running things unless we are the last runner and the queue

is full... synchronized (runners) { if (runners.size() == 1 && queue.remainingCapacity() == 0) { // keep this runner alive scheduler.execute(this); } else { runners.remove(this); } }
7. Thanks Johannes, the fix does look correct, and I've committed to 3x and trunk. If we have another release of 1.4, we should backport this.
8. We are still seeing this issue even after using Johannes fix. All runners are exiting and the main producer thread hangs on line 196 queue.put . I am thinking it may be because queue is getting drained and filled fast (queue size 50 , number of threads 20) . So there might be a race condition on the queue capacity check.Queue appears to be below capacity to the last runner then fills up by simultaneous calls to put . I still see the issue after backporting what is in 3.x branch for testing it with solr 1.4.1. I guess a solution may be to use larger queue capacities for now but the race conditions still seem to be present.
9. bq. So there might be a race condition on the queue capacity check. Yeah. What about moving the queue.put() inside the synchronized(runners) block to fix this?
10. **body:** bq. What about moving the queue.put() inside the synchronized(runners) block to fix this? On second thought, that looks like a pretty bad idea ;-) Looks like a recipe for deadlock since the runners lock will be held if put then blocks.
    label: code-design
11. Here's a patch that uses offer instead of put in a retry loop.
12. Committed the latest patch - hopefully that finally fixes this issue!
13. **body:** I tried out the comitted patch with SOLR 1.4.1 (built SOLR on my own). Unfortunately I still have an issue when having the following scenario: Server: Runs SOLR servlet Client: Adds documents to the index using the StreamingUpdateSolrServer in several threads. If the server crashes or becomes unreachable for the client, the StreamingUpdateSolrServer on the client will end up in an infinite loop, trying to send requests to the server. It should be possible to stop/shutdown/dispose the StreamingUpdateSolrServer somehow from another thread. Clearing the queue and stopping the executor service from outside stopps the infinite loop.
    label: code-design
14. Bulk close for 3.1.0 release