Item 198
git_comments:

1. this part is for resolving dependency convergence error for zookeeper

git_commits:

1. **summary:** Upgrade zookeeper version to 3.5.8 (#6558)
   message: Upgrade zookeeper version to 3.5.8 (#6558)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Upgrade zookeeper version to 3.5.8
   body: ## Description Per https://github.com/apache/incubator-pinot/issues/6554, upgrade zookeeper version from 3.4.11 to 3.5.8 to fix: [ZOOKEEPER-2184] (https://issues.apache.org/jira/browse/ZOOKEEPER-2184) : Zookeeper Client should re-resolve hosts when connection attempts fail. Note: - Found an issue on Zookeeper v3.4.13 running on JDK 14+ with an unresolvable hostname issue, so directly jump to 3.5.8. - From zookeeper 3.5.x, `org.apache.zookeeper.server.ZooKeeperServerMain` added an AdminServer, so there is one more exception in the throw list. - From zookeeper 3.5.x, ZK client will fail the initialization immediately if ZK server is unreachable, so ZkStarter needs to catch the exception and retry. ## Upgrade Notes Does this PR prevent a zero down-time upgrade? (Assume upgrade order: Controller, Broker, Server, Minion) * [ ] Yes (Please label as **<code>backward-incompat</code>**, and complete the section below on Release Notes) Does this PR fix a zero-downtime upgrade introduced earlier? * [ ] Yes (Please label this as **<code>backward-incompat</code>**, and complete the section below on Release Notes) Does this PR otherwise need attention when creating release notes? Things to consider: - New configuration options - Deprecation of configurations - Signature changes to public methods/interfaces - New plugins added or old plugins removed * [ ] Yes (Please label this PR as **<code>release-notes</code>** and complete the section on Release Notes) ## Release Notes If you have tagged this as either backward-incompat or release-notes, you MUST add text here that you would like to see appear in release notes of the next release. If you have a series of commits adding or enabling a feature, then add this section only in final commit that marks the feature completed. Refer to earlier release notes to see examples of text ## Documentation If you have introduced a new feature or configuration, please add it to the documentation as well. See https://docs.pinot.apache.org/developers/developers-and-contributors/update-document

github_pulls_comments:

github_pulls_reviews:

1. Where is `AdminServer.AdminServerException` thrown from?
2. Same here
3. Don't quite follow this part. Why do we connect 10 times? It will sleep for at least 10 seconds
4. From ZooKeeperServerMain, it's added in zookeeper 3.5.x
5. the reason is that new client will fail and throw exception if zk server is unreachable unlike the previous behavior to wait to connect. So we need to handle the catch exception and retry here.
6. In that case, we should break the retry if it connects successfully? Also throw exception if all 10 retries failed?
7. ah, yes! added a break after the `client.close()`
8. Throw exception when all 10 retries failed?
9. yes.

jira_issues:

1. **summary:** Zookeeper Client should re-resolve hosts when connection attempts fail
   description: Testing in a Docker environment with a single Kafka instance using a single Zookeeper instance. Restarting the Zookeeper container will cause it to receive a new IP address. Kafka will never

be able to reconnect to Zookeeper and will hang indefinitely. Updating DNS or /etc/hosts with the new IP address will not help the client to reconnect as the zookeeper/client/StaticHostProvider resolves the connection string hosts at creation time and never re-resolves. A solution would be for the client to notice that connection attempts fail and attempt to re-resolve the hostnames in the connectString.

jira_issues_comments:

1. [~bhavanki]: Hi Bill, I want to try and fix this defect and would like to ask you for a review on my proposal. *What is the issue?* ZK client resolve the host name to ip's when it starts i.e., not when it tries to reconnect to the ZK cluster. When restarting ZooKeeper Dockers containers - the ip address of the ZooKeeper server might change (at least this is my understanding), causing the client fail reconnecting to the cluster. *Proposed fix:* Have _StaticHostProvider.next(int)_ resolve the host addresses of serverAddresses at the start of the method. This method is called from _SendThread.startConnect()_ and _SendThread.pingRwServer()_ which are used for re-connections flow. Any thoughts?
2. I've modified StaticHostProvider to track whether 'next()' has been called twice in a row without an intervening call to 'onconnected()' and attempt to re-resolve the IP address for the (apparent) connection failure.
3. The submitted patch is against the branch-3.4 branch only.
4. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12768117/ZOOKEEPER-2184.patch against trunk revision 1709293. +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 7 new or modified tests. -1 patch. The patch command could not apply the patch. Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-Build/2924//console This message is automatically generated.
5. Hi [~rthille], Thanks for the patch. I have a few comments and concerns about this patch: # I think it would be better to not track connectedSinceNext and try to resolve addresses again after spinning over the list of servers, perhaps right after where we do a Thread.sleep(spinDelay), line 111 after applying your patch. # I'm not sure why you deleted that block of code in the constructor of StaticHostProvider. # The change in ZxidRolloverTest.java doesn't seem to be part of this patch.
6. Cancelling due to comments.
7. For "1" is the change you suggest in order to reduce the latency of going to the 'next()' server? That makes sense. Ideally, I'd love to kick off a thread to do the resolution and immediately return the next server, but I'm a C/Python programmer, not a Java programmer, so I'm not going there :-) For 2, I'll have to re-run through it when I get a chance (later today probably), but I believe that that code converts hostnames to IP addresses, so later on we don't have the original hostnames in order to re-resolve. For 3, yeah, I think I changed that because I was seeing the ERROR() output for something that was obviously expected and not an error and that was distracting me from the real errors I was seeing during development. I'll remove that from the next patch.
8. [~fpj], [~rthille]: moving this to 3.4.8 (happy to help with the review when you've sorted the above comments)
9. I am moving this out to 3.4.10 for now.
10. I'm sorry for losing track of this issue, it is important and not hard to fix, we should do it soon.
11. [~rthille] Any chance you want to pick back up on this where you left off? If not do you mind if run with it?
12. GitHub user fpj opened a pull request: https://github.com/apache/zookeeper/pull/150 ZOOKEEPER-2184: Zookeeper Client should re-resolve hosts when connection attempts fail This is a version of the patch for ZK-2184 for the 3.4 branch, compatible with Java 6. You can merge this pull request into a Git repository by running: $ git pull https://github.com/fpj/zookeeper ZK-2184 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/zookeeper/pull/150.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #150 ---- commit fbaa47e3a96f166cfae45070b0724780f13714e9 Author: fpj <fpj@apache.org> Date: 2017-01-14T16:58:15Z ZOOKEEPER-2184: Zookeeper Client should re-resolve hosts when connection attempts fail ----
13. Github user fpj commented on the issue: https://github.com/apache/zookeeper/pull/150 The error is expected because we haven't setup QA to build out of the 3.4 branch.
14. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96120333 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,12 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses)

Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { --- End diff -- Silly nit: space around expression.

15. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96120303 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,12 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the --- End diff -- This line could be part of the @return tag, no?
16. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96120265 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -45,6 +45,9 @@ private int currentIndex = -1; + // Don't re-resolve on first next() call + private boolean connectedSinceNext = true; --- End diff -- Sincere question: it is worth making this field *volatile*?
17. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96120356 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,12 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.indexOf(':')); + } + --- End diff -- Extra space?
18. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96121135 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,12 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.indexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch

(UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); + } + } + } ++currentIndex; + connectedSinceNext = false; if (currentIndex == serverAddresses.size()) { --- End diff -- As `serverAddresses.size()` cannot be 0 (per constructor checking) this if condition and line 137 could be rewritten as: `` currentIndex = currentIndex % serverAddresses.size(); `` or even `` currentIndex = ++currentIndex % serverAddresses.size(); `` Eliminating the need of line 137 too. On master, branch-3.4 and branch-3.5. **Just a silly optimization, tough.**

19. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96125997 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -45,6 +45,9 @@ private int currentIndex = -1; + // Don't re-resolve on first next() call + private boolean connectedSinceNext = true; --- End diff -- All calls to `next` and `onConnected` are from the `sendThread`. I don't see a reason for making volatile, unless we are doing it defensively. Let me know if I'm missing anything. Note that this pull request is for the 3.4 branch, we need a different patch for 3.5 and master.
20. I took a stab at a pull request for the 3.4 branch and I'd appreciate some feedback. It is easy to miss some important detail with this hostname/address manipulation, so if anyone has a chance to double check what I've done, I'd appreciate. For the 3.4 branch, we promise compatibility with Java 6, so the patch does not immediately apply because it uses {{getHostString}}, which is available only from Java 7. Consequently, I have added some code to get around that. For the 3.5 branch, we will need a different patch because of the reconfiguration changes to {{StaticHostProvider}}. I'll work on it once the 3.4 patch gets a +1.
21. Github user rakeshadr commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96163423 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch (UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); --- End diff -- Please use {} instead of string concatenation. Also, for better debugging, can we pass 'e' object as argument to the logger instead of concat the exception message.
22. Github user rakeshadr commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96163440 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; +

InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); --- End diff -- I failed to find any test case which covers the newly added condition, could you please point me to that. Thanks!

23. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96170041 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch (UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); --- End diff -- Should we create an unresolved address like what we did on server side `recreateSocketAddresses` if we can't resolve the address? If the address is not resolvable but maybe it is still usable?
24. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96170084 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { --- End diff -- Java 6 has getHostString, but it's package private. Use reflection can access that. Maybe we should use the library version instead?
25. Github user hanm commented on the issue: https://github.com/apache/zookeeper/pull/150 Does C client has similar issue? Also, ZOOKEEPER-2184 looks like a dup of ZOOKEEPER-338.
26. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96226819 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -45,6 +45,9 @@ private int

currentIndex = -1; + // Don't re-resolve on first next() call + private boolean connectedSinceNext = true; - -- End diff -- My reasoning was only defensively, but on a 2nd look, it would be overkill. So, no problem without being volatile, imo.

27. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96227005 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch (UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); --- End diff -- +1
28. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96431605 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,12 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the --- End diff -- I've added a phrase to the return tag.
29. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96433850 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort =

curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch (UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); --- End diff -- I don't think I can do both, see the API docs of slf4j: https://www.slf4j.org/api/org/slf4j/Logger.html In the case you are, tell me which one is your favorite. I'd say the curly braces.

30. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96439377 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { --- End diff -- We can do it, but I'm not super convinced we should because we are essentially using a method with undocumented API. Perhaps it does the same as the one in Java 7, with the difference that it is public, but I'm worried that there could be some correctness issue involved. Do you know more about it? In any case, I'm going to push the changes so that we see how it looks like.
31. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96442215 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch (UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); --- End diff -- If the `StaticHostProvider` constructor didn't throw an `UnknownHostException`, then I'd think that all names and addresses we have are good. I'm not sure what could cause an `UnknownHostException` in `next()` other than some transient error. If that's right, then I'm not sure we should be adding or removing anything.
32. Github user rakeshadr commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96453176 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but

earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch (UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); --- End diff -- https://www.slf4j.org/api/org/slf4j/Logger.html#warn(org.slf4j.Marker, java.lang.String, java.lang.Throwable) Can we do like, LOG.warn("Cannot re-resolve server: {}, exception: ", curAddr, e);

33. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96509432 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { --- End diff -- instead of `null` checking `ia` couldn't we use `addr.isUnresolved()`?
34. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96512625 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { +

serverAddresses.remove(currentIndex); --- End diff -- In the case where during the construction of the `StaticHostProvider` we add a host H1 which resolves to addresses A,B,C. Then at some point in the future A goes offline and the DNS system has changed H1 to resolve to hosts D,E,F. With the current code wouldn't `serverAddresses` now contain hosts B,C,D,E,F instead of only hosts D,E,F? Is this something even worth fixing?

35. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96509843 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,12 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the --- End diff -- nit: Can we swap the first two lines of this comment, as the reader does not know what class getHostString belongs to until he/she reaches line 2?
36. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96574584 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { --- End diff -- OK, it is a reasonable concern of using package private API via reflection and I honestly don't know the implementation difference between Java 6 / 7 regarding getHostString - let's stick to explicit implementation in this file.
37. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96574629 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } + } + } catch (UnknownHostException e) { + LOG.warn("Cannot re-resolve server: " + curAddr + " UnknownHostException: " + e); --- End diff -- OK.
38. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96642443 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); -

InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { --- End diff -- @hanm have a look at this, please.

39. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96694626 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { --- End diff -- Yup, we could: http://grepcode.com/file/repository.grepcode.com/java/root/jdk/openjdk/6-b27/java/net/InetSocketAddress.java#258
40. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96696184 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +86,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } --- End diff -- As we are adding new addresses into `serverAddresses`, would it make sense to re-shuffle the list? ``` Collections.shuffle(this.serverAddresses); ``` As we did in constructor???
41. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96699234 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { --- End diff -- It might be better to wrap the reflection in an abstraction in a static block in this file so inspection of the class will be done only once, save some runtime inspection cycles.
42. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96699514 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { + Method m =

InetSocketAddress.class.getDeclaredMethod("getHostString"); + m.setAccessible(true); + resolvedAddresses = InetAddress.getAllByName((String) m.invoke(address)); + } catch (IllegalAccessException e) { --- End diff -- Maybe we can catch multiple exceptions in a single shot - such as `catch(IllegalAccessException | NoSuchMethodException | InvocationTargetException e)` to save some typings, given the exception handling logic is exact the same.

43. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96699622 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { + Method m = InetSocketAddress.class.getDeclaredMethod("getHostString"); + m.setAccessible(true); + resolvedAddresses = InetAddress.getAllByName((String) m.invoke(address)); + } catch (IllegalAccessException e) { + resolvedAddresses = InetAddress.getAllByName(getHostString(address)); + } catch (NoSuchMethodException e) { + resolvedAddresses = InetAddress.getAllByName(getHostString(address)); + } catch (InvocationTargetException e) { + resolvedAddresses = InetAddress.getAllByName(getHostString(address)); + } --- End diff -- The signature of getDeclaredMethod said it also could throw SecurityException, not sure if we should catch it or not here.
44. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96701932 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { + Method m = InetSocketAddress.class.getDeclaredMethod("getHostString"); + m.setAccessible(true); + resolvedAddresses = InetAddress.getAllByName((String) m.invoke(address)); + } catch (IllegalAccessException e) { --- End diff -- Nope 'cause JDK6 here, no?
45. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r96702291 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { + Method m = InetSocketAddress.class.getDeclaredMethod("getHostString"); + m.setAccessible(true); + resolvedAddresses = InetAddress.getAllByName((String) m.invoke(address)); + } catch (IllegalAccessException e) { --- End diff -- Right, I forgot the context, thanks for pointing this out @eribeiro
46. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r97435427 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { --- End diff -- I'm still not sure we should do this. I'm concerned about making that method visible while the original intention was not to expose it. Are you aware of any other project that has done this for `getHostString`?
47. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r97436195 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { + Method m = InetSocketAddress.class.getDeclaredMethod("getHostString"); + m.setAccessible(true); + resolvedAddresses = InetAddress.getAllByName((String) m.invoke(address)); + } catch

(IllegalAccessException e) { + resolvedAddresses = InetAddress.getAllByName(getHostString(address)); + } catch (NoSuchMethodException e) { + resolvedAddresses = InetAddress.getAllByName(getHostString(address)); + } catch (InvocationTargetException e) { + resolvedAddresses = InetAddress.getAllByName(getHostString(address)); + } --- End diff -- That's possibly another issue with this way of exposing `getHostString, the presence of a security manager could prevent us from doing it as expected.

48. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r97436370 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { --- End diff -- OK.
49. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r97437693 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +86,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); + for (InetAddress resolvedAddress : resolvedAddresses) { + InetSocketAddress newAddr = new InetSocketAddress(resolvedAddress, thePort); + if (!serverAddresses.contains(newAddr)) { + serverAddresses.add(newAddr); + } + } --- End diff -- We shuffle initially to avoid having all clients connecting to the same server in the case they are all given the same connect string. If the array of addresses has already been shuffled (in the constructor), then the order followed in this method will be the shuffled one. I don't see a strong reason for re-shuffling, as we are not bringing it back to the original order by resolving again.
50. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r97446760 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +73,69 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * It evaluates to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString; + InetAddress ia = addr.getAddress(); + + if (ia != null) { + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); +

hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!curAddr.getHostString().equals(curAddr.getAddress().getHostAddress())) { + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + } else { + serverAddresses.remove(currentIndex); --- End diff -- @afine check the new changes to see if they address this and make sense.

51. Github user hanm commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r97921838 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,26 +62,20 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) throws UnknownHostException { for (InetSocketAddress address : serverAddresses) { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia!=null) ? ia.getHostAddress(): - address.getHostName()); + InetAddress resolvedAddresses[]; + try { --- End diff -- @fpj Let's not do this (expose package private JDK methods) given the concerns you raised earlier (as I also commented previously).
52. Github user edwardoliveira commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98263409 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while(i < serverAddresses.size()) { + if(getHostString(serverAddresses.get(i)) == getHostString(curAddr)) { --- End diff -- Ops, are we comparing strings with `==` ?! ;) Nit: space between `if` and `(`.
53. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98263678 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address

parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while(i < serverAddresses.size()) { + if(getHostString(serverAddresses.get(i)) == getHostString(curAddr)) { --- End diff -- Ops, are we comparing strings with `==` ?! ;) Nit: space between `if` and (`.

54. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98263780 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { --- End diff -- nit: space between `if` and `(`.
55. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98265031 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { --- End diff -- okay, just me being pedant here, but it's a sincere question: would it be *worth* to move this class to `src/java/test/org/apache/zookeeper/client/StaticHostProvider.java` and then remove the `public` modifier so that `getNextAdded()` and `getNextRemoved()` can be package protected as they are used for tests? At first, I think it doesn't see worth this kind of change, but I am uncomfortable with leaving those methods as public if they are used for tests by now. :thinking: Wdyt?
56. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98280911 --- Diff:

src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while(i < serverAddresses.size()) { + if(getHostString(serverAddresses.get(i)) == getHostString(curAddr)) { --- End diff -- Why call `getHostString(curAddr)` (here in a loop, at line 137, 138, 141) if the `currAddr` doesn't change? Better call once between line 136 and 137 and assign to a variable, no?

57. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98281621 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; --- End diff --

silly refactoring: ``` List<String> toRemove = new ArrayList<>(serverAddresses.size()); for (String addr : serverAddresses) { if (getHostString(addr).equals(hostString) { toRemove.add(addr); } } LOG.debug("Removing addresses: {}", toRemove); nextRemoved += toRemove.size(); serverAddresses.removeAll(toRemove); ```

58. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98282550 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; --- End diff -- ``` Line 136: String hostString = getHostString(currAddr); ```

59. Github user fpj commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98385538 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress

resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; --- End diff -- Although this is more Java-like, it requires the creation of an additional ArrayList, which is less efficient than creating a int counter. Unless there is something wrong with the current code, I'd rather leave as is.

60. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98392957 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,104 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; --- End diff -- You right.
61. Github user fpj commented on the issue: https://github.com/apache/zookeeper/pull/150 @hanm I believe we do have the same issue with the C client, I don't see it re-resolving addresses there, I need to have a closer look, though.
62. Github user edwardoliveira commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98483994 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -122,18 +122,19 @@ public int size() { private int nextAdded = 0; private int nextRemoved = 0; - public int getNextAdded() { + int getNextAdded() { return nextAdded; } - public int getNextRemoved() { + int getNextRemoved() { return nextRemoved; } public InetSocketAddress next(long spinDelay) { // Handle possible connection error by re-resolving hostname if possible if (!connectedSinceNext) { InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { --- End diff -- @fpj Sorry for yet another comment, mate (my last one, I promise). I cited this previously but certainly got lost in my verbosite. * Replace `if (!getHostString(currAddr).equals(...)) {` by 'if (!curHostString.equals()` at Line 138; * Replace `getHostString(curAddr)` with `curHostString` at line 139; * Replace `getHostString(curAddr)` with `curHostString` at line 142; **+1. LGTM. Really great job!.** Best regards,
63. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98484702 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -122,18 +122,19 @@ public int size() { private int nextAdded = 0; private int nextRemoved = 0; - public int getNextAdded() { + int getNextAdded() { return nextAdded; } - public int getNextRemoved() { + int getNextRemoved() { return nextRemoved; } public InetSocketAddress next(long spinDelay) { // Handle possible connection error by

re-resolving hostname if possible if (!connectedSinceNext) { InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); --- End diff -- edwardoliveira just now @fpj Sorry for yet another comment, mate (my last one, I promise). I cited this previously but certainly got lost in my verbosite. Replace if (!getHostString(currAddr).equals(...)) { by 'if (!curHostString.equals()` at Line 138; Replace getHostString(curAddr) with curHostString at line 139; Replace getHostString(curAddr) with curHostString at line 142; **IMHO, +1. Really great job!** :+1: Best regards,

64. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98543324 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,106 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + int getNextAdded() { + return nextAdded; + } + + int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(curAddr)); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- I think that this fixes the issue I described. nit: Just wondering if it would be easier to use a map from (hoststring,port) -> serverAddress to make things clearer/remove a loop?
65. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98535935 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,106 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + int getNextAdded() { + return nextAdded; + } + + int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { -

++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { --- End diff -- nit: why not use curHostString instead of calling getHostString two more times?

66. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98549400 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,106 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + int getNextAdded() { + return nextAdded; + } + + int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { --- End diff -- @afine https://github.com/apache/zookeeper/pull/150#discussion_r98484702 :sunglasses:

67. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98564047 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,106 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + int getNextAdded() { + return nextAdded; + } + + int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { --- End diff -- whoops, apologies for the duplicate. ðŸ˜§

68. Github user eribeiro commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/150#discussion_r98565216 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -87,15 +75,106 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses)

Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if(addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + int getNextAdded() { + return nextAdded; + } + + int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!getHostString(curAddr).equals(curAddr.getAddress().getHostAddress())) { --- End diff -- No problem at all. ðŸ˜ƒ

69. Github user ijuma commented on the issue: https://github.com/apache/zookeeper/pull/150 Thanks for pushing this useful improvement over the line @fpj.
70. Github user rakeshadr commented on the issue: https://github.com/apache/zookeeper/pull/150 Thanks everyone for the great effort & time in pushing this issue. I could see long discussions in the PR and I hope this work is nearing completion. Could you please update the progress and would like to know the chances of pushing this asap, thanks!.
71. I haven't had much time to work on this issue, but here is my current assessment. This issue seemed easy to fix at first, but it is fairly fundamental with respect to how we resolve host names. Currently, we resolve host names when we start a client and never resolve it again. This is the cause of the problem reported in the issue because in the scenario described, the zookeeper container is re-started and changes addresses, which prevents the client from connecting to the zookeeper server. The proposed patch here tries to re-resolve the hostname every time the client fails to connect to the resolved address. It kind of works, but it makes {{StaticHostProvider}} a bit messy because the expectation with the current wiring is that we won't have to resolve again. The ideal situation for the problematic scenario is that we resolve the host name every time we try to connect to a server, but that would be a fairly fundamental change to how we resolve addresses in ZooKeeper. I was also looking at the C client and it might get a bit messy too there because I don't think we currently keep the association between the host name and the resolved address, so we don't really know what to resolve again. It might be possible to do it via the canonical name in {{getaddrinfo}}, but I'm not sure how that works with windows. One specific proposal to avoid having clients never finding a server ever again without deep changes to the current wiring is to resolve again everything in the case the client tries all and none succeeds. That would be a fairly straightforward change to both Java and C client, but it would not resolve addresses again in the case the a strict subset has changed addresses and at least one server is reachable.
72. another option would be to have a background worker that periodically wakes up and re-resolves hosts every few minutes. if we ever get a connection failure we could use that to kick the background worker to run right away.
73. Hi All, I have worked on some other issue where I had to change StaticHostProvider to contain ServerCfg class which include the host string provided at config time, the resolved Inetaddress and SSL cert fingerprint. And also fixing all the plumbing everywhere to carry or operate upon ServerCfg. Will this be sufficient to address this issue?. https://github.com/apache/zookeeper/pull/185/files#diff-1b64f5144158570491cfdec2b93b5c79 I have modified this PR to fit needs of having a chance at getting SSL support committed hence I removed these changes and published a different PR: https://github.com/apache/zookeeper/pull/188 (has the StaticHostProvider host changes removed and all the plumbing restored) for your reference to changes with and without StaticHostProvider modified. Let me know if this a direction worth while pursuing I can carve out just the StaticHostProvider changes (without the SSL cert fingerprint in ServerCfg) and publish them as a PR. thanks Powell.
74. GitHub user geek101 opened a pull request: https://github.com/apache/zookeeper/pull/199 ZOOKEEPER-2184: Resolve address only on demand. Wrap hostname and port into a new ServerCfg

class and fix all the places to use it instead of InetSocketAddress. This class can be used in the future to encapsulate other nice config information for example certificate fingerprint associated with the host etc. You can merge this pull request into a Git repository by running: $ git pull https://github.com/geek101/zookeeper branch-3.5-hostname-resolve-always Alternatively you can review and apply these changes as the patch at: https://github.com/apache/zookeeper/pull/199.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #199 ---- commit ec219a4f40fe46a9743ad13ff910f9be9f383f3c Author: Powell Molleti <powellm79@yahoo.com> Date: 2017-03-17T07:59:23Z Resolve address only on demand. Wrap hostname and port into a new ServerCfg class and fix all the places to use it instead of InetSocketAddress. This class can be used in the future to encapsulate other nice config information for example certificate fingerprint associated with the host etc. ----

75. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 75 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. -1 release audit. The applied patch generated 2 release audit warnings (more than the trunk's current 0 warnings). -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/440//testReport/ Release audit warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/440//artifact/trunk/patchprocess/patchReleaseAuditProblems.txt Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/440//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/440//console This message is automatically generated.
76. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 75 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/441//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/441//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/441//console This message is automatically generated.
77. not sure what the failure above is but the cpp unit tests work for me locally for that PR I submitted. {noformat} [exec] [exec] *** Error in `./zktest-mt': corrupted double-linked list: 0x00000000012a5810 *** [exec] [exec] Zookeeper_readOnly::testReadOnly : elapsed 4101 : OK [exec] [exec] OK (74) [exec] [exec] FAIL: zktest-mt [exec] [exec] == [exec] [exec] 1 of 2 tests failed [exec] [exec] Please report to user@zookeeper.apache.org [exec] [exec] == [exec] [exec] make[1]: Leaving directory `/home/jenkins/jenkins-slave/workspace/PreCommit-ZOOKEEPER-github-pr-build/build/test/test-cppunit' [exec] [exec] /bin/bash: line 5: 8114 Aborted (core dumped) ZKROOT=/home/jenkins/jenkins-slave/workspace/PreCommit-ZOOKEEPER-github-pr-build/src/c/../.. CLASSPATH=$CLASSPATH:$CLOVER_HOME/lib/clover.jar ${dir}$tst [exec] [exec] make[1]: *** [check-TESTS] Error 1 [exec] [exec] make: *** [check-am] Error 2 {noformat}
78. Github user adyach commented on the issue: https://github.com/apache/zookeeper/pull/199 @geek101 will it help to update the list of the ip address in case I use load balancer in connection string? I am asking since I have not found when zookeeper updates list of got ip address. In case we have a situation, when all ip address are not valid anymore.
79. Github user adyach commented on the issue: https://github.com/apache/zookeeper/pull/199 Seems like this is https://github.com/apache/zookeeper/pull/150
80. Github user geek101 commented on the issue: https://github.com/apache/zookeeper/pull/199 @adyach can you give me an example of a connection string that you are implying this will help me understand the problem better. Since the client code needs ip addresses of the ZK ensemble, does the DNS resolution of this load balancer hostname supposed to return that set of ip addresses?
81. Github user adyach commented on the issue: https://github.com/apache/zookeeper/pull/199 `your.zookeeper.loadbalancer` The problem is that it is resolved only once when connected, but instances behind load balancer can be replaced.

82. Github user geek101 commented on the issue: https://github.com/apache/zookeeper/pull/199 @adyach this patch's goal is to make sure when ever a new connection is being established DNS lookup will be performed. If you have given a hostname in connection string then if the underlying IP is gone the TCP connection will break and new TCP connection will be attempted by Zookeeper code and at this time DNS lookup will again be performed. I hope this helps if you can post your example of connection string it will be more helpful. Zookeeper connection string is supposed to contain the ZK ensemble set/subset hostnames/ips.
83. Github user rcillo commented on the issue: https://github.com/apache/zookeeper/pull/150 This feature is highly valuable for the community. It could solve the problem of every team deploying Kafka on the cloud. Kafka has a static configuration with the IP addresses of Zookeeper nodes. If you need to replace these nodes and consequently change their IP addresses, you need to change Kafka configuratino file and then restart all Kafka nodes so that they will reload the updated configuration. If this feature is merged, everyone deploying Kafka on the cloud could configure it using a load balancer address, that would be re-resolved from time to time, so that new Zookeeper instances would be automatically reachable from Kafka without the need of restarts. This would greatly improve the availability of Kafka. Looking forward to have this merged.
84. Github user djenriquez commented on the issue: https://github.com/apache/zookeeper/pull/150 Any reasons why this hasn't been merged yet or the attention given? I agree whole-heartedly with @rcillo, this is a gigantic feature for anyone depending on Zookeeper in the cloud running on immutable/disposal infrastructure. Would love to get this merged, but it being 6 months old makes me wonder if it has been de-prioritized by project owners/collaborators? If so, can someone explain why has this been de-prioritized?
85. Github user hanm commented on the issue: https://github.com/apache/zookeeper/pull/150 This PR has to be rebased first before it can be merged. Hi Flavio - will you follow up with this or you prefer someone else take this over? @fpj @rakeshadr I've updated the JIRA to mark it as a blocker for next release (3.4.11, 3.5.4), to prevent this issue lagging again. Good to get this in given its impact and relatively little effort given the PR is already in a good shape.
86. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 75 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/848//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/848//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/848//console This message is automatically generated.
87. Github user djenriquez commented on the issue: https://github.com/apache/zookeeper/pull/150 Hi guys, any update for this PR? Many thanks!!
88. Github user hanm commented on the issue: https://github.com/apache/zookeeper/pull/150 We need get this moving, but let's first wait for feedback from @fpj before letting someone else taking over this JIRA.
89. Github user nicorevin commented on the issue: https://github.com/apache/zookeeper/pull/150 @fpj any updates on it? It seems like a blocker for clustering kafka (and everyone using zkclient) in docker/kubernetes.
90. Github user edvorkin commented on the issue: https://github.com/apache/zookeeper/pull/150 +1 one here. This feature is necessary for running Zookeeper in the cloud under AWS ASG. Every time node fails, ASG reassigns new IP for new zookeeper and there is no way kafka will know about it. We need to treat zookeeper servers as cattle, not pets, and kill and spin new one at will without affecting kafka.
91. This is the remaining blocker for 3.4.11 - any insights [~fpj] ? (see recent comments) Would be nice to get this one in. I don't believe this is a regression - if we don't see movement soon I will likely downgrade the priority and move out to the next release.
92. **body:** It doesn't look like anyone feels we need to block 3.4 for this - pushing to 3.4.12.
    label: code-design
93. Github user the-xs commented on the issue: https://github.com/apache/zookeeper/pull/150 Should we still wait or get this moving to fix the merge conflicts?
94. Hi, does anyone know when this issue can be fixed? This is related to https://issues.apache.org/jira/browse/KAFKA-5473. We are wondering if we should have a short term fix in Kafka or just wait for the fix in ZK.

95. Github user jorgheymans commented on the issue: https://github.com/apache/zookeeper/pull/150 just got stung by this as well, assumed zk clients would be clever enough to reresolve :-/ Since there is a lot of interest in this why not just rebase-merge and let ppl test out the snapshot builds ?
96. Github user riccardofreixo commented on the issue: https://github.com/apache/zookeeper/pull/150 We're running Kafka in Kubernetes, so this bug was biting us regularly. We applied the patch in the kafka clusters of our client and are running in prod. Solves our problem and created no additional problems for us.
97. Github user sslavic commented on the issue: https://github.com/apache/zookeeper/pull/150 @riccardofreixo have you tried using ClusterIP Service for ZooKeeper StatefulSet and providing that ClusterIP (or service hostname) to Kafka / ZooKeeper clients as sole ZooKeeper hostname? StatefulSet can have multiple replicas, but to ZooKeeper clients all of the members no matter how many of them there are (1, 3, 5, ..) would be accessible under single ClusterIP. Even when Pods of StatefulSet die and get re-scheduled for whatever reason, they will likely get new IP, but IP of ClusterIP Service remains stable so ZooKeeper clients should be able to reconnect, without need to reresolve IP address of the host. If there's a quorum, Pod that died does not necessarily have to become available quickly, clients should still be able to connect even without losing session.
98. Github user riccardofreixo commented on the issue: https://github.com/apache/zookeeper/pull/150 @sslavic thanks for the suggestion. We haven't tried that approach, and as far as I can tell it sounds like it would work. You'd still have the re-resolution problem if you deleted/recreated the service, but that should be quite rare. Had we thought of that before, we probably wouldn't have patched the client. Now we have though, we'll keep it patched. I still think this should be fixed on the zk-client, as there are other circumstances other than Kube where the IP addresses may change and you wouldn't have an easy solution such as ClusterIP.
99. Github user phunt commented on the issue: https://github.com/apache/zookeeper/pull/150 Given the insights from the Kafka and K8s folks this looks like a good one to focus on. @fpj any chance you can update this PR to address the conflicts?
100. Github user jeffwidman commented on the issue: https://github.com/apache/zookeeper/pull/150 Any movement on this?
101. Github user bwmills commented on the issue: https://github.com/apache/zookeeper/pull/150 As noted by @rcillo back in June - this feature is highly valuable for the community. It's certainly of critical importance to our production services in K8s. Any updates are much appreciated.
102. Github user phunt commented on the issue: https://github.com/apache/zookeeper/pull/150 I suspect folks were out on vacation. I was. :-) It doesn't seem like @fpj has time to look at this - can someone else pick it up and address the recent comments?
103. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/150 @phunt @afine @fpj I'm happy to pick this up tomorrow.
104. GitHub user anmolnar opened a pull request: https://github.com/apache/zookeeper/pull/451 ZOOKEEPER-2184: Zookeeper Client should re-resolve hosts when connection attempts fail This one is the pick-up of @fpj 's original PR: #150 Targeting and rebased on the 3.4 branch. You can merge this pull request into a Git repository by running: $ git pull https://github.com/anmolnar/zookeeper ZOOKEEPER-2184 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/zookeeper/pull/451.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #451 ---- commit 229760d7757f47e271a8e059c1aeac10f0847a2a Author: fpj <fpj@...> Date: 2017-01-14T16:58:15Z ZOOKEEPER-2184: Zookeeper Client should re-resolve hosts when connection attempts fail ----
105. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r163570637 --- Diff: src/java/test/org/apache/zookeeper/client/StaticHostProviderTest.java --- @@ -117,8 +117,32 @@ public void testTwoInvalidHostAddresses() { list.add(new InetSocketAddress("a", 2181)); list.add(new InetSocketAddress("b", 2181)); new StaticHostProvider(list); + } + + @Test + public void testReResolving() { + byte size = 1; + ArrayList<InetSocketAddress> list = new ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to multiple addresses --- End diff -- Common domain names like facebook.com, google.com or apache.org don't use round-robin DNS anymore for some reason, so they don't resolve to multiple addresses. As a consequence this test doesn't validate the branch related to multiple addresses anymore unfortunately. Not sure how to address that, PowerMock would be the best to mock static `getAllByName()` method, but that would involve introducing a new test dependency.
106. Github user mfenes commented on the issue: https://github.com/apache/zookeeper/pull/451 Re-resolving at StaticHostProvider level may not be sufficient as InetAddress.getAllByName(String host) itself uses a

Java-level cache inside InetAddress and turns to name service (e.g. DNS) only if the host could not be found in the Java-level cache. Unfortunately, when Java resolves a new host using the name service, it puts the host and its addresses in the cache with TTL cache FOREVER. This means, once a host gets resolved by Java, it will never again turn to the name service to re-resolve it. If a host's addresses get updated in DNS, the address cache in Java will still contain the old entry forever. So re-resolving at StaticHostProvider won't help in this case, as InetAddress.getAllByName(String host) will still return the old address(es) I think. Check the getCachedAddresses method inside InetAddress, the get() method of static final class Cache inside InetAddress and sun.net.InetAddressCachePolicy.get() which returns cachePolicy with default value -1 (FOREVER) if it is not overridden by Security properties "networkaddress.cache.ttl" and "networkaddress.cache.negative.ttl".

107. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 @mfenes The only solution I can think of is to set DNS cache TTL `networkaddress.cache.ttl` to a configurable, non-infinite value.
108. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 Just confirmed on 3.4 branch: ZK uses 30 secs cache TTL on my mac.
109. Github user mfenes commented on the issue: https://github.com/apache/zookeeper/pull/451 Looking at the static initialization block in InetAddressCachePolicy more deeply, the default TTL is 30 seconds if there is no SecurityManager installed. So caching a positive lookup forever in the Java-level cache is the default only if there is a SecurityManager installed and the TTL is not overridden by "networkaddress.cache.ttl" to a different value. Default caching policy for a negative lookup is 0 (never cache). Now the only question is whether 30 seconds default caching is ok or too much for ZK.
110. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 @phunt @afine Did you have a chance to take a look? I think we've addressed all issues that were mentioned in the original PR.
111. Github user jeffwidman commented on the issue: https://github.com/apache/zookeeper/pull/451 Should this PR be targeting `branch-3.4` or target `trunk` and then backport to the 3.4 series?
112. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165525113 --- Diff: src/java/test/org/apache/zookeeper/client/StaticHostProviderTest.java --- @@ -16,7 +16,7 @@ * limitations under the License. */ -package org.apache.zookeeper.test; +package org.apache.zookeeper.client; --- End diff -- this doesn't look right
113. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165525652 --- Diff: src/java/test/org/apache/zookeeper/test/ReadOnlyModeTest.java --- @@ -239,13 +243,13 @@ public void testSessionEstablishment() throws Exception { public void testSeekForRwServer() throws Exception { // setup the logger to capture all logs - Layout layout = Logger.getRootLogger().getAppender("CONSOLE") + Layout layout = org.apache.log4j.Logger.getRootLogger().getAppender("CONSOLE") --- End diff -- why is this necessary?
114. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165527366 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,29 +62,12 @@ */ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { - try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); - for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } - } - } catch (UnknownHostException e) { + try { --- End diff -- something is wrong with the indentation here
115. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165529085 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -91,15 +79,106 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) {

Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + int getNextAdded() { + return nextAdded; + } + + int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { --- End diff -- would you mind explaining exactly under which conditions we reresolve the hostname and under which conditions we try the next one in the host list? My reading is that this reresolves everything if the client fails to connect to two hosts in a row. Is this the desired behavior? And do we always reresolve all serverAddresses?

116. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165524377 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -25,6 +25,8 @@ import java.util.Collection; import java.util.Collections; import java.util.List; +import java.lang.reflect.InvocationTargetException; --- End diff -- i think these imports are unused, and there are some others elsewhere in the code
117. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 @jeffwidman > Should this PR be targeting branch-3.4 or target trunk and then backport to the 3.4 series? The original PR targets 3.4 which is explained in a comment from @fpj on the jira: https://issues.apache.org/jira/browse/ZOOKEEPER-2184?focusedCommentId=15823099&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-15823099 > For the 3.5 branch, we will need a different patch because of the reconfiguration changes to StaticHostProvider. I'll work on it once the 3.4 patch gets a +1.
118. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165658743 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -25,6 +25,8 @@ import java.util.Collection; import java.util.Collections; import java.util.List; +import java.lang.reflect.InvocationTargetException; --- End diff -- Good catch, thanks.
119. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165658965 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -57,29 +62,12 @@ */ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { - try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); - for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } - } - } catch (UnknownHostException e) { + try { --- End diff -- Fixing.
120. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165665505 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -91,15 +79,106 @@ public

StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + int getNextAdded() { + return nextAdded; + } + + int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { --- End diff -- It should try to re-resolve whenever the client is unable to connect to a server (connectedSinceNext == false). @fpj gives a good explanation in the original Jira: https://issues.apache.org/jira/browse/ZOOKEEPER-2184?focusedCommentId=15873730&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-15873730 > I haven't had much time to work on this issue, but here is my current assessment. > This issue seemed easy to fix at first, but it is fairly fundamental with respect to how we resolve host names. Currently, we resolve host names when we start a client and never resolve it again. This is the cause of the problem reported in the issue because in the scenario described, the zookeeper container is re-started and changes addresses, which prevents the client from connecting to the zookeeper server. > The proposed patch here tries to re-resolve the hostname every time the client fails to connect to the resolved address. It kind of works, but it makes StaticHostProvider a bit messy because the expectation with the current wiring is that we won't have to resolve again. > The ideal situation for the problematic scenario is that we resolve the host name every time we try to connect to a server, but that would be a fairly fundamental change to how we resolve addresses in ZooKeeper. > I was also looking at the C client and it might get a bit messy too there because I don't think we currently keep the association between the host name and the resolved address, so we don't really know what to resolve again. It might be possible to do it via the canonical name in getaddrinfo, but I'm not sure how that works with windows. > One specific proposal to avoid having clients never finding a server ever again without deep changes to the current wiring is to resolve again everything in the case the client tries all and none succeeds. That would be a fairly straightforward change to both Java and C client, but it would not resolve addresses again in the case the a strict subset has changed addresses and at least one server is reachable.

121. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r165666521 --- Diff: src/java/test/org/apache/zookeeper/client/StaticHostProviderTest.java --- @@ -16,7 +16,7 @@ * limitations under the License. */ -package org.apache.zookeeper.test; +package org.apache.zookeeper.client; --- End diff -- It was move to client package, because it uses package-private methods of StaticHostProvider. I moved back to 'test' package and change the affected methods to public.
122. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166102194 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(-

address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { --- End diff -- what happens when a host that resolves to multiple addresses changes to resolving to just one?

123. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166103404 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return

serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- as i mentioned in https://github.com/apache/zookeeper/pull/150/files#r98543324 this all gets a little complicated? What do you think about using a map to trap all these associations?

124. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166097161 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -6,9 +6,9 @@ * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at - * - * http://www.apache.org/licenses/LICENSE-2.0 - * + * <p> --- End diff -- was this accidental?
125. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166105622 --- Diff: src/java/test/org/apache/zookeeper/test/StaticHostProviderTest.java --- @@ -117,8 +116,32 @@ public void testTwoInvalidHostAddresses() { list.add(new InetSocketAddress("a", 2181)); list.add(new InetSocketAddress("b", 2181)); new StaticHostProvider(list); + } + + @Test + public void testReResolving() { + byte size = 1; + ArrayList<InetSocketAddress> list = new ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to multiple addresses + list.add(InetSocketAddress.createUnresolved("www.apache.org", 1234)); --- End diff -- I'm wondering if it's possible to mock this out? It would be great if our unit tests were not dependent on some other infrastructure.
126. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166286530 --- Diff: src/java/test/org/apache/zookeeper/test/ReadOnlyModeTest.java --- @@ -239,13 +243,13 @@ public void testSessionEstablishment() throws Exception { public void testSeekForRwServer() throws Exception { // setup the logger to capture all logs - Layout layout = Logger.getRootLogger().getAppender("CONSOLE") + Layout layout = org.apache.log4j.Logger.getRootLogger().getAppender("CONSOLE") --- End diff -- I few lines above there're org.apache.log4j references which conflicts with the 'Logger' class, hence it needs to be explicitly referenced.
127. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166286605 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -6,9 +6,9 @@ * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at - * - * http://www.apache.org/licenses/LICENSE-2.0 - * + * <p> --- End diff -- Yes it was.
128. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166328583 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() !=

null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { --- End diff -- That's actually a very good point. I'm looking into that you suggested below (using a Map instead of Array) which will probably solve this issue too.

129. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166329760 --- Diff: src/java/test/org/apache/zookeeper/test/StaticHostProviderTest.java --- @@ -117,8 +116,32 @@ public void testTwoInvalidHostAddresses() { list.add(new InetSocketAddress("a", 2181)); list.add(new InetSocketAddress("b", 2181)); new StaticHostProvider(list); + } + + @Test + public void testReResolving() { + byte size = 1; + ArrayList<InetSocketAddress> list = new ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to multiple addresses + list.add(InetSocketAddress.createUnresolved("www.apache.org", 1234)); --- End diff -- Good point I spent hours finding a way for it. Unfortunately static members can't be mocked with Mockito. We should use PowerMock for it (as described here https://blog.codecentric.de/en/2016/03/junit-testing-using-mockito-powermock/), but that'd be a new dependency not sure if it's acceptable here.
130. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166341540 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be

empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- It seems that this change makes things a little a bit even more complicated, because `next()` method needs to have a list of IP addresses to iterate on. Maintenance is easier with a Map, but I need to find a way to use it as a list.

131. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166476047 --- Diff: src/java/test/org/apache/zookeeper/test/ReadOnlyModeTest.java --- @@ -239,13 +243,13 @@ public void testSessionEstablishment() throws Exception { public void testSeekForRwServer() throws Exception { // setup the logger to capture all logs - Layout layout = Logger.getRootLogger().getAppender("CONSOLE") + Layout layout = org.apache.log4j.Logger.getRootLogger().getAppender("CONSOLE") --- End diff -- not going to push too hard on this, but I think you can just use log4j everywhere like in `QuorumPeerMainTest`
132. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166476703 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string

of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- Not sure I understand this point. I believe there are implementations of HashMap that provide predictable ordering of the keys, so you should still be able to use next. Let me know if I am misunderstanding you here.

133. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166479687 --- Diff: src/java/test/org/apache/zookeeper/test/StaticHostProviderTest.java --- @@ -117,8 +116,32 @@ public void testTwoInvalidHostAddresses() { list.add(new InetSocketAddress("a", 2181)); list.add(new InetSocketAddress("b", 2181)); new StaticHostProvider(list); + } + + @Test + public void testReResolving() { + byte size = 1; + ArrayList<InetSocketAddress> list = new ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to multiple addresses + list.add(InetSocketAddress.createUnresolved("www.apache.org", 1234)); --- End diff -- Yeah, this is annoying. Although, another possibility would be to put the dns calls in a method and then subclass `StaticHostProvider` for the tests and overriding this method to return what you want.
134. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166669488 --- Diff: src/java/test/org/apache/zookeeper/test/ReadOnlyModeTest.java --- @@ -239,13 +243,13 @@ public void testSessionEstablishment() throws Exception { public void testSeekForRwServer() throws Exception { // setup the logger to capture all logs - Layout layout = Logger.getRootLogger().getAppender("CONSOLE") + Layout layout = org.apache.log4j.Logger.getRootLogger().getAppender("CONSOLE") --- End diff -- slf4j's string formatter feature is being used in these tests. I wouldn't refactor them to be honest.
135. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166670125 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch

(UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- How to shuffle the elements?

136. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r166671671 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return

serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { --- End diff -- It might be easier to just remove the special case of `resolvedAddresses.length == 1` and let the other part iterate over the list and remove the affected elements.

137. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r167013851 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- You could put the `keySet` in a list and shuffle?

138. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r167024767 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- The key is the hoststring which is like `issues.apache.org` and could have multiple InetAddresses which are the values of the key in the map. If I shuffle only the keys, I can iterate on only the keys and not sure how to choose from the addresses.
139. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 Going one step back I wonder why we try to deal with multiple addresses at all. HostProvider should just make a transformation from unresolved InetSocketAddresses to resolved InetSocketAddresses. The easiest way as I can see it is to create a new instance of InetSocketAddress if the input is unresolved every time `next()` is called. Otherwise just pass it through. JVM will deal with the rest: resolution, caching and re-resolution once the cache is expires (30 secs).
140. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r167027624 --- Diff: src/java/test/org/apache/zookeeper/test/StaticHostProviderTest.java --- @@ -117,8 +116,32 @@ public void testTwoInvalidHostAddresses() { list.add(new InetSocketAddress("a", 2181)); list.add(new InetSocketAddress("b", 2181)); new StaticHostProvider(list); + } + + @Test + public void testReResolving() { + byte size = 1; + ArrayList<InetSocketAddress> list = new

ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to multiple addresses + list.add(InetSocketAddress.createUnresolved("www.apache.org", 1234)); --- End diff -- Makes sense, I'm working on it.

141. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168524336 --- Diff: src/java/test/org/apache/zookeeper/test/StaticHostProviderTest.java --- @@ -117,8 +116,32 @@ public void testTwoInvalidHostAddresses() { list.add(new InetSocketAddress("a", 2181)); list.add(new InetSocketAddress("b", 2181)); new StaticHostProvider(list); + } + + @Test + public void testReResolving() { + byte size = 1; + ArrayList<InetSocketAddress> list = new ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to multiple addresses + list.add(InetSocketAddress.createUnresolved("www.apache.org", 1234)); --- End diff -- This is done.
142. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168564981 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -18,6 +18,10 @@ package org.apache.zookeeper.client; +import org.apache.yetus.audience.InterfaceAudience; --- End diff -- nit: move this back
143. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168565039 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -18,6 +18,10 @@ package org.apache.zookeeper.client; +import org.apache.yetus.audience.InterfaceAudience; --- End diff -- nit: move this back
144. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168578617 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again

hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- Why don't we simplify things and just do this always? In other words, always remove all serverAddresses that correspond to the curHostString. That way if curHostString goes from resolving to many addresses to just 1, we remove all of them?

145. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168568596 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -47,59 +51,169 @@ private int currentIndex = -1; + // Don't re-resolve on first next() call + private boolean connectedSinceNext = true; + + private Resolver resolver; + /** * Constructs a SimpleHostSet. - * + * * @param serverAddresses * possibly unresolved ZooKeeper server addresses * @throws IllegalArgumentException * if serverAddresses is empty or resolves to an empty list */ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { + this.resolver = new Resolver() { + @Override + public InetAddress[] getAllByName(String name) throws UnknownHostException { + return InetAddress.getAllByName(name); + } + }; + init(serverAddresses); + } + + /** + * Introduced for testing purposes. getAllByName() is a static method of InetAddress, therefore cannot be easily mocked. + * By abstraction of Resolver interface we can easily inject a mocked implementation in tests. + * + * @param serverAddresses + * possibly unresolved ZooKeeper server addresses + * @param resolver + * custom resolver implementation + * @throws IllegalArgumentException + * if serverAddresses is empty or resolves to an empty list + */ + public StaticHostProvider(Collection<InetSocketAddress> serverAddresses, Resolver resolver) { + this.resolver = resolver; + init(serverAddresses); + } + + /** + * Common init method for all constructors. + * Resolve all unresolved server addresses, put them in a list and shuffle. + */ + private void init(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = this.resolver.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } + Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible

connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); --- End diff -- can we save a call to `getHostString` by using `curHostString`?

146. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168572502 --- Diff: src/java/test/org/apache/zookeeper/test/ClientPortBindTest.java --- @@ -104,7 +104,7 @@ public void testBindByAddress() throws Exception { try { startSignal.await(CONNECTION_TIMEOUT, TimeUnit.MILLISECONDS); - Assert.assertTrue("count == 0", startSignal.getCount() == 0); + Assert.assertTrue("count == " + startSignal.getCount(), startSignal.getCount() == 0); --- End diff -- I'm not a huge fan of calling `getCount` more than once, since I think the value could change between invocations. Why not just use the return value from `await` in the line above?
147. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168576884 --- Diff: src/java/test/org/apache/zookeeper/test/StaticHostProviderTest.java --- @@ -119,6 +120,68 @@ public void testTwoInvalidHostAddresses() { new StaticHostProvider(list); } + @Test + public void testReResolvingSingle() { + byte size = 1; + ArrayList<InetSocketAddress> list = new ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to a single address + list.clear(); --- End diff -- do we need this?
148. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168575513 --- Diff: src/java/test/org/apache/zookeeper/test/ReadOnlyModeTest.java --- @@ -239,13 +243,13 @@ public void testSessionEstablishment() throws Exception { public void testSeekForRwServer() throws Exception { // setup the logger to capture all logs - Layout layout = Logger.getRootLogger().getAppender("CONSOLE") + Layout layout = org.apache.log4j.Logger.getRootLogger().getAppender("CONSOLE") --- End diff -- I guess my real question is, do we need any of the changes in this file?
149. Github user afine commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168576090 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -47,59 +51,169 @@ private int currentIndex = -1; + // Don't re-resolve on first next() call + private boolean connectedSinceNext = true; + + private Resolver resolver; + /** * Constructs a SimpleHostSet. - * + * * @param serverAddresses * possibly unresolved ZooKeeper server addresses * @throws IllegalArgumentException * if serverAddresses is empty or resolves to an empty list */ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { + this.resolver = new Resolver() { + @Override + public InetAddress[] getAllByName(String name) throws UnknownHostException { + return InetAddress.getAllByName(name); + } + }; + init(serverAddresses); + } + + /** + * Introduced for testing purposes. getAllByName() is a static method of InetAddress, therefore cannot be easily mocked. + * By abstraction of Resolver interface we can easily inject a mocked implementation in tests. + * + * @param serverAddresses + * possibly unresolved ZooKeeper server addresses + * @param resolver + * custom resolver implementation + * @throws IllegalArgumentException + * if serverAddresses is empty or resolves to an empty list + */ + public StaticHostProvider(Collection<InetSocketAddress> serverAddresses, Resolver resolver) { + this.resolver = resolver; + init(serverAddresses); + } + + /** + * Common init method for all constructors. + * Resolve all unresolved server addresses, put them in a list and shuffle. + */ + private void init(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = this.resolver.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch

(UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } + Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; --- End diff -- instead of exposing all these metrics, which involve making changes to application logic. Why don't we just expose the `serverAddresses` to the test?

150. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168580127 --- Diff: src/java/main/org/apache/zookeeper/client/StaticHostProvider.java --- @@ -58,48 +61,122 @@ public StaticHostProvider(Collection<InetSocketAddress> serverAddresses) { for (InetSocketAddress address : serverAddresses) { try { - InetAddress ia = address.getAddress(); - InetAddress resolvedAddresses[] = InetAddress.getAllByName((ia != null) ? ia.getHostAddress() : - address.getHostName()); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(getHostString(address)); for (InetAddress resolvedAddress : resolvedAddresses) { - // If hostName is null but the address is not, we can tell that - // the hostName is an literal IP address. Then we can set the host string as the hostname - // safely to avoid reverse DNS lookup. - // As far as i know, the only way to check if the hostName is null is use toString(). - // Both the two implementations of InetAddress are final class, so we can trust the return value of - // the toString() method. - if (resolvedAddress.toString().startsWith("/") - && resolvedAddress.getAddress() != null) { - this.serverAddresses.add(- new InetSocketAddress(InetAddress.getByAddress(- address.getHostName(), - resolvedAddress.getAddress()), - address.getPort())); - } else { - this.serverAddresses.add(new InetSocketAddress(resolvedAddress.getHostAddress(), address.getPort())); - } + this.serverAddresses.add(new InetSocketAddress(resolvedAddress, address.getPort())); } } catch (UnknownHostException e) { LOG.error("Unable to connect to server: {}", address, e); } } - + if (this.serverAddresses.isEmpty()) { throw new IllegalArgumentException("A HostProvider may not be empty!"); } Collections.shuffle(this.serverAddresses); } + /** + * Evaluate to a hostname if one is available and otherwise it returns the + * string representation of the IP address. + * + * In Java 7, we have a method getHostString, but earlier versions do not support it. + * This method is to provide a replacement for InetSocketAddress.getHostString(). + * + * @param addr + * @return Hostname string of address parameter + */ + private String getHostString(InetSocketAddress addr) { + String hostString = ""; + + if (addr == null) { + return hostString; + } + if (!addr.isUnresolved()) { + InetAddress ia = addr.getAddress(); + + // If the string starts with '/', then it has no hostname + // and we want to avoid the reverse lookup, so we return + // the string representation of the address. + if (ia.toString().startsWith("/")) { + hostString = ia.getHostAddress(); + } else { + hostString = addr.getHostName(); + } + } else { + // According to the Java 6 documentation, if the hostname is + // unresolved, then the string before the colon is the hostname. + String addrString = addr.toString(); + hostString = addrString.substring(0, addrString.lastIndexOf(':')); + } + + return hostString; + } + public int size() { return serverAddresses.size(); } + // Counts the number of addresses added and removed during + // the last call to next. Used mainly for test purposes. + // See StasticHostProviderTest. + private int nextAdded = 0; + private int nextRemoved = 0; + + public int getNextAdded() { + return nextAdded; + } + + public int getNextRemoved() { + return nextRemoved; + } + public InetSocketAddress next(long spinDelay) { - ++currentIndex; - if (currentIndex == serverAddresses.size()) { - currentIndex = 0; + // Handle possible connection error by re-resolving hostname if possible + if (!connectedSinceNext) { + InetSocketAddress curAddr = serverAddresses.get(currentIndex); + String curHostString = getHostString(curAddr); + if (!curHostString.equals(curAddr.getAddress().getHostAddress())) { + LOG.info("Resolving again hostname: {}", getHostString(curAddr)); + try { + int thePort = curAddr.getPort(); + InetAddress resolvedAddresses[] = InetAddress.getAllByName(curHostString); + nextAdded = 0; + nextRemoved = 0; + if (resolvedAddresses.length == 1) { + serverAddresses.set(currentIndex, new

InetSocketAddress(resolvedAddresses[0], thePort)); + nextAdded = nextRemoved = 1; + LOG.debug("Newly resolved address: {}", resolvedAddresses[0]); + } else { + int i = 0; + while (i < serverAddresses.size()) { + if (getHostString(serverAddresses.get(i)).equals(curHostString) && --- End diff -- That's exactly the point. (think I've already suggested the same thing in an outdated post)

151. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168763559 --- Diff: src/java/test/org/apache/zookeeper/test/ReadOnlyModeTest.java --- @@ -239,13 +243,13 @@ public void testSessionEstablishment() throws Exception { public void testSeekForRwServer() throws Exception { // setup the logger to capture all logs - Layout layout = Logger.getRootLogger().getAppender("CONSOLE") + Layout layout = org.apache.log4j.Logger.getRootLogger().getAppender("CONSOLE") --- End diff -- Makes sense. I revert the changes.
152. Github user anmolnar commented on a diff in the pull request: https://github.com/apache/zookeeper/pull/451#discussion_r168764368 --- Diff: src/java/test/org/apache/zookeeper/test/StaticHostProviderTest.java --- @@ -119,6 +120,68 @@ public void testTwoInvalidHostAddresses() { new StaticHostProvider(list); } + @Test + public void testReResolvingSingle() { + byte size = 1; + ArrayList<InetSocketAddress> list = new ArrayList<InetSocketAddress>(size); + + // Test a hostname that resolves to a single address + list.clear(); --- End diff -- Removed.
153. Linking to an earlier related issue.
154. Github user ijuma commented on the issue: https://github.com/apache/zookeeper/pull/451 Thanks for picking this up @anmolnar, looking forward to this being fixed. :)
155. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 @ijuma Sure, no problem. I'm waiting for some feedback from the community here and on the mailing list and I hope I can commit soon.
156. Github user ijuma commented on the issue: https://github.com/apache/zookeeper/pull/451 @anmolnar, it's been more than 1 month since the last comment on this PR. Is there anything that still needs to be addressed? The original PR was submitted in January 2017 and it got stalled after a while, I'm hoping the same doesn't happen here. :)
157. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 @ijuma I feel your pain. :) No worries, I'm on it. Doing my best to push committers and others to review changes. Additionally I'd like to make a small refactoring to the proposed logic before committing, because I'm not entirely convinced about this manual caching/shuffling logic that was implemented originally. You can see the details in one of my comments above and on the `dev` mailing list.
158. Github user anmolnar commented on the issue: https://github.com/apache/zookeeper/pull/451 @afine @ijuma I've finished refactoring of StaticHostProvider. The implementation follows that I explained in my email as **Option-3**: > - Do not cache IPs, > - Shuffle the names and resolve with getAllByName() every time when next() is called, > - Use getAllByName(), but shuffle the list and return the first IP to properly handle hostnames associated with multiple IPs, > - JDK's built-in caching will prevent name servers from being flooded and will do the re-resolution automatically when cache expires, The `Resolver` interface which is also introduced in this patch is the solution for the problem that @afine raised: to properly mock out the `getAllByName()` call in unit tests.
159. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 9 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1756//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1756//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1756//console This message is automatically generated.
160. +1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 9 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit

tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1757//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1757//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1757//console This message is automatically generated.

161. +1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 9 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1759//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1759//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1759//console This message is automatically generated.
162. Patch committed to branch-3.4 as [https://github.com/apache/zookeeper/commit/2e26c8836edc800c60b204a1d3da0285edb415d6] This Jira will be resolved after the patch is ported to branch-3.5 and master.
163. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1765//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1765//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1765//console This message is automatically generated.
164. +1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1766//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1766//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1766//console This message is automatically generated.
165. +1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1787//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1787//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1787//console This message is automatically generated.
166. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-

build/1788//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1788//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1788//console This message is automatically generated.

167. +1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1789//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1789//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1789//console This message is automatically generated.
168. +1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1842//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1842//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1842//console This message is automatically generated.
169. Temporary resolve issue for 3.4.13 so we can generate release note. Will reopen issue for 3.5 and 3.6.
170. again need to temporarily resolve the issue to get the right release notes for 3.4.13.
171. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 1 new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. -1 contrib tests. The patch failed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1924//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1924//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1924//console This message is automatically generated.
172. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 1 new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1927//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1927//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1927//console This message is automatically generated.
173. -1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 1 new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1930//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1930//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output:

https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1930//console This message is automatically generated.

174. +1 overall. GitHub Pull Request Build +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 3.0.1) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1934//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1934//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-github-pr-build/1934//console This message is automatically generated.
175. committed to master: https://github.com/apache/zookeeper/commit/0a311873deb1847703c9b62716c626ce43d4ba48 branch-3.5: https://github.com/apache/zookeeper/commit/1e65b9f4873fc995308972433ea8a664e98fe41f