Item 371
git_comments:

1. registration paths
2. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * 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 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
3. re-read the bookie list
4. construct the zookeeper
5. * * ZooKeeper based {@link RegistrationClient}.
6. initialize resources
7. initialize registration client
8. initialize feature provider
9. create readonly bookies node if not exists
10. this callback is already not executed in zookeeper thread
11. * * Set registration manager class * * @param regClientClass * ClientClass
12. * * Get Registration Client Class. * * @return registration manager class.
13. Registration Client
14. * * Initialize the registration client with provided resources. * * <p>The existence of zkSupplier is for backward compatability. * @param conf client configuration * @param statsLogger stats logger * @param zkOptional a supplier to supply zookeeper client. * @return RegistrationClient initialize(ClientConfiguration conf, ScheduledExecutorService scheduler, StatsLogger statsLogger, Optional<ZooKeeper> zkOptional) throws BKException; @Override void close(); /** * Get the list of writable bookie identifiers. * * @return a future represents the list of writable bookies.
15. * * Watch the changes of bookies. * * <p>The topology changes of bookies will be propagated to the provided listener. * * @param listener listener to receive the topology changes of bookies.
16. * * Get the list of readonly bookie identifiers. * * @return a future represents the list of readonly bookies.
17. * * Unwatch the changes of bookies. * * @param listener listener to receive the topology changes of bookies.

git_commits:

1. **summary:** ISSUE #666: Introduce registration client for bookkeeper client to do bookie discover
   message: ISSUE #666: Introduce registration client for bookkeeper client to do bookie discover Descriptions of the changes in this PR: while in BOOKKEEPER-628, It is trying to improve/generalize the bookie registration process. This PR follows the work of #663, which provide bookie side registration, and it is for client side work, it tries to introduce RegistrationClient for bookkeeper client to do bookie discover. This PR follows the work of #663, which provide bookie side registration, Author: Jia Zhai <zhaijia@apache.org> Reviewers: Sijie Guo <sijie@apache.org> This closes #667 from zhaijack/client_registration2, closes #666

github_issues:

1. **title:** Introduce registration client for bookkeeper client
   body: while in [BOOKKEEPER-628](https://issues.apache.org/jira/browse/BOOKKEEPER-628), It is trying to improve/generalize the bookie registration process. This issue is for client side work, it tries to introduce RegistrationClient for bookkeeper client to do bookie discover. And this follows the work of #662.
2. **title:** Introduce registration client for bookkeeper client
   body: while in [BOOKKEEPER-628](https://issues.apache.org/jira/browse/BOOKKEEPER-628), It is trying to improve/generalize the bookie registration process. This issue is for client side work, it tries to introduce RegistrationClient for bookkeeper client to do bookie discover. And this follows the work of #662.

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
2. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
3. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
4. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
5. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
6. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
7. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
8. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
   label: code-design
9. **summary:** Improve bookie registration interface
   description: The idea is to improve/generalize the bookie registration process
10. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
11. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
12. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
13. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
14. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
15. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
16. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
17. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
18. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
19. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
20. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
21. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
22. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
23. **summary:** Improve bookie registration interface
    description: The idea is to improve/generalize the bookie registration process
    label: code-design
24. **summary:** Improve bookie registration interface

description: The idea is to improve/generalize the bookie registration process

jira_issues_comments:

1. I've created a patch to showcase the interfaces. Tries to generalize interfaces, these will be used by bkserver & bkclient.
2. thanks [~rakeshr] since bookie registration involve zookeeper session expire handling, how you could make it generic? general comments on your patch: 1. please involve version in this interface. you could refere LedgerManager interface. 2. this abstraction consider both client and server side change. so you need to ensure client and bookie agrees on using same implementation of this interface. check LedgerManagerFactory. it would be better to separate client-only, server-only and common part into different interfaces and managed by one factory.
3. UnKnown should be Unknown. It's a single word. This stuff has a good bit of overlap with the rack awareness stuff sijie did. Have you looked at that patch? I agree with Sijie that something needs to be done about session expire handling. Perhaps register could take a callback, and you watch on the registered node. If it disappears, then you trigger the callback. Generally I like this idea of this though.
4. Thanks [~hustlmsp], [~ivank@yahoo-inc.com] for the reviews. I've separated out the client/server interfaces and if agrees, I'll integrate this to our code base. bq.zookeeper session expire handling Presently ZooKeeperWatcherBase callback is doing the Expiry handling. I'm thinking to move the zookeeper instantiation to the RegistrationManager#init() method, and returns zk object. In bkclient, if user doesn't pass zk object will do the zk instantiation and returns the object? How does it sound?
5. {code} I'm thinking to move the zookeeper instantiation to the RegistrationManager#init() method, and returns zk object. {code} in general, the interface is to allow different implementations. returning a zk object against its purpose. this interface should be something like HubServerManager interface in hedwig. https://github.com/apache/bookkeeper/blob/trunk/hedwig-server/src/main/java/org/apache/hedwig/server/topics/HubServerManager.java
6. I've attached modified interfaces, added ManagerListener to listen for zk connection events. [~ikelly] [~hustlmsp] please have a look at this and if ok, will start implementing the idea.
7. the interface looks good. but I would step back asking the question: what is the goal of this interface? it sounds to me that the interface is to hide zookeeper specific implementations. but from the interface, it is already tight with zookeeper (RegistrationManager#init(AbstractConfiguration conf, ZooKeeper zk, int managerVersion)). As this interface doesn't block other tickets and all these stuffs (e.g. bookie registration, cookies) are quite zookeeper specific operations, I'd prefer generifying the interface after fixed zookeeper session handling in BOOKEEPER-537. so we could get clearer how the interface would look like. Thoughts?
8. **body:** There's two reasons to provide an interface. Firstly to create well defined isolated contracts between the components of your system, and secondly to allow multiple implementations of this contract to coexist. In this case, we only want the first, as there's no driving usecase to move away from zookeeper for bookie registration. Following from this, RegistrationManagerFactory is unneeded. The interfaces in RegistrationManager are mostly fine though. I don't think that BookieRegistrationManager should have separate calls for the Cookie. bookieId <-> cookie is a 1-1 relationship. Why does ClientRegistrationManager have a createBookieInstance call? Also, the placement policy should have nothing to do with this interface.It's a decision that can be taken in total isolation on the client. I agree with Sijie that it would be good to sort out the session handling before doing this work though. Or doing it in tandem. At least for the listener part.
   label: code-design
9. Thanks for the comments. Here my idea was to unify the zk usage in bookie server operations, also in client side operations through one registration interface. Presently the zk apis usage are scattered in Bookie, Cookie. IMHO, this will be helpful to focus at common place for any ZK related logics like - ZK session handling, ZK acls. bq.Why does ClientRegistrationManager have a createBookieInstance call? Yeah, this should be at server side, which will be used by bookie format logic.
10. Thanks for the comments. Here my idea was to unify the zk usage in bookie server operations, also in client side operations through one registration interface. Presently the zk apis usage are scattered in Bookie, Cookie. IMHO, this will be helpful to focus at common place for any ZK related logics like - ZK session handling, ZK acls. bq.Why does ClientRegistrationManager have a createBookieInstance call? Yeah, this should be at server side, which will be used by bookie format logic.
11. bq.I agree with Sijie that it would be good to sort out the session handling before doing this work though. Or doing it in tandem. At least for the listener part. I agree with both of you taking up this after pushing

BOOKEEPER-537. [~hustlmsp] whats your opinion about pushing listener part separately with this JIRA.

12. [~rakeshr] [~ikelly] isn't ManagerListener related to zookeeper session handling? could we think of it after handling session expires?
13. Yeah, zk session handling. Sure will revist after BOOKEEPER-537, I'will add dependency tag.
14. Do you mean push the listener stuff in a different JIRA?
15. I thought of pushing smaller chunk through subtask, but its overlapping with BOOKEEPER-537 and postponed my plans. Any thoughts?
16. which smaller chunk though?
17. bq.which smaller chunk though? I meant, RegistrationManager#ManagerListener part, which will be used to listen for zk connection events.
18. Hmm, actually looking at this again, perhaps it could go in before BOOKKEEPER-537, if the previous comments are addressed. Then BOOKKEEPER-537 could build on top.
19. I don't think it is a good idea to refactor things that would be changed in future. since you don't actually know whether the refactor meets the changes or not. if this is really really need to be in, I would suggest doing the metadata related part (not session expire part), like Cookies. And we could iterate the session part later.
20. moved to 4.4.0 as session expire handling moved to 4.4.0
21. [~zhaijia] Have you seen this? You're working on something similar now, no?
22. Migrated to github https://github.com/apache/bookkeeper/issues/621