Item 333
git_comments:

1. * * Sends ratis command to destroy pipeline on the given datanode. * * @param dn - Datanode on which pipeline needs to be destroyed * @param pipelineID - ID of pipeline to be destroyed * @param ozoneConf - Ozone configuration * @param grpcTlsConfig - grpc tls configuration * @throws IOException
2. * * Utility class for Ratis pipelines. Contains methods to create and destroy * ratis pipelines.
3. * 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.
4. * * Removes pipeline from SCM. Sends ratis command to destroy pipeline on all * the datanodes. * * @param pipeline - Pipeline to be destroyed * @param ozoneConf - Ozone configuration * @param grpcTlsConfig * @throws IOException
5. Setting to zero by default means this limit doesn't take effect.
6. Upper limit for how many pipelines can be created. Only for test purpose now.
7. Datanodes from pipeline in some states can also be considered available for pipeline allocation. Thus the number of these pipeline shall be deducted from total heaviness calculation.
8. no limit applied.
9. Randomly picks nodes when all nodes are equal or factor is ONE.
10. Per datanode limit
11. Global limit
12. Only put limits for Factor THREE pipelines.
13. * * Get the number of pipeline created. * @return number of pipeline
14. pipeline creation failed this time.
15. allow only one FACTOR THREE pipeline.
16. * * Sets the total number of pipelines to create. * @param val number of pipelines * @return MiniOzoneCluster.Builder
17. MiniOzoneCluster should have global pipeline upper limit.

git_commits:

1. **summary:** HDDS-1569 Support creating multiple pipelines with same datanode. Contributed by Li Cheng.
   message: HDDS-1569 Support creating multiple pipelines with same datanode. Contributed by Li Cheng. This closes #28

github_issues:

github_issues_comments:

github_pulls:

1. **title:** HDDS-1569 Support creating multiple pipelines with same datanode
   body: ## NOTICE https://issues.apache.org/jira/browse/HDDS-1569 related PR: https://github.com/apache/hadoop/pull/1431 Changes: 1. Use PipelinePlacementPolicy as default for Factor THREE Ratis pipeline. 2. Handle differently in some parts for Factor ONE and Factor THREE pipeline. 3. Add limits for pipeline creation. 4. Adjust a bunch of unit tests accordingly.

github_pulls_comments:

1. /retest
2. /retest
3. /retest

4. /retest
5. +1, pending CI. Thanks for the update @timmylicheng

github_pulls_reviews:

1. **body:** .datanode can be removed.
   label: code-design
2. If this property is only for test, suggest remove it from the ozone-default.xml
3. Any change here?
4. Don't change this level. The action is "close pipeline" due to some error happened on Datanode side.
5. should we divide the "heavyNodeCriteria * nodeManager.getNodeCount(HddsProtos.NodeState.HEALTHY" using the "factor" ?
6. define a new resultcode for this case or use FAILED_TO_FIND_SUITABLE_NODE
7. Suggest change Node2PipelineMap, use Pipeline instead of PipelineID as the value of the Map, to simpify this query.
8. Import * is not recommended.
9. !p.getPipelineState().equals(PipelineState.CLOSED)
10. **body:** line longer than 80 chars
    label: code-design
11. SCMException is a generic exception, please use specific exception if possible
12. setPipelineNumberLimit?
13. **body:** > Any change here? Not effective changes. It was due to complaints from checkstyle
    label: code-design
14. Ok. Will remove from ozone-default.xml
15. Ok sure
16. Rolled it back
17. IDE did it. Change it manually
18. **body:** Haven't made logical changes to it. Just quiet off checkstyle. Intend to keep it this way.
    label: code-design
19. Good point. Should divide by factor number.
20. **body:** checkstyle doesn't seem to check imports. Don't bother to adjust it until checkstyle complains.
    label: code-design
21. Updated.
22. So all the policies including ContainerPlacementPolicy are throwing out SCMException with specified reason in messages. The PipelinePlacementPolicy follows them and so as upstream SCMPipelineManager. So here I align the old InsufficientDatanodesException with SCMException. I believe with specified error message, it's ok to use a general exception since the test is specifically testing pipeline creation.
23. Ok
24. As I looked a little deeper, changing Node2PipelineMap mapping may affect more than this commit. Shall we have a separate discussion on the internal data structure for NodeStateManager?
25. I switch to traverse the list from node manager side.
26. **body:** why do we need to add the dn->pipelineId in 75 since line 73 already add it?
    label: code-design
27. **body:** Can we add some description for the meaning of default 0?
    label: documentation
28. This seems to be a case where the PipelineStateMap mismatch with the Node2PipelineMap. Do you see cases from unit tests or cluster?
29. **body:** NIT: suggest rename heavyNodeCriteria to maxPipelinePerDatanode. heavyNodeCriteria can be a function name using maxPipelinePerDatanode.
    label: code-design
30. Or should we consider only OPEN pipeline here?
31. Do we still assume one RATIS.ONE pipeline per DN here?
32. **body:** Should we move this logic into RatisPipelineProvider so that the exception handling will natually fit into the try/final?
    label: code-design
33. Can we use assertEquals with some message so that when it fails, we can see the difference in the ResultCode.
34. Do we have specific exception result code to check here?

35. Is there a reason that we need to set a large per DN pipeline limit here?
36. **body:** Can you add more comments here? If the # of SCM pipeline limit is 3, should we set OZONE_SCM_PIPELINE_NUMBER_LIMIT to 3 instead of 2 * numOfDatanodes?
    label: documentation
37. **body:** NIT: can we expand the wildcard import?
    label: code-design
38. **body:** NIT: expand wildcard import
    label: code-design
39. Should we expose a method for MiniOzoneCluster to set per DN pipeline limit?
40. **body:** NIT: rename to setTotalPipelineNumLimit
    label: code-design
41. Line 73 is adding v to new keyset when key is absent and line 75 is adding v to the existing keyset. Reason why we don't have to have extra if here is computeIfAbsent and computeIfPresent are both judging if key is present or not. So we don't have to see if Map.get(k) is null.
42. Sure.
43. Sure
44. Yes. The RATIS ONE pipeline is still using old fashion of picking up nodes. Only RATIS THREE pipeline is using the complex new fashion in PipelinePlacementPolicy.
45. Good point. Updating.
46. Sure
47. It doesn't have one before, while I can add some check here.
48. I can make it smaller.
49. 3 means the default number pipelineNumLimit, which means there is no use-set limit. I should've made this a global reference.
50. IDE did this. Fixing
51. Conf setting is universal way of setting the limit. The current way in MiniOzoneCluster to set the global pipeline limit is rather a convenient method for internal use.
52. Sure
53. IDE did this. Fixing
54. Correct me if I'm wrong, dn2ObjectMap.computeIfAbsent(dnId, k -> ConcurrentHashMap.newKeySet()) ensures for a dnId we will have a KeySet(). Then the returned value (KeySet) allows line 73 to continue the operation to add pipeline id with a .add(pipeline.getId). JDK document can be found here: https://docs.oracle.com/javase/8/docs/api/java/util/Map.html#computeIfAbsent-K-java.util.function.Function-
55. Make sense to me.
56. I'm fine with current approach. I'm asking because I see a few other mini ozone cluster based tests require similar settings, adding a wrapper can make this easier. But we can do that as a follow up jira.
57. Other pipelines like STALE pipelines would possibly turn into OPEN at any time. So here I just deduct out CLOSED pipelines.
58. Not from test, but it's an exception thrown from stateManager.getPipeline method. So I figured adding some error messages for it because the pipeline creation won't stop here.
59. Yea. Line 75 is for when we already one existing keyset. For multi pipeline case, the keyset may be created by previous pipeline allocation and the new one would just be added into the existing keyset.
60. The document says existing value(keyset) will be returned. So the line 74-75 is not necessary. Returns: the current (existing or computed) value associated with the specified key, A good example can be found here: https://stackoverflow.com/a/39268896
61. I see. Thanks for the tipping. Based on the example here, Shall I update the removePipeline method? I'm making a change and please take a look.

jira_issues:

1. **summary:** Add ability to SCM for creating multiple pipelines with same datanode
   description: - Refactor _RatisPipelineProvider.create()_ to be able to create pipelines with datanodes that are not a part of sufficient pipelines - Define soft and hard upper bounds for pipeline membership - Create SCMAllocationManager that can be leveraged to get a candidate set of datanodes based on placement policies - Add the datanodes to internal datastructures

jira_issues_comments:

1. -- 'Define soft and hard upper bounds for pipeline membership' Assuming it's talking about how many pipelines every datanode can be engaged in, it's defined as node heaviness in #HDDS-1577 as in pipeline placement policy. And the current default hard upper limit is 5. That's said, the max number of pipelines where every datanode can be placed may vary after sufficient field testing.
2. [~xyao] [~Sammi] [~swagle] In terms of the 'SCMAllocationManager', do we need to support concurrent pipeline placement? Or we just support concurrent request and use a blocking queue to do sequential allocation? One major cost for concurrent allocation is that info in datanode (like datanode <-> pipeline mapping) must have locks, which could add complexity to failure handling in pipeline placement since counts + 1 has to happen before ultimate placement success and the fallback process would require the lock again. There is absolutely some solution to it (like count + 1 upon registry and mapping + 1 upon placement success, which makes it only lock count + 1). I'm just having questions on whether it's necessary because usually these larger resource control plane action can be sequential as long as it's reasonably quick.
3. [~swagle] One more question, what do you mean by "internal datastructures"? What data structures do you think datanodes (assuming DatanodeDetails) should be part of?
4. [~timmylicheng] I believe by internal data structures, I meant the _PipelineStateMap_ but I will take a look at the doc/code and get back if any other data structures that need an update. The only way new pipelines are created should be through the Background pipeline creator job. We should not create any pipelines on client requests, in fact, we should assume pipelines are available to SCM already and no ad-hoc pipelines will be created. If a single thread creates pipelines, no need of blocking queues or synchronization, except the utilization counters used for selecting pipeline need to be Atomic (current its round-robin allocation).
5. **body:** I separate SCMAllocationManager to another Jira https://issues.apache.org/jira/browse/HDDS-2116. I think the current BackgroundCreator has a sync way to create pipeline and we can start from there. As [~swagle] mentioned, we now don't assume many ad-hoc pipeline creation operations, the current sync call from background shall be able to handle this. Once we hit performance bottle neck for pipeline creation, we can invest to have a SCMAllocationManager with self-managed thread model. **label:** code-design
6. PR [https://github.com/apache/hadoop-ozone/pull/28] is under review.
7. Thanks [~timmylicheng] for the contribution and all for the reviews. I've merged the PR to feature branch.