Item 321
git_comments:

1. Copyright 2016 Twitter. All rights reserved. Licensed 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.
2. * * Helper class that captures the RAM requirements of each component
3. Automatically adjust the number of containers
4. * * Assign a particular instance to a container with a given containerId * * @return true if the container incorporated the instance, otherwise return false
5. Get the instances using a resource compliant round robin allocation
6. Currently not yet support disk or cpu config for different components, so just use the default value.
7. * * Check whether the Instance has enough RAM and whether it can fit within the container limits. * * @param instanceResources The resources allocated to the instance * @return true if the instance is valid, false otherwise
8. * * Get the RAM requirements of all the components * * @return The list of components and their RAM requirements. Returns null if one or more * instances do not have valid resource requirements.
9. * * Allocate a new container taking into account the maximum resource requirements specified * in the config * * @return the number of containers
10. * * Get the instances' allocation based on the Resource Compliant Round Robin algorithm * * @return Map < containerId, list of InstanceId belonging to this container >
11. Copyright 2016 Twitter. All rights reserved. Licensed 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.
12. Construct the PackingPlan
13. Insert it into the map
14. Take the heron internal container into account and the application master for YARN scheduler
15. * * ResourceCompliantRoundRobin packing algorithm * <p> * This IPacking implementation generates a PackingPlan using a round robin algorithm. * <p> * Following semantics are guaranteed: * 1. Supports heterogeneous containers. * The user provides the number of containers to use as well as * the maximum container size and a padding percentage. * The padding percentage whose values range from [0, 100], determines the additional per container * resources allocated for system-related processes (e.g., the stream manager). * The algorithm might produce containers whose size is slightly higher than * the maximum container size provided by the user depending on the per-container instance allocation * and the padding configuration. * <p> * 2. The user provides the maximum CPU, RAM and Disk that can be used by each container through * the com.twitter.heron.api.Config.TOPOLOGY_CONTAINER_CPU_REQUESTED, * com.twitter.heron.api.Config.TOPOLOGY_CONTAINER_RAM_REQUESTED, * com.twitter.heron.api.Config.TOPOLOGY_CONTAINER_DISK_REQUESTED parameters. * If the parameters are not specified then a default value is used for the maximum container * size. * <p> * 3. The user provides a percentage of each container size that will be added to the resources * allocated by the container through the com.twitter.heron.api.Config.TOPOLOGY_CONTAINER_PADDING_PERCENTAGE * If the parameter is not specified then a default value of 10 is used (10% of the container size) * <p> * 4. The ram required for one instance is calculated as: * value in com.twitter.heron.api.Config.TOPOLOGY_COMPONENT_RAMMAP if exists, otherwise, * the default ram value for one instance. * <p> * 5. The cpu required for one instance is calculated as the default cpu value for one instance. * <p> * 6. The disk required for one instance is calculated as the default disk value for one instance. * <p> * 7. The ram required for a container is calculated as: * (ram for instances in container) + (paddingPercentage * ram for instances in container) * <p> * 8. The cpu required for a container is calculated as: * (cpu for instances in container) + (paddingPercentage * cpu for instances in container) * <p> * 9. The disk required for a container is calculated as: * (disk for instances in container) + ((paddingPercentage * disk for instances in container) * <p> * 10. The size of resources required by the whole topology is equal to * ((# containers used by ResourceCompliantRR)* size of each container * + size of a container that includes one instance with default resource requirements). * We add some resources to consider the Heron internal container (i.e. the one containing Scheduler * and TMaster). * <p> * 11. The pack() return null if PackingPlan fails to pass the safe check, for instance, * the size of ram for an instance is less than the minimal required value or .
16. Calculate the resource required for single instance
17. * * Test the scenario where container level resource config are set
18. * * Test the scenario ram map config is partially set
19. Ram for bolt should be the value in component ram map
20. Two components
21. Explicit set component ram map
22. No explicit resources required
23. Ram for bolt/spout should be the value in component ram map
24. Setup the spout parallelism
25. Setup the bolt parallelism
26. The containerRam should be ignored, since we set the complete component ram map
27. Explicit set resources for container
28. * * Test the scenario ram map config is fully set

29. The first container consists 2 spouts and 2 bolts (4 instances) and the second container consists of 2 spouts and 1 bolt (3 instances)
30. * * test even packing of instances
31. Copyright 2016 Twitter. All rights reserved. Licensed 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.
32. Set up the topology and its config
33. All instances' resource requirement should be equal So the size of set should be 1
34. The first 3 containers consist of 1 spout and 1 bolt (2 instances) and the fourth container consists of 1 spout (1 instance)
35. * * Test the scenario where the max container size is the default
36. * * Test the scenario where the user defined number of containers is not sufficient.
37. * * Test the scenario where the max container size is the default and padding is configured
38. Explicit set insufficient ram for container
39. Explicit set resources for container the value should be ignored, since we set the complete component ram map
40. Copyright 2016 Twitter. All rights reserved. Licensed 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.
41. * * Definition of Resources. Used to form packing structure output.
42. Set up the topology and its config
43. Explicit set insufficient ram for container

git_commits:

1. **summary:** Added the resource compliant round robin algorithm (#1160)
   message: Added the resource compliant round robin algorithm (#1160)

github_issues:

1. **title:** Improve Round Robin packing algo to provide memory guarantees for each instance and to use configurable padding.
   body: Currently the Round Robin algorithm can generate packing plans with instances whose assigned memory is neither one specified by the user nor the default one. The extension will ensure that the memory assigned to each instance is the default one unless specified by the user. The extension will also use configurable padding.
2. **title:** Improve Round Robin packing algo to provide memory guarantees for each instance and to use configurable padding.
   body: Currently the Round Robin algorithm can generate packing plans with instances whose assigned memory is neither one specified by the user nor the default one. The extension will ensure that the memory assigned to each instance is the default one unless specified by the user. The extension will also use configurable padding.

github_issues_comments:

github_pulls:

1. **title:** Added the resource compliant round robin algorithm
   body: fixes #1148 This branch includes the resource compliant round robin packing algorithm. Some classes were moved to utils so that both resource compliant RR and first fit decreasing use them.

github_pulls_comments:

1. @kramasamy This version of the round robin packing algorithm will by default return a null packing if the number of containers specified by the user is not sufficient to satisfy the resource requirements of the components. Optionally the user can configure the algorithm (using topologyConfig) to automatically increase the number of containers if the user-specified number is not sufficient so that a packing plan is created. Is this an acceptable behavior? Let me know if you agree with the defaults.
2. @avflor - the approach is fine with me. @maosongfu, @ajorgensen and @billonahill can you please check this PR?
3. One broader question - will it affect the existing tuning configuration of current running jobs? Will it require lesser resources than what is provided.
4. @kramasamy It might affect existing jobs if the number of containers provided as input is not sufficient to satisfy the memory requirements of each instance. The current round robin algorithm will reduce the memory requirements of each instance to fit into a smaller number of containers. This algorithm will fail. If I make the autoadjustment as default then this algo will not affect current jobs. They will not fail but they may use more containers than previously. This is because now the memory requirements of each instance are guaranteed to be satisfied. However with homogeneous containers you may end up seeing more resources being used than the current round robin. I think if you want to guarantee the memory resources per instance, this algorithm is better since it provides the guarantee and can automatically figure out the number of containers. However, if you are fine not providing this guarantee but use fewer homogeneous containers, then you should use the existing algo. In case of YARN, I would recommend the enhanced algo since it can support heterogeneous containers and thus will not waste resources. This algorithm supports configurable padding that is added on top of the container_ram defined in the topology. Thus your users, might see slightly bigger containers as it is not assumed that they take padding into account when defining

the container size. This is because the padding value is dependent on the number of instances packed in a container and thus is applied after packing. Let me know if you want me to change the default for auto-adjustment.
5. @avflor - thanks for the detailed explanation. conservatively, can we code this as a separate algorithm - something like Enhanced Round Robin. I can ask a few topologies to turn on the Enhanced Round Robin and see how it works before we roll out widely. If the existing algorithm is changed and we don't know its behavior, it will be difficult to roll back - unless we roll back the release. Thoughts?
6. @kramasamy I have already coded it as a separate algorithm under the round robin package so that it does not affect existing jobs. We can try it and see. I was thinking whether it makes sense to have the packing algo being configurable for each topology. E.g a topology where minimizing resources is important can use binpacking, another one round robin etc. What do you think?
7. @avflor - it is definitely possible to configure each topology with a different packing algorithm. All you have to do is during `heron submit` pass the option ``` --config-property "heron.class.packing.algorithm=<packing-classname>" ``` it should work. let me know if this does not work.
8. @kramasamy Cool I'll try that. I thought packing algo is a global config.
9. @avflor - you can override any config at the command line using `--config-property` as long as you know its implications. The only inconvenience is you have to type it every time you submit in the CLI. With this PR #1119, you could setup an .heronrc file in your home directory and set the config property there and you don't have to do in command line every time.
10. @billonahill @ajorgensen Thanks for the detailed feedback. Some parts of the code I copied from the current round robin algo and I'm not sure If I should change them or keep the same for consistency. I will mark individual changes so that you can clarify what is best to do.
11. @billonahill @kramasamy @ajorgensen I think I addressed all the comments. I moved the Resource class out of the PackingPlan and this change affected multiple files. Please let me know if anything else is needed.
12. @billonahill @kramasamy @ajorgensen BTW, is there any style formatter that applies the twitter guidelines like additional newlines? Checkstyle does not give me any errors on newlines and intellij does not fix the issue when reformatting the code. What do you usually use for that? Is there an intellij plugin for this?
13. @avflor - we already enforce checkstyle when you compile. probably the empty line rule might not be there.
14. **body:** @kramasamy Yeah, it's not there. Maybe we should add it. It will make the process much easier.
    label: code-design
15. **body:** Yeah, it looks like some version of this one might be the one we need to checkstyles:
    http://checkstyle.sourceforge.net/apidocs/com/puppycrawl/tools/checkstyle/checks/whitespace/EmptyLineSeparatorCheck.html
    https://github.com/twitter/heron/blob/master/tools/java/src/com/twitter/bazel/checkstyle/coding_style.xml I'm not sure which one is needed for IntelliJ, but we'd set it here:
    https://github.com/twitter/heron/blob/master/scripts/resources/idea/codeStyleSettings.xml
    label: code-design
16. **body:** @billonahill I think I made all the changes. Regarding the constants in the tests: A number 4 in an equation doesn't necessarily mean 4 spouts. It can reflect 2 spouts and 2 bolts in one container. So replacing the number with spoutParalellim (or BoltParallelsim) can be misleading. I made the change for the tests that this makes sense. I kept the constants for the tests that are more complex.
    label: code-design
17. **body:** Thanks @avflor for being patient with my comments. :) The changes made have made things much easier to grok. If the 4s and 3s don't align with bolt and spout counts, maybe a comment about what they are would do the trick. Without looking at the details of the algo it's hard to decipher why some of those numbers are being used in the test assertions.
    label: code-design
18. @billonahill I added some comments to clarify the numbers in the assertions. Hope it works now :)
19. Thanks @avflor! LGTM. :) @ajorgensen @maosongfu any further comments?
20. @ajorgensen - if you have any additional comments, let us know - so that we can merge it.
21. @avflor - can you please resolve the conflicts?
22. @kramasamy Fixed

github_pulls_reviews:

1. **body:** Sorry is `vpu vores` a typo that should say `cpu cores`?
   label: documentation
2. Extra new lines.
3. **body:** Same typo here it looks like
   label: documentation
4. extra new lines
5. So this will say that two ram requirements for different components will be equal correct. IE `new RamRequirement("foo", 10).equals(new RamRequirement("bar", 10))` would return true. The hashCode on the other hand takes the component name into account. So while two of these would be equal they would not hash to the same value. Just want to make sure this is intentional
6. Do we need TOPOLOGY_ADJUST_NUMBER_CONTAINERS? We could remove it and just state that the contract of this IPacking instance is that it will adjust.
7. **body:** This should contain more detailed info to help the user understand what happened (e.g., how many were needed, how many were requested, etc).
   label: code-design
8. **body:** This should include more info, like increasing from Z to Y for what reason.
   label: code-design

9. Typically we don't put model objects in a util package. Could we put the elsewhere and only have static utility helpers under util? A class representing RamRequirement seems like a core object, not a utility.
10. 4 here and below appears to be a magic number. What does it represent and should it be a constant or a config setting with a default?
11. PackingPlan has an inner class Resource, which is a tuple of these 3 objects. Should we make that an outer class and leverage it here?
12. remove empty line
13. extra newline
14. **body:** extra newline here and above
    label: code-design
15. do these 2 methods need to be protected?
16. This test should assert that a reasonable exception is thrown, instead of returning null
17. extra newline
18. **body:** why +1 and *10 and /100 here and elsewhere? could those become variables that implies what they are? same for other tests
    label: code-design
19. **body:** extra newline here and elsewhere in this class
    label: code-design
20. **body:** How is testContainerRequestedResources2 different than testContainerRequestedResources? Could we name them as such or include a description in the javadocs?
    label: documentation
21. **body:** For consistency and clarity should change these to numContainers and numAdjustments. noAdjustments sounds like a boolean.
    label: code-design
22. what does returning null mean? If one of the components isn't valid, no RamRequirements are returned for any of them? Should clarify the null semantics.
23. should clarify the semantics of what returning null means, which is unclear. It seems to happen if either ramRequirements are null (which appears to be a valid case) as well as if adjustNumberOfContainers if false, which appears to be a severe case.
24. **body:** let make this error message as descriptive as possible with things like the values that triggered the exception.
    label: code-design
25. **body:** should rename index and j to something more descriptive
    label: code-design
26. **body:** could you either name testInsufficientContainers[1|2|3] more descriptively or clarify in their javadocs what makes them different.
    label: documentation
27. **body:** It seems this pattern of assertion for cpu/ram/disk is repeated in multiple tests. Could these be broken out into descriptive methods like assertCpu(...) that are reused?
    label: code-design
28. @billonahill This is the same as in current round robin packing test. Should I keep it as is or change it in both places?
29. @billonahill Same as round robin. Should I keep it?
30. @billonahill Same as round robin
31. **body:** @billonahill @kramasamy I can remove that but I thought it might be confusing for users to see a larger number of containers than what they declared. That is why I let them clarify the behavior. Do you think it is better to remove that and always adjust if needed?
    label: code-design
32. **body:** @billonahill The container and RamRequirements classes are only used by the packing algorithm. In that sense I don't see them as core components. In the context of packing I'm using them as helper classes that's why I put them under utils. If "utils" is misleading, then how do you think we should name this package?
    label: code-design
33. @billonahill Same as round robin. Should I change in both places?
34. **body:** @billonahill As you may have noticed the equations are not the same across tests. It is hard to abstract them since they depend on the result of the packing algorithm so they are tailored for the specific cases used in these tests.
    label: code-design
35. @billonahill Actually they are both severe cases and they are caused my misconfiguration of the topology parameters. I added a log message in both cases.
36. I think it would be reasonable to remove it and sometimes get more containers. In that way the container setting would also be considered a hint.
37. I think it's fine to put these classes directly under c.t.heron.packing if there common objects leveraged by different packing algos.
38. Got it. I thought they were similar except for the constants. The logic behind the math in the assertions is not entirely obvious so if there's something that could be done to help clarify the equation to the reader that would help.
39. yes please.
40. yes please.
41. If this is due to invalid configs, throwing an exception seems preferred over returning null.
42. **body:** If this exception were thrown the reader would have little info about what went wrong. Instead could getRAMInstances throw a more detailed exception instead of returning null to this method, which then throws an exception? Best to throw as soon as an erroneous condition is found, which in this case is within getRAMInstances.
    label: code-design

43. **body:** Can we replace 7 with `totalInstances`? The more we clearly define the integers users, the easier it is to understand. `int totalInstances = spoutParallelism + boltParallelism` Same below.
    label: code-design
44. Instead of 4 and 3 could we use `spoutParallelism` and `boltParallelism` here (and elsewhere) respectively. Ideally we define ints with a member variable that states what it represents and then we only use the members, not the ints.
45. **body:** This is a bit of a nit, but we divide by 100 for the padding right? In that case whenever we do that could we write this as `padding / 100` to make it more clear why the 100 exists?
    label: code-design
46. 2 here is numContainers right? if so can we replace it?
47. The 4s and 3s here appear to be spoutParallelism and boltParallelism. Is so can we replace them? Same for below and in testContainerRequestedResourcesTwoContainers.
48. They are not spoutParallelism and BoltParallelism. In this test, the first container will contain 2 spouts and 2 bolts (4) and the second one (2 spouts and 1 bolt) so it cannot be replaced.

jira_issues:

jira_issues_comments: