Item 354
git_comments:

1. Initialize ZkBaseDataAccessor with usesExternalZkClient = false so that ZkBaseDataAccessor::close() would close ZkClient as well to prevent thread leakage
2. Initialize BestPossibleExternalViewVerifier with usesExternalZkClient = false so that BestPossibleExternalViewVerifier::close() would close ZkClient to prevent thread leakage
3. false by default
4. usesExternalZkClient = true because ZkClient is given by the caller at close(), we will not close this ZkClient because it might be being used elsewhere
5. Set the flag since external ZkClient is used
6. Initialize ClusterLiveNodesVerifier with usesExternalZkClient = false so that ClusterLiveNodesVerifier::close() would close ZkClient to prevent thread leakage
7. usesExternalZkClient = true because ZkClient is given by the caller at close(), we will not close this ZkClient because it might be being used elsewhere
8. Set the flag since external ZkClient is used
9. false by default
10. usesExternalZkClient = true because ZkClient is given by the caller at close(), we will not close this ZkClient because it might be being used elsewhere
11. Initialize StrictMatchExternalViewVerifier with usesExternalZkClient = false so that StrictMatchExternalViewVerifier::close() would close ZkClient to prevent thread leakage

git_commits:

1. **summary:** Fix ZkClient leakage by correctly setting usesExternalZkClient flag (#1562)
   message: Fix ZkClient leakage by correctly setting usesExternalZkClient flag (#1562) ZkBaseDataAccessor and Helix API that uses it using its Builder were having zkClient thread leak issues because the usesExternalZkClient flag was not being set properly. Also, the family of Verifiers were having a similar issue. This change solves this by making sure the private/protected constructors used by the Builders of these classes set the flag correctly to false so that in their respective close() functions, the underlying zkClient (created by the Builder) would get closed.
   label: code-design

github_issues:

1. **title:** ZkClient leak in select Helix API classes
   body: ### Describe the bug There was a report saying ZkBaseDataAccessor, when created using its Builder, was leaking ZkClient threads. A similar leak was observed for Verifier classes. ### To Reproduce The Builder does not properly use a private constructor that sets the usesExternalZkClient flag correctly, so it is reproducible. ### Expected behavior Create a private constructor used by the Builder and make sure the flag is set correctly. See the example of ConfigAccessor and ZkHelixAdmin - both do this correctly. ### Additional context Add any other context about the problem here.
   label: code-design
2. **title:** ZkClient leak in select Helix API classes
   body: ### Describe the bug There was a report saying ZkBaseDataAccessor, when created using its Builder, was leaking ZkClient threads. A similar leak was observed for Verifier classes. ### To Reproduce The Builder does not properly use a private constructor that sets the usesExternalZkClient flag correctly, so it is reproducible. ### Expected behavior Create a private constructor used by the Builder and make sure the flag is set correctly. See the example of ConfigAccessor and ZkHelixAdmin - both do this correctly. ### Additional context Add any other context about the problem here.
   label: code-design

github_issues_comments:

github_pulls:

1. **title:** Fix ZkClient leakage by correctly setting usesExternalZkClient flag
   body: ### Issues - [x] My PR addresses the following Helix issues and references them in the PR description: Resolves #1561 ### Description - [x] Here are some details about my PR, including

screenshots of any UI changes: ZkBaseDataAccessor and Helix API that uses it using its Builder were having zkClient thread leak issues because the usesExternalZkClient flag was not being set properly. Also, the family of Verifiers were having a similar issue. This change solves this by making sure the private/protected constructors used by the Builders of these classes set the flag correctly to false so that in their respective close() functions, the underlying zkClient (created by the Builder) would get closed. ### Tests - [x] The following tests are written for this issue: Manually monitored and tested - [x] The following is the result of the "mvn test" command on the appropriate module: > [INFO] [INFO] Results: [INFO] [INFO] Tests run: 1251, Failures: 0, Errors: 0, Skipped: 0 [INFO] [INFO] ---------- [INFO] BUILD SUCCESS [INFO] ---------- [INFO] Total time: 01:25 h ### Documentation (Optional) - In case of new functionality, my PR adds documentation in the following wiki page: (Link the GitHub wiki you added) ### Commits - My commits all reference appropriate Apache Helix GitHub issues in their subject lines. In addition, my commits follow the guidelines from "[How to write a good git commit message] (http://chris.beams.io/posts/git-commit/)": 1. Subject is separated from body by a blank line 1. Subject is limited to 50 characters (not including Jira issue reference) 1. Subject does not end with a period 1. Subject uses the imperative mood ("add", not "adding") 1. Body wraps at 72 characters 1. Body explains "what" and "why", not "how" ### Code Quality - My diff has been formatted using helix-style.xml (helix-style-intellij.xml if IntelliJ IDE is used)

github_pulls_comments:

1. > A general question, I see some of verifiers use external client to be true, some are false. Is that on purpose? Could you elaborate more why we do this? Good question to ask here. Some constructors have `usesExternalZkClient = false` and others have `usesExternalZkClient=true`. Why? The answer lies in where the ZkClient object is being passed in. You will see a pattern where the top-level, public constructors that accept an instance of `zkClient` (external zkclient) have the flag set to true, and the private constructors whose zkClient objects originate from the Builder (created from `createZkClient()` method) will have this flag set to false. The difference here is that if the flag is set to true, the zkclient will NOT be closed because it is an external ZkClient (that might be used elsewhere - we don't control the fate of this ZkClient). If the flat is set to false, then the ZkClient will be closed because we control this ZkClient (we created it). @dasahcc let me know if this clarifies it. I've added more descriptive comments.
2. **body:** This PR is ready to be merged. ZkBaseDataAccessor and Helix API that uses it using its Builder were having zkClient thread leak issues because the usesExternalZkClient flag was not being set properly. Also, the family of Verifiers were having a similar issue. This change solves this by making sure the private/protected constructors used by the Builders of these classes set the flag correctly to false so that in their respective close() functions, the underlying zkClient (created by the Builder) would get closed.
   label: code-design

github_pulls_reviews:

1. **body:** minor: should we throw out IllegalArgumentException instead?
   label: code-design
2. Yes. Good catch.

jira_issues:

jira_issues_comments: