Item 366
git_comments:

1. * 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.
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. * 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. intf = Invoker.createProxy(perfConfiguration.getNextMicroserviceName(), "impl", Intf.class);
5. self: perf-1/perf-a next: perf-2/perf-b
6. * 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.
7. * 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.
8. * 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.
9. can not get cpu usage in windows, so skip this information
10. * 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.
11. * Copyright 2017 Huawei Technologies Co., Ltd * * 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. * 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.

13. * 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.
14. netClient do not like httpClient can not support normal and ssl by the same instance so we do this wrap
15. find again, get the same result
16. * 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.
17. * 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.
18. * 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.
19. * 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.
20. should not throw exception
21. * 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.
22. * 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.
23. metrics
24. redis
25. execute in vertx context
26. ApplicationContext applicationContext = Mockito.mock(ApplicationContext.class);
    BeanUtils.setContext(applicationContext);
    Mockito.when(applicationContext.getBean(ExecutorManager.EXECUTOR_REACTIVE)).thenReturn(new ReactiveExecutor());
27. metrics
28. redis
29. ensure have instance
30. this will make "client.thread-count" bigger than which in microservice.yaml maybe it's better to remove "client.thread-count", just use "rest/highway.thread-count"
31. not in correct context: 1.normal thread 2.vertx worker thread 3.other vertx thread select a existing context
32. reactive mode
33. standard reactive mode
34. execute in vertx context

git_commits:

1. **summary:** [SCB-188] Transport optimize for reactive (#495)
   message: [SCB-188] Transport optimize for reactive (#495) * SCB-188 change var name * SCB-188 when receive response, but do not support response content type, log more information * SCB-188 [WIP] new vertx http client support set ssl flag in request level, so no need to create multiple http client verticles * SCB-188 rest transport changed to use new restClient * SCB-188 do not support multiple clientPool in one net thread, because it's unnecessary, all performance test did not use this mode, and got good result. * SCB-188 add sync flag to invocation * SCB-188 ClientPoolManager optimized for reactive mode * SCB-188 [WIP] make deploy ClientVerticle simpler * SCB-188 [WIP] wrap netClient to be like httpClient, both support normal and ssl connection * SCB-188 [WIP] move ClientPoolFactory logic out from xxxClientVerticle * SCB-188 [WIP] extract buildHttpClientOptions utils * SCB-188 [WIP] config-cc switch to new ClientPoolManager mechanism * SCB-188 [WIP] service-registry switch to new ClientPoolManager mechanism * SCB-188 [WIP] rest client switch to new ClientPoolManager mechanism * SCB-188 highway switch to new ClientPoolManager mechanism * SCB-188 producer operation which return CompletableFuture, if not config operation executor, then use reactive executor * SCB-188 add more useful performance test case * SCB-188 fix tracing-tests bug
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SCB-188] Transport optimize for reactive
   body:

github_pulls_comments:

1. [![Coverage Status](https://coveralls.io/builds/14944452/badge)](https://coveralls.io/builds/14944452) Coverage increased (+0.08%) to 87.544% when pulling **6aa452aa49ba74b31e491351abbf27928ea7bf1a on wujimin:transport-optimize-for-reactive** into **485478edf7a24dc420cee2f95134aefb4aa70420 on apache:master**.
2. [![Coverage Status](https://coveralls.io/builds/14958762/badge)](https://coveralls.io/builds/14958762) Coverage increased (+0.07%) to 87.539% when pulling **c85b8d2bcfbae4d0e44caa0fad46b69f97f77c8f on wujimin:transport-optimize-for-reactive** into **8369fab8767d3983fbeb9b84f9d715cb8e9cade0 on apache:master**.
3. [![Coverage Status](https://coveralls.io/builds/14961816/badge)](https://coveralls.io/builds/14961816) Coverage increased (+0.1%) to 87.588% when pulling **701e8ce085e1705763bb91e7d4d6208ce24cd828 on wujimin:transport-optimize-for-reactive** into **36cd7d605e366cb263f3bb4d6fea0fbede08f1cc on apache:master**.
4. [![Coverage Status](https://coveralls.io/builds/14962165/badge)](https://coveralls.io/builds/14962165) Coverage increased (+0.1%) to 87.517% when pulling **e9d630f563f8f26d1e98c9c00fe0750a4cfaf07e on wujimin:transport-optimize-for-reactive** into **9c4416d69c8ba8db146806ca1ced3876b3a0c411 on apache:master**.

github_pulls_reviews:

1. **body:** documents should modified accordingly
   label: documentation
2. META-INF LICENCE.txt & NOTICE.txt files are missing
3. done
4. done
5. Hi, you don't need add the LICENCE.txt & NOTICE.txt by hand any more. There is a maven plugin which could take care of these license files when building the artifacts.
6. Please add the license header here.
7. License header.
8. Please update the license header.
9. done
10. done
11. deleted
12. done

jira_issues:

1. **summary:** optimize transport for reactive mode
   description:

jira_issues_comments:

1. wujimin opened a new pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

2. coveralls commented on issue #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#issuecomment-355918924 [![Coverage Status](https://coveralls.io/builds/14944452/badge)](https://coveralls.io/builds/14944452) Coverage increased (+0.08%) to 87.544% when pulling **6aa452aa49ba74b31e491351abbf27928ea7bf1a on wujimin:transport-optimize-for-reactive** into **485478edf7a24dc420cee2f95134aefb4aa70420 on apache:master**. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
3. liubao68 commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160124352 ########## File path: transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayConfig.java ########## @@ -42,10 +42,4 @@ public static int getClientThreadCount() { DynamicPropertyFactory.getInstance().getIntProperty("cse.highway.client.thread-count", 1); return address.get(); } - - public static int getClientConnectionPoolPerThread() { - DynamicIntProperty address = - DynamicPropertyFactory.getInstance().getIntProperty("cse.highway.client.connection-pool-per-thread", 1); Review comment: documents should modified accordingly ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
4. liubao68 commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160125490 ########## File path: demo/perf/src/main/resources/microservice.yaml ########## @@ -15,28 +15,47 @@ ## limitations under the License. Review comment: META-INF LICENCE.txt & NOTICE.txt files are missing ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
5. wujimin commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160295319 ########## File path: demo/perf/src/main/resources/microservice.yaml ########## @@ -15,28 +15,47 @@ ## limitations under the License. Review comment: done ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
6. wujimin commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160296199 ########## File path: transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayConfig.java ########## @@ -42,10 +42,4 @@ public static int getClientThreadCount() { DynamicPropertyFactory.getInstance().getIntProperty("cse.highway.client.thread-count", 1); return address.get(); } - - public static int getClientConnectionPoolPerThread() { - DynamicIntProperty address = - DynamicPropertyFactory.getInstance().getIntProperty("cse.highway.client.connection-pool-per-thread", 1); Review comment: done ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
7. WillemJiang commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160298905 ########## File path: demo/perf/src/main/resources/microservice.yaml ########## @@ -15,28 +15,47 @@ ## limitations under the License. Review comment: Hi, you don't need add the LICENCE.txt & NOTICE.txt by hand any more. There is a maven plugin which could take care of these license files when building the artifacts. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
8. coveralls commented on issue #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#issuecomment-356165065 [![Coverage Status](https://coveralls.io/builds/14958762/badge)](https://coveralls.io/builds/14958762) Coverage increased (+0.07%) to 87.539% when pulling **c85b8d2bcfbae4d0e44caa0fad46b69f97f77c8f on wujimin:transport-optimize-for-reactive** into **8369fab8767d3983fbeb9b84f9d715cb8e9cade0 on apache:master**. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
9. WillemJiang commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160329484 ########## File path: demo/perf/pom.xml ########## @@ -0,0 +1,56 @@ +<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Review comment: Please add the license header here. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
10. WillemJiang commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160329588 ########## File path: demo/perf/src/main/java/io/servicecomb/demo/perf/Intf.java ########## @@ -0,0 +1,9 @@ +package io.servicecomb.demo.perf; Review comment: License header. ---------- This is

an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

11. WillemJiang commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160329771 ########## File path: demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMain.java ########## @@ -0,0 +1,48 @@ +/* Review comment: Please update the license header. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
12. wujimin commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160333220 ########## File path: demo/perf/pom.xml ########## @@ -0,0 +1,56 @@ +<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Review comment: done ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
13. wujimin commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160333227 ########## File path: demo/perf/src/main/java/io/servicecomb/demo/perf/Intf.java ########## @@ -0,0 +1,9 @@ +package io.servicecomb.demo.perf; Review comment: done ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
14. wujimin commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160333258 ########## File path: demo/perf/src/main/resources/microservice.yaml ########## @@ -15,28 +15,47 @@ ## limitations under the License. Review comment: deleted ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
15. coveralls commented on issue #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#issuecomment-356206948 [![Coverage Status](https://coveralls.io/builds/14961816/badge)](https://coveralls.io/builds/14961816) Coverage increased (+0.1%) to 87.588% when pulling **701e8ce085e1705763bb91e7d4d6208ce24cd828 on wujimin:transport-optimize-for-reactive** into **36cd7d605e366cb263f3bb4d6fea0fbede08f1cc on apache:master**. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
16. wujimin commented on a change in pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#discussion_r160340407 ########## File path: demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMain.java ########## @@ -0,0 +1,48 @@ +/* Review comment: done ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
17. coveralls commented on issue #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495#issuecomment-356215931 [![Coverage Status](https://coveralls.io/builds/14962165/badge)](https://coveralls.io/builds/14962165) Coverage increased (+0.1%) to 87.517% when pulling **e9d630f563f8f26d1e98c9c00fe0750a4cfaf07e on wujimin:transport-optimize-for-reactive** into **9c4416d69c8ba8db146806ca1ced3876b3a0c411 on apache:master**. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
18. WillemJiang closed pull request #495: [SCB-188] Transport optimize for reactive URL: https://github.com/apache/incubator-servicecomb-java-chassis/pull/495 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/core/pom.xml b/core/pom.xml index ceee759e1..2899b4ab9 100644 --- a/core/pom.xml +++ b/core/pom.xml @@ -67,5 +67,9 @@ <artifactId>log4j</artifactId> <scope>test</scope> </dependency> + <dependency> + <groupId>io.servicecomb</groupId> + <artifactId>foundation-test-scaffolding</artifactId> + </dependency> </dependencies> </project> diff --git a/core/src/main/java/io/servicecomb/core/Invocation.java b/core/src/main/java/io/servicecomb/core/Invocation.java index 40a79d148..9d867a397 100644 --- a/core/src/main/java/io/servicecomb/core/Invocation.java +++ b/core/src/main/java/io/servicecomb/core/Invocation.java @@ -73,6 +73,8 @@ public void setMetricsData(Object metricsData) { private long startProcessingTime; + private boolean sync = true; + public void setStartTime(long startTime) { this.startTime = startTime; } @@ -207,4 +209,12 @@ public void triggerFinishedEvent() { this.invocationType, finishedTime - startProcessingTime, finishedTime - startTime)); } + + public boolean isSync() { + return sync; + } + + public void setSync(boolean sync) { + this.sync = sync; + } } diff --git a/core/src/main/java/io/servicecomb/core/definition/schema/ProducerSchemaFactory.java b/core/src/main/java/io/servicecomb/core/definition/schema/ProducerSchemaFactory.java index 0d879dd75..52f5d5759 100644 --- a/core/src/main/java/io/servicecomb/core/definition/schema/ProducerSchemaFactory.java +++ b/core/src/main/java/io/servicecomb/core/definition/schema/ProducerSchemaFactory.java @@ -18,6 +18,8 @@ package io.servicecomb.core.definition.schema; import java.util.Set; +import java.util.concurrent.CompletableFuture; +import java.util.concurrent.Executor; import javax.inject.Inject; @@ -32,6 +34,8 @@ import

io.servicecomb.core.definition.MicroserviceMeta; import io.servicecomb.core.definition.OperationMeta; import io.servicecomb.core.definition.SchemaMeta; +import io.servicecomb.core.executor.ExecutorManager; +import io.servicecomb.foundation.common.utils.BeanUtils; import io.servicecomb.serviceregistry.RegistryUtils; import io.servicecomb.swagger.engine.SwaggerEnvironment; import io.servicecomb.swagger.engine.SwaggerProducer; @@ -76,9 +80,14 @@ public SchemaMeta getOrCreateProducerSchema(String microserviceName, String sche SchemaMeta schemaMeta = getOrCreateSchema(context); SwaggerProducer producer = swaggerEnv.createProducer(producerInstance, schemaMeta.getSwagger()); + Executor reactiveExecutor = BeanUtils.getBean(ExecutorManager.EXECUTOR_REACTIVE); for (OperationMeta operationMeta : schemaMeta.getOperations()) { SwaggerProducerOperation producerOperation = producer.findOperation(operationMeta.getOperationId()); operationMeta.putExtData(Const.PRODUCER_OPERATION, producerOperation); + + if (CompletableFuture.class.equals(producerOperation.getProducerMethod().getReturnType())) { + operationMeta.setExecutor(ExecutorManager.findExecutor(operationMeta, reactiveExecutor)); + } } return schemaMeta; diff --git a/core/src/main/java/io/servicecomb/core/executor/ExecutorManager.java b/core/src/main/java/io/servicecomb/core/executor/ExecutorManager.java index 121743ccf..096b19df8 100644 --- a/core/src/main/java/io/servicecomb/core/executor/ExecutorManager.java +++ b/core/src/main/java/io/servicecomb/core/executor/ExecutorManager.java @@ -25,6 +25,8 @@ import io.servicecomb.foundation.common.utils.BeanUtils; public final class ExecutorManager { + public static final String KEY_EXECUTORS_PREFIX = "cse.executors.Provider."; + public static final String KEY_EXECUTORS_DEFAULT = "cse.executors.default"; public static final String EXECUTOR_GROUP_THREADPOOL = "cse.executor.groupThreadPool"; @@ -38,13 +40,21 @@ private ExecutorManager() { // 只会在初始化时执行，一点点重复的查找，没必要做缓存 public static Executor findExecutor(OperationMeta operationMeta) { - Executor executor = findByKey("cse.executors.Provider." + operationMeta.getSchemaQualifiedName()); + return findExecutor(operationMeta, null); + } + + public static Executor findExecutor(OperationMeta operationMeta, Executor defaultOperationExecutor) { + Executor executor = findByKey(KEY_EXECUTORS_PREFIX + operationMeta.getSchemaQualifiedName()); if (executor != null) { return executor; } + if (defaultOperationExecutor != null) { + return defaultOperationExecutor; + } + // 尝试schema级别 - executor = findByKey("cse.executors.Provider." + operationMeta.getSchemaMeta().getName()); + executor = findByKey(KEY_EXECUTORS_PREFIX + operationMeta.getSchemaMeta().getName()); if (executor != null) { return executor; } diff --git a/core/src/main/java/io/servicecomb/core/provider/consumer/InvokerUtils.java b/core/src/main/java/io/servicecomb/core/provider/consumer/InvokerUtils.java index a7662bbd9..4da592bee 100644 --- a/core/src/main/java/io/servicecomb/core/provider/consumer/InvokerUtils.java +++ b/core/src/main/java/io/servicecomb/core/provider/consumer/InvokerUtils.java @@ -84,6 +84,8 @@ public static Response innerSyncInvoke(Invocation invocation) { public static void reactiveInvoke(Invocation invocation, AsyncResponse asyncResp) { try { triggerStartedEvent(invocation); + invocation.setSync(false); + ReactiveResponseExecutor respExecutor = new ReactiveResponseExecutor(); invocation.setResponseExecutor(respExecutor); diff --git a/core/src/test/java/io/servicecomb/core/definition/schema/TestProducerSchemaFactory.java b/core/src/test/java/io/servicecomb/core/definition/schema/TestProducerSchemaFactory.java index 7a4a681bc..01b422421 100644 --- a/core/src/test/java/io/servicecomb/core/definition/schema/TestProducerSchemaFactory.java +++ b/core/src/test/java/io/servicecomb/core/definition/schema/TestProducerSchemaFactory.java @@ -16,14 +16,16 @@ */ package io.servicecomb.core.definition.schema; +import java.util.concurrent.CompletableFuture; +import java.util.concurrent.Executor; import javax.xml.ws.Holder; +import org.hamcrest.Matchers; import org.junit.AfterClass; import org.junit.Assert; import org.junit.BeforeClass; import org.junit.Test; -import org.mockito.Mockito; -import org.springframework.context.ApplicationContext; import io.servicecomb.core.Const; import io.servicecomb.core.Endpoint; @@ -32,6 +34,8 @@ import io.servicecomb.core.definition.OperationMeta; import io.servicecomb.core.definition.SchemaMeta; import io.servicecomb.core.definition.loader.SchemaLoader; +import io.servicecomb.core.executor.ExecutorManager; +import io.servicecomb.core.executor.ReactiveExecutor; import io.servicecomb.core.unittest.UnitTestMeta; import io.servicecomb.foundation.common.utils.BeanUtils; import io.servicecomb.foundation.common.utils.ReflectUtils; @@ -47,6 +51,8 @@ import io.servicecomb.swagger.invocation.converter.ConverterMgr; import io.servicecomb.swagger.invocation.exception.CommonExceptionData; import io.servicecomb.swagger.invocation.exception.InvocationException; +import mockit.Mock; +import mockit.MockUp; public class TestProducerSchemaFactory { private static SwaggerEnvironment swaggerEnv = new BootstrapNormal().boot(); @@ -59,6 +65,10 @@ public int add(int x, int y) { return x + y; } + + public CompletableFuture<String> async() { + return null; + } } @BeforeClass @@ -85,7 +95,24 @@ public void putSelfBasePathIfAbsent(String microserviceName, String basePath) { "compositeSwaggerGeneratorContext", compositeSwaggerGeneratorContext); - BeanUtils.setContext(Mockito.mock(ApplicationContext.class)); + Executor reactiveExecutor = new ReactiveExecutor(); + Executor normalExecutor = (cmd) -> { + }; + new MockUp<BeanUtils>() { + @SuppressWarnings("unchecked") + @Mock + <T> T getBean(String name) { + if (ExecutorManager.EXECUTOR_REACTIVE.equals(name)) { + return (T) reactiveExecutor; + } + + return (T) normalExecutor; + } + }; + + // ApplicationContext applicationContext = Mockito.mock(ApplicationContext.class); + // BeanUtils.setContext(applicationContext); + // Mockito.when(applicationContext.getBean(ExecutorManager.EXECUTOR_REACTIVE)).thenReturn(new ReactiveExecutor()); UnitTestMeta.init(); @@ -131,4 +158,10 @@ public String getInvocationQualifiedName() { CommonExceptionData data = (CommonExceptionData) exception.getErrorData(); Assert.assertEquals("Cse Internal Server Error", data.getMessage()); } + + @Test + public void testCompletableFuture() { + OperationMeta operationMeta = schemaMeta.ensureFindOperation("async"); + Assert.assertThat(operationMeta.getExecutor(), Matchers.instanceOf(ReactiveExecutor.class)); + } } diff --git a/core/src/test/java/io/servicecomb/core/executor/TestExecutorManager.java b/core/src/test/java/io/servicecomb/core/executor/TestExecutorManager.java new file mode 100644 index

000000000..0f8bbb57b --- /dev/null +++ b/core/src/test/java/io/servicecomb/core/executor/TestExecutorManager.java @@ -0,0 +1,136 @@ +/* + * 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. + */ +package io.servicecomb.core.executor; + +import java.util.concurrent.Executor; + +import org.junit.After; import org.junit.Assert; import org.junit.Before; import org.junit.Test; + +import io.servicecomb.core.definition.OperationMeta; import io.servicecomb.core.definition.SchemaMeta; import io.servicecomb.foundation.common.utils.BeanUtils; import io.servicecomb.foundation.test.scaffolding.config.ArchaiusUtils; import mockit.Expectations; import mockit.Mocked; + +public class TestExecutorManager { + @Mocked + Executor defaultExecutor; + + @Before + public void setup() { + ArchaiusUtils.resetConfig(); + } + + @After + public void teardown() { + ArchaiusUtils.resetConfig(); + } + + @Test + public void findExecutor_oneParam(@Mocked Executor executor, @Mocked OperationMeta operationMeta) { + new Expectations(BeanUtils.class) { + { + BeanUtils.getBean(ExecutorManager.EXECUTOR_DEFAULT); + result = executor; + } + }; + + Assert.assertSame(executor, ExecutorManager.findExecutor(operationMeta)); + } + + @Test + public void findExecutor_twoParam_opCfg_withoutOpDef(@Mocked Executor executor, + @Mocked OperationMeta operationMeta) { + String schemaQualifiedName = "schemaId.opId"; + String opBeanId = "opBeanId"; + ArchaiusUtils.setProperty(ExecutorManager.KEY_EXECUTORS_PREFIX + schemaQualifiedName, opBeanId); + new Expectations(BeanUtils.class) { + { + operationMeta.getSchemaQualifiedName(); + result = schemaQualifiedName; + BeanUtils.getBean(opBeanId); + result = executor; + } + }; + + Assert.assertSame(executor, ExecutorManager.findExecutor(operationMeta, null)); + } + + @Test + public void findExecutor_twoParam_opCfg_withOpDef(@Mocked Executor executor, + @Mocked Executor defExecutor, + @Mocked OperationMeta operationMeta) { + String schemaQualifiedName = "schemaId.opId"; + String opBeanId = "opBeanId"; + ArchaiusUtils.setProperty(ExecutorManager.KEY_EXECUTORS_PREFIX + schemaQualifiedName, opBeanId); + new Expectations(BeanUtils.class) { + { + operationMeta.getSchemaQualifiedName(); + result = schemaQualifiedName; + BeanUtils.getBean(opBeanId); + result = executor; + } + }; + + Assert.assertSame(executor, ExecutorManager.findExecutor(operationMeta, defExecutor)); + } + + @Test + public void findExecutor_twoParam_schemaCfg_withOpDef(@Mocked OperationMeta operationMeta, + @Mocked Executor defExecutor) { + Assert.assertSame(defExecutor, ExecutorManager.findExecutor(operationMeta, defExecutor)); + } + + @Test + public void findExecutor_twoParam_schemaCfg_withoutOpDef(@Mocked Executor executor, + @Mocked SchemaMeta schemaMeta, + @Mocked OperationMeta operationMeta) { + String schemaName = "schemaId"; + String opBeanId = "opBeanId"; + ArchaiusUtils.setProperty(ExecutorManager.KEY_EXECUTORS_PREFIX + schemaName, opBeanId); + new Expectations(BeanUtils.class) { + { + schemaMeta.getName(); + result = schemaName; + BeanUtils.getBean(opBeanId); + result = executor; + } + }; + + Assert.assertSame(executor, ExecutorManager.findExecutor(operationMeta, null)); + } + + @Test + public void findExecutor_twoParam_defaultCfg(@Mocked Executor executor, + @Mocked SchemaMeta schemaMeta, + @Mocked OperationMeta operationMeta) { + String beanId = "beanId"; + ArchaiusUtils.setProperty(ExecutorManager.KEY_EXECUTORS_DEFAULT, beanId); + new Expectations(BeanUtils.class) { + { + BeanUtils.getBean(beanId); + result = executor; + } + }; + + Assert.assertSame(executor, ExecutorManager.findExecutor(operationMeta, null)); + } +} diff --git a/demo/demo-perf-client/pom.xml b/demo/demo-perf-client/pom.xml deleted file mode 100644 index 753ccb959..000000000 --- a/demo/demo-perf-client/pom.xml +++ /dev/null @@ -1,45 +0,0 @@ -<!-- - ~ 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. - --> - -<project xmlns="http://maven.apache.org/POM/4.0.0" - xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" - xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> - <modelVersion>4.0.0</modelVersion> - <parent> - <groupId>io.servicecomb.demo</groupId> - <artifactId>demo-parent</artifactId> - <version>0.6.0-SNAPSHOT</version> - </parent> - <artifactId>demo-perf-client</artifactId> - <dependencies> - <dependency> - <groupId>io.servicecomb.demo</groupId> - <artifactId>pojo-client</artifactId> - </dependency> - </dependencies> - - <properties> - <demo.main>io.servicecomb.demo.client.perf.PerfClient</demo.main> - </properties> - <build> - <plugins> - <plugin> - <groupId>org.apache.maven.plugins</groupId> - <artifactId>maven-dependency-plugin</artifactId> - </plugin> - </plugins> - </build> -</project> diff --git a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/ClientThread.java b/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/ClientThread.java deleted file mode 100644 index cd0e95bc1..000000000 --- a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/ClientThread.java +++ /dev/null @@ -1,53 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.demo.client.perf; - -import io.servicecomb.demo.pojo.client.PojoClient; -import io.servicecomb.demo.server.Test; -import io.servicecomb.demo.server.TestRequest; -import io.servicecomb.demo.server.User; -import io.servicecomb.foundation.common.CommonThread; - -public class ClientThread extends CommonThread { - @Override - public void run() { - Test test = PojoClient.test; - - while (isRunning()) { - int idx = 0; - for (;;) { - User user = new User(); - - TestRequest request = new TestRequest(); - request.setUser(user); - request.setIndex(idx); - request.setData(PojoClient.buffer); - - try { - User result = test.wrapParam(request); - - if (result.getIndex() != idx) { - System.out.printf("error result:%s, expect idx %d\n", result, idx); - } - } catch (Throwable e) { - // e.printStackTrace(); - } - } - } - } -} diff --git a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/ClientVerticle.java b/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/ClientVerticle.java deleted file mode 100644 index c477bcf03..000000000 --- a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/ClientVerticle.java +++ /dev/null @@ -1,75 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.demo.client.perf; - -import io.servicecomb.core.CseContext; -import io.servicecomb.core.Invocation; -import io.servicecomb.core.definition.SchemaMeta; -import io.servicecomb.core.invocation.InvocationFactory; -import io.servicecomb.core.provider.consumer.InvokerUtils; -import io.servicecomb.core.provider.consumer.ReferenceConfig; -import io.servicecomb.demo.pojo.client.PojoClient; -import io.servicecomb.demo.server.Test; -import io.servicecomb.demo.server.TestRequest; -import io.servicecomb.demo.server.User; -import io.servicecomb.swagger.invocation.exception.CommonExceptionData; -import io.servicecomb.swagger.invocation.exception.InvocationException; -import io.vertx.core.AbstractVerticle; - -public class ClientVerticle extends AbstractVerticle { - - Test test = PojoClient.test; - - ReferenceConfig config = - CseContext.getInstance().getConsumerProviderManager().setTransport("pojo", Config.getTransport()); - - int idx = 0; - - @Override - public void start() throws Exception { - vertx.setTimer(100, this::send); - } - - protected void send(Long event) { - User user = new User(); - - TestRequest request = new TestRequest(); - request.setUser(user); - request.setIndex(idx); - request.setData(PojoClient.buffer); - - SchemaMeta schemaMeta = - config.getMicroserviceMeta().ensureFindSchemaMeta("server"); - Object[] args = new Object[] {request}; - Invocation invocation = InvocationFactory.forConsumer(config, schemaMeta, "wrapParam", args); - InvokerUtils.reactiveInvoke(invocation, ar -> { - if (ar.isSuccessed()) { - User result = ar.getResult(); - if (result.getIndex() != idx) { - System.out.printf("error result:%s, expect idx %d\n", result, idx); - } - } else { - CommonExceptionData data = - (CommonExceptionData) ((InvocationException) ar.getResult()).getErrorData(); - System.out.println(data.getMessage()); - } - - send(null); - }); - } -} diff --git a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/Config.java b/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/Config.java deleted file mode 100644 index 568ca6bcb..000000000 --- a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/Config.java +++ /dev/null @@ -1,50 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.demo.client.perf; - -public class Config { - private static int clientThread; - - private static String transport; - - private static String mode; - - public static String getTransport() { - return transport; - } - - public static void setTransport(String transport) { - Config.transport = transport; - } - - public static int getClientThread() { - return clientThread; - } - - public static void setClientThread(int clientThread) { - Config.clientThread = clientThread; - } - - public static String getMode() { - return mode; - } - - public static void setMode(String mode) { - Config.mode = mode; - } -} diff --git a/demo/demo-perf-client/src/main/resources/META-INF/LICENSE.txt b/demo/demo-perf-client/src/main/resources/META-INF/LICENSE.txt deleted file mode 100644 index d64569567..000000000 --- a/demo/demo-perf-client/src/main/resources/META-INF/LICENSE.txt +++ /dev/null @@ -1,202 +0,0 @@ - Apache License - Version 2.0, January 2004 - http://www.apache.org/licenses/ - - TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION - - 1. Definitions. - - "License" shall mean the terms and conditions for use, reproduction, - and distribution as defined by Sections 1 through 9 of this document. - - "Licensor" shall mean the copyright owner or entity authorized by - the copyright owner that is granting the License. - - "Legal Entity" shall mean the union of the acting entity and all - other entities that control, are controlled by, or are under common - control with that entity. For the purposes of this definition, - "control" means (i) the power, direct or indirect, to cause the - direction or management of such entity, whether by contract or - otherwise, or (ii) ownership of fifty percent (50%) or more of the - outstanding shares, or (iii) beneficial ownership of such entity. - - "You" (or "Your") shall mean an individual or Legal Entity - exercising permissions granted by this License. - - "Source" form shall mean the preferred form for making modifications, - including but not limited to software source code, documentation - source, and configuration files. - - "Object" form shall mean any form resulting from mechanical - transformation or translation of a Source form, including but - not limited to compiled object code, generated documentation, - and conversions to other media types. - - "Work" shall mean the work of authorship, whether in Source or - Object form, made available under the License, as indicated by a - copyright notice that is included in or attached to the work - (an example is provided in the Appendix below). - - "Derivative Works" shall mean any work, whether in Source or Object - form, that is based on (or derived from) the Work and for which the - editorial revisions,

annotations, elaborations, or other modifications - represent, as a whole, an original work of authorship. For the purposes - of this License, Derivative Works shall not include works that remain - separable from, or merely link (or bind by name) to the interfaces of, - the Work and Derivative Works thereof. - - "Contribution" shall mean any work of authorship, including - the original version of the Work and any modifications or additions - to that Work or Derivative Works thereof, that is intentionally - submitted to Licensor for inclusion in the Work by the copyright owner - or by an individual or Legal Entity authorized to submit on behalf of - the copyright owner. For the purposes of this definition, "submitted" - means any form of electronic, verbal, or written communication sent - to the Licensor or its representatives, including but not limited to - communication on electronic mailing lists, source code control systems, - and issue tracking systems that are managed by, or on behalf of, the - Licensor for the purpose of discussing and improving the Work, but - excluding communication that is conspicuously marked or otherwise - designated in writing by the copyright owner as "Not a Contribution." - - "Contributor" shall mean Licensor and any individual or Legal Entity - on behalf of whom a Contribution has been received by Licensor and - subsequently incorporated within the Work. - - 2. Grant of Copyright License. Subject to the terms and conditions of - this License, each Contributor hereby grants to You a perpetual, - worldwide, non-exclusive, no-charge, royalty-free, irrevocable - copyright license to reproduce, prepare Derivative Works of, - publicly display, publicly perform, sublicense, and distribute the - Work and such Derivative Works in Source or Object form. - - 3. Grant of Patent License. Subject to the terms and conditions of - this License, each Contributor hereby grants to You a perpetual, - worldwide, non-exclusive, no-charge, royalty-free, irrevocable - (except as stated in this section) patent license to make, have made, - use, offer to sell, sell, import, and otherwise transfer the Work, - where such license applies only to those patent claims licensable - by such Contributor that are necessarily infringed by their - Contribution(s) alone or by combination of their Contribution(s) - with the Work to which such Contribution(s) was submitted. If You - institute patent litigation against any entity (including a - cross-claim or counterclaim in a lawsuit) alleging that the Work - or a Contribution incorporated within the Work constitutes direct - or contributory patent infringement, then any patent licenses - granted to You under this License for that Work shall terminate - as of the date such litigation is filed. - - 4. Redistribution. You may reproduce and distribute copies of the - Work or Derivative Works thereof in any medium, with or without - modifications, and in Source or Object form, provided that You - meet the following conditions: - - (a) You must give any other recipients of the Work or - Derivative Works a copy of this License; and - - (b) You must cause any modified files to carry prominent notices - stating that You changed the files; and - - (c) You must retain, in the Source form of any Derivative Works - that You distribute, all copyright, patent, trademark, and - attribution notices from the Source form of the Work, - excluding those notices that do not pertain to any part of - the Derivative Works; and - - (d) If the Work includes a "NOTICE" text file as part of its - distribution, then any Derivative Works that You distribute must - include a readable copy of the attribution notices contained - within such NOTICE file, excluding those notices that do not - pertain to any part of the Derivative Works, in at least one - of the following places: within a NOTICE text file distributed - as part of the Derivative Works; within the Source form or - documentation, if provided along with the Derivative Works; or, - within a display generated by the Derivative Works, if and - wherever such third-party notices normally appear. The contents - of the NOTICE file are for informational purposes only and - do not modify the License. You may add Your own attribution - notices within Derivative Works that You distribute, alongside - or as an addendum to the NOTICE text from the Work, provided - that such additional attribution notices cannot be construed - as modifying the License. - - You may add Your own copyright statement to Your modifications and - may provide additional or different license terms and conditions - for use, reproduction, or distribution of Your modifications, or - for any such Derivative Works as a whole, provided Your use, - reproduction, and distribution of the Work otherwise complies with - the conditions stated in this License. - - 5. Submission of Contributions. Unless You explicitly state otherwise, - any Contribution intentionally submitted for inclusion in the Work - by You to the Licensor shall be under the terms and conditions of - this License, without any additional terms or conditions. - Notwithstanding the above, nothing herein shall supersede or modify - the terms of any separate license agreement you may have executed - with Licensor regarding such Contributions. - - 6. Trademarks. This License does not grant permission to use the trade - names, trademarks, service marks, or product names of the Licensor, - except as required for reasonable and customary use in describing the - origin of the Work and reproducing the content of the NOTICE file. - - 7. Disclaimer of Warranty. Unless required by applicable law or - agreed to in writing, Licensor provides the Work (and each - Contributor provides its Contributions) on an "AS IS" BASIS, - WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or - implied, including, without limitation, any warranties or conditions - of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A - PARTICULAR PURPOSE. You are solely responsible for determining the - appropriateness of using or redistributing the Work and assume any - risks associated with Your exercise of permissions under this License. - - 8. Limitation of Liability. In no event and under no legal theory, - whether in tort (including negligence), contract, or otherwise, - unless required by applicable law (such as deliberate and grossly - negligent acts) or agreed to in writing, shall any Contributor be - liable to You for damages, including any direct, indirect, special, - incidental, or consequential damages of any character arising as a - result of this License or out of the use or inability to use the - Work (including but not limited to damages for loss of goodwill, - work stoppage, computer failure or malfunction, or any and all - other commercial damages or losses), even if such Contributor - has been advised of the possibility of such damages. - - 9. Accepting Warranty or Additional Liability. While redistributing - the Work or Derivative Works thereof, You may choose to offer, - and charge a fee for, acceptance of support, warranty, indemnity, - or other liability obligations and/or rights consistent with this - License. However, in accepting such obligations, You may act only - on Your own behalf and on Your sole responsibility, not on behalf - of any other Contributor, and only if You agree to indemnify, - defend, and hold each Contributor harmless for any liability - incurred by, or claims asserted against, such Contributor by reason - of your accepting any such warranty or additional liability. - - END OF TERMS AND CONDITIONS - - APPENDIX: How to apply the Apache License to your work. - - To apply the Apache License to your work, attach the following - boilerplate notice, with the fields enclosed by brackets "[]" - replaced with your own identifying information. (Don't include - the brackets!) The text should be enclosed in the appropriate - comment syntax for the file format. We also recommend that a - file or class name and description of purpose be included on the - same "printed page" as the copyright notice for easier - identification within third-party archives. - - Copyright [yyyy] [name of copyright owner] - - 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. diff --git a/demo/demo-perf-client/src/main/resources/META-INF/NOTICE.txt b/demo/demo-perf-client/src/main/resources/META-INF/NOTICE.txt deleted file mode 100644 index 2e215bf2e..000000000 --- a/demo/demo-perf-client/src/main/resources/META-INF/NOTICE.txt +++ /dev/null @@ -1,11 +0,0 @@ - === - == NOTICE file corresponding to the section 4 d of == - == the Apache License, Version 2.0, == - == in this case for the Apache Camel distribution. == - === - - This product includes software developed by - The Apache Software Foundation (http://www.apache.org/). - - Please read the different LICENSE files present in the licenses directory of - this distribution. diff --git a/demo/demo-perf-client/src/main/resources/config/cse.demo.perf.properties b/demo/demo-perf-client/src/main/resources/config/cse.demo.perf.properties deleted file mode 100644 index 285380a51..000000000 --- a/demo/demo-perf-client/src/main/resources/config/cse.demo.perf.properties +++ /dev/null @@ -1,21 +0,0 @@ -# -# 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. -# - -cse.demo.client.thread=100 -cse.demo.client.transport=highway -cse.demo.client.mode=sync -#cse.demo.client.mode=reactive diff --git a/demo/perf/pom.xml b/demo/perf/pom.xml new file mode 100644 index 000000000..1a7c51f08 --- /dev/null +++ b/demo/perf/pom.xml @@ -0,0 +1,74 @@ +<?xml version="1.0" encoding="UTF-8"?> +<!-- + ~ 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. + --> + +<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" + xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> + <modelVersion>4.0.0</modelVersion> + <parent> + <groupId>io.servicecomb.demo</groupId> + <artifactId>demo-parent</artifactId> + <version>0.6.0-SNAPSHOT</version> + </parent> + <artifactId>perf</artifactId> + + <dependencies> + <dependency> + <groupId>io.servicecomb</groupId> + <artifactId>provider-pojo</artifactId> + </dependency> + <dependency> + <groupId>io.servicecomb</groupId> + <artifactId>provider-springmvc</artifactId> + </dependency> + <dependency> + <groupId>io.servicecomb</groupId> + <artifactId>transport-highway</artifactId> + </dependency> + <dependency> + <groupId>io.servicecomb</groupId> + <artifactId>transport-rest-vertx</artifactId> + </dependency> + + <dependency> + <groupId>ch.qos.logback</groupId> + <artifactId>logback-classic</artifactId> + </dependency> + + <dependency> + <groupId>io.vertx</groupId> + <artifactId>vertx-redis-client</artifactId> + <version>3.5.0</version> + </dependency> + <dependency> + <groupId>io.servicecomb</groupId> + <artifactId>metrics-core</artifactId> + </dependency> + </dependencies> + + <properties> + <demo.main>io.servicecomb.demo.perf.PerfMain</demo.main> + </properties> + <build> + <plugins> + <plugin> + <groupId>org.apache.maven.plugins</groupId> + <artifactId>maven-dependency-plugin</artifactId> + </plugin> + </plugins> + </build> +</project> \ No newline at end of file diff --git a/demo/perf/readme.MD b/demo/perf/readme.MD new file mode 100644 index 000000000..00062abf5 --- /dev/null +++ b/demo/perf/readme.MD @@ -0,0 +1,45 @@ +suppose jar named perf.jar +1.copy perf.jar to different directory + +2.create microservice.yaml in each directory +# change microserviceName to be perf1/perf2/perf3, and so on +service_description: + name: perf1 +cse: + references: + # use which transport to invoke + transport: rest + +# sync mode consumer count +sync-count: 10 +# async mode consumer count +async-count: 20 +# use sync mode or not +# sync: /v1/syncQuery/{id}?step={step}&all={all}&fromDB={fromDB} +# async:/v1/asyncQuery/{id}?step={step}&all={all}&fromDB={fromDB} +sync: false +# producer microserviceName +producer: perf1 +id: 1 +# every producer determine: +# 1)if step equals all, then this is final step, direct return or query from db +# 2)otherwise inc step and invoke next microservice +# 3)if self name if perf1, then next microservice is perf2 +step: 1 +all: 1 +fromDB: false +response-size: 1 + +# redis parameter +redis: + client: + count: 8 + host: + port: + password: + +3.start producers +java -jar perf.jar + +4.start consumer +java -jar perf.jar -c \ No newline at end of file diff --git a/demo/perf/src/main/java/io/servicecomb/demo/perf/Impl.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/Impl.java new file mode 100644 index 000000000..a609fd405 --- /dev/null +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/Impl.java @@ -0,0 +1,74 @@ +/* + * 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. + */ +package io.servicecomb.demo.perf; + +import java.util.concurrent.CompletableFuture; + +import org.springframework.web.bind.annotation.GetMapping; +import org.springframework.web.bind.annotation.PathVariable; +import org.springframework.web.bind.annotation.RequestMapping; +import

org.springframework.web.bind.annotation.RequestParam; + +import io.servicecomb.provider.rest.common.RestSchema; + +@RestSchema(schemaId = "impl") +@RequestMapping(path = "/v1") +public class Impl { + private Intf intf; + + public Impl() { + // intf = Invoker.createProxy(perfConfiguration.getNextMicroserviceName(), + // "impl", + // Intf.class); + } + + @GetMapping(path = "/syncQuery/{id}") + public String syncQuery(@PathVariable(name = "id") String id, + @RequestParam(name = "step") int step, @RequestParam(name = "all") int all, + @RequestParam(name = "fromDB") boolean fromDB) { + if (step == all) { + if (fromDB) { + return RedisClientUtils.syncQuery(id); + } + + return new StringBuilder(64 + PerfConfiguration.responseData.length()) + .append(id) + .append(" from memory: ") + .append(PerfConfiguration.responseData) + .toString(); + } + + return intf.syncQuery(id, step, all, fromDB); + } + + @GetMapping(path = "/asyncQuery/{id}") + public CompletableFuture<String> asyncQuery(@PathVariable(name = "id") String id, + @RequestParam(name = "step") int step, @RequestParam(name = "all") int all, + @RequestParam(name = "fromDB") boolean fromDB) { + if (step == all) { + if (fromDB) { + return RedisClientUtils.asyncQuery(id); + } + + CompletableFuture<String> future = new CompletableFuture<>(); + future.complete("value of " + id + " from memory."); + return future; + } + + return intf.asyncQuery(id, step, all, fromDB); + } +} diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientVerticle.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/Intf.java similarity index 72% rename from foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientVerticle.java rename to demo/perf/src/main/java/io/servicecomb/demo/perf/Intf.java index fd6c61248..07f064cb5 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientVerticle.java +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/Intf.java @@ -14,12 +14,13 @@ * See the License for the specific language governing permissions and * limitations under the License. */ -package io.servicecomb.foundation.vertx.client.tcp; -public class TcpClientVerticle extends AbstractTcpClientVerticle<TcpClientConnection, TcpClientConnectionPool> { +package io.servicecomb.demo.perf; - @Override - public TcpClientConnectionPool createClientPool() { - return new TcpClientConnectionPool(clientConfig, context, netClient); - } +import java.util.concurrent.CompletableFuture; + +public interface Intf { + String syncQuery(String id, int step, int all, boolean fromDB); + + CompletableFuture<String> asyncQuery(String id, int step, int all, boolean fromDB); } diff --git a/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfConfiguration.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfConfiguration.java new file mode 100644 index 000000000..bb1ba9245 --- /dev/null +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfConfiguration.java @@ -0,0 +1,134 @@ +/* + * 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. + */ +package io.servicecomb.demo.perf; + +import org.springframework.beans.factory.annotation.Value; +import org.springframework.stereotype.Component; + +import com.google.common.base.Strings; + +@Component +public class PerfConfiguration { + public static String selfMicroserviceName; + + public static String nextMicroserviceName; + + public static int syncCount; + + public static int asyncCount; + + public static boolean sync; + + public static String producer; + + public static String id; + + public static int step; + + public static int all; + + public static boolean fromDB; + + public static int responseSize; + + public static String responseData; + + public static int redisClientCount; + + public static String redisHost; + + public static int redisPort; + + public static String redisPassword; + + @Value(value = "${service_description.name}") + public void setSelfMicroserviceName(String selfMicroserviceName) { + PerfConfiguration.selfMicroserviceName = selfMicroserviceName; + + // self: perf-1/perf-a + // next: perf-2/perf-b + char last = selfMicroserviceName.charAt(selfMicroserviceName.length() - 1); + nextMicroserviceName = selfMicroserviceName.substring(0, selfMicroserviceName.length() - 1) + (char) (last + 1); + } + + @Value(value = "${response-size}") + public void setResponseSize(int responseSize) { + PerfConfiguration.responseSize = responseSize; + PerfConfiguration.responseData = Strings.repeat("a", responseSize); + } + + @Value(value = "${sync-count}") + public void setSyncCount(int syncCount) { + PerfConfiguration.syncCount = syncCount; + } + + @Value(value = "${async-count}") + public void setAsyncCount(int asyncCount) { + PerfConfiguration.asyncCount = asyncCount; + } + + @Value(value = "${sync}") + public void setSync(boolean sync) { + PerfConfiguration.sync = sync; + } + + @Value(value = "${producer}") + public void setProducer(String producer) { + PerfConfiguration.producer = producer; + } + + @Value(value = "${id}") + public void setId(String id) { + PerfConfiguration.id = id; + } + + @Value(value = "${step}") + public void setStep(int step) { + PerfConfiguration.step = step; + } + + @Value(value = "${all}") + public void setAll(int all) { + PerfConfiguration.all = all; + } + + @Value(value = "${fromDB}") + public void setFromDB(boolean fromDB) { + PerfConfiguration.fromDB = fromDB; + } + + @Value(value = "${redis.client.count}") + public void setRedisClientCount(int redisClientCount) { + PerfConfiguration.redisClientCount = redisClientCount; + } + + @Value(value = "${redis.host}") + public void setRedisHost(String redisHost) { + PerfConfiguration.redisHost = redisHost; + } + + @Value(value = "${redis.port}") + public void setRedisPort(int redisPort) { + PerfConfiguration.redisPort = redisPort; + } + + @Value(value = "${redis.password:null}") + public void setRedisPassword(String redisPassword) { + PerfConfiguration.redisPassword = redisPassword; + } +} diff --git a/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfConsumer.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfConsumer.java new file mode 100644 index 000000000..89eb67dd7 --- /dev/null +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfConsumer.java @@ -0,0 +1,102 @@ +/* + * 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. + */ +package io.servicecomb.demo.perf; + +import java.util.concurrent.CompletableFuture; +import java.util.concurrent.ExecutionException; +import java.util.concurrent.Executor; +import java.util.concurrent.Executors; + +import org.slf4j.Logger; +import org.slf4j.LoggerFactory; +import org.springframework.stereotype.Component; + +import io.servicecomb.provider.pojo.Invoker; + +@Component +public class PerfConsumer { + private static final Logger LOGGER = LoggerFactory.getLogger(PerfConsumer.class); + + private Intf intf; + + public void runConsumer() throws InterruptedException, ExecutionException { + intf = Invoker.createProxy(+ PerfConfiguration.producer, + "impl", + Intf.class); + + if (PerfConfiguration.sync) { + runSyncConsumers(); + return; + } + + runAsyncConsumers(); + } + + private void runAsyncConsumers() throws InterruptedException, ExecutionException { + CompletableFuture<String> future = intf.asyncQuery(PerfConfiguration.id, + PerfConfiguration.step, + PerfConfiguration.all, + PerfConfiguration.fromDB); + LOGGER.info("runAsyncConsumer: {}", future.get()); + + for (int idx = 0; idx < PerfConfiguration.asyncCount; idx++) { + runAsyncConsumer(); + } + } + + private void runAsyncConsumer() { + CompletableFuture<String> future = intf.asyncQuery(PerfConfiguration.id, + PerfConfiguration.step, + PerfConfiguration.all, + PerfConfiguration.fromDB); + future.whenComplete((r, e) -> { + if (e == null) { + runAsyncConsumer(); + return; + } + throw new IllegalStateException("invoke failed.", e); + }); + } + + private void runSyncConsumers() { + LOGGER.info("runSyncConsumer: {}", + intf.syncQuery(PerfConfiguration.id, + PerfConfiguration.step, + PerfConfiguration.all, + PerfConfiguration.fromDB)); + + Executor executor = Executors.newFixedThreadPool(PerfConfiguration.syncCount); + for (int idx = 0; idx < PerfConfiguration.syncCount; idx++) { + executor.execute(this::runSyncConsumer); + } + } + + private void runSyncConsumer() { + try { + for (;;) { + intf.syncQuery(PerfConfiguration.id, + PerfConfiguration.step, + PerfConfiguration.all, + PerfConfiguration.fromDB); + } + } catch (Throwable e) { + throw new IllegalStateException("invoke failed.", e); + } + } +} diff --git a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/PerfClient.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMain.java similarity index 55% rename from demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/PerfClient.java rename to demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMain.java index 992b358c9..ff872d1e2 100644 --- a/demo/demo-perf-client/src/main/java/io/servicecomb/demo/client/perf/PerfClient.java +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMain.java @@ -15,32 +15,35 @@ * limitations under the License. */ -package io.servicecomb.demo.client.perf; +package io.servicecomb.demo.perf; + +import java.util.Arrays; +import java.util.List; +import java.util.concurrent.Executors; +import java.util.concurrent.TimeUnit; -import io.servicecomb.core.CseContext; import io.servicecomb.foundation.common.utils.BeanUtils; -import io.servicecomb.foundation.common.utils.Log4jUtils; import io.servicecomb.foundation.vertx.VertxUtils; -import io.vertx.core.Vertx; +import io.servicecomb.metrics.core.publish.DataSource; -public class PerfClient { +public class PerfMain { public static void main(String[] args) throws Exception { - Log4jUtils.init(); BeanUtils.init(); - System.out.println("mode:" + Config.getMode()); + // redis RedisClientUtils.init(VertxUtils.getOrCreateVertxByName("transport", null)); - CseContext.getInstance().getConsumerProviderManager().setTransport("pojo", Config.getTransport()); - System.out.printf("test %s performance\n", Config.getTransport()); + // metrics + DataSource dataSource = BeanUtils.getContext().getBean(DataSource.class); + PerfMetricsFilePublisher metricsLog = new PerfMetricsFilePublisher(dataSource); + Executors.newScheduledThreadPool(1).scheduleWithFixedDelay(metricsLog::onCycle, 0, 1, TimeUnit.SECONDS); - if ("reactive".equals(Config.getMode())) { - Vertx vertx = VertxUtils.getOrCreateVertxByName("perfClient", null); - VertxUtils.deployVerticle(vertx, ClientVerticle.class, Config.getClientThread()); - return; - } - - for (int idx = 0; idx < Config.getClientThread(); idx++) { - new ClientThread().start(); + List<String> argList = Arrays.asList(args); + if (argList.contains("-c")) { + PerfConsumer consumer = BeanUtils.getContext().getBean(PerfConsumer.class); + consumer.runConsumer(); } } } diff --git a/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMetricsFilePublisher.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMetricsFilePublisher.java new file mode 100644 index 000000000..7a9ed07d4 --- /dev/null +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/PerfMetricsFilePublisher.java @@ -0,0 +1,82 @@ +/* + * 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. + */ + +package io.servicecomb.demo.perf; + +import java.util.Map.Entry; + +import org.slf4j.Logger; +import org.slf4j.LoggerFactory; + +import io.servicecomb.metrics.common.ConsumerInvocationMetric; +import io.servicecomb.metrics.common.ProducerInvocationMetric; +import io.servicecomb.metrics.common.RegistryMetric; +import io.servicecomb.metrics.core.publish.DataSource; + +public class PerfMetricsFilePublisher { + private static final Logger LOGGER = LoggerFactory.getLogger(PerfMetricsFilePublisher.class); + + private DataSource dataSource; + + public PerfMetricsFilePublisher(DataSource dataSource) { + this.dataSource = dataSource; + } + + public void onCycle() { + RegistryMetric metric = dataSource.getRegistryMetric(); + + StringBuilder sb = new StringBuilder(); + sb.append("\n"); + + double cpu = metric.getInstanceMetric().getSystemMetric().getCpuLoad(); + // can not get cpu usage in windows, so skip this information + if (cpu >= 0) { + sb.append("cpu: ") + .append((long) cpu * Runtime.getRuntime().availableProcessors()) + .append("%\n"); + } + + sb.append("consumer:\n" + + " total tps latency(ms) name\n"); + for (Entry<String, ConsumerInvocationMetric> entry : metric.getConsumerMetrics().entrySet()) { + String opName = entry.getKey(); +

sb.append(String + .format(" %-16d%-16d%-16.3f%s\n", + entry.getValue().getConsumerCall().getTotal(), + (long) entry.getValue().getConsumerCall().getTps(), + entry.getValue().getConsumerLatency().getAverage(), + opName)); + } + + sb.append("producer:\n" + + " total tps latency(ms) queue(ms) execute(ms) name\n"); + for (Entry<String, ProducerInvocationMetric> entry : metric.getProducerMetrics().entrySet()) { + String opName = entry.getKey(); + sb.append(+ String.format(" %-16d%-16d%-16.3f%-16.3f%-16.3f%s\n", + entry.getValue().getProducerCall().getTotal(), + (long) entry.getValue().getProducerCall().getTps(), + entry.getValue().getProducerLatency().getAverage(), + entry.getValue().getLifeTimeInQueue().getAverage(), + entry.getValue().getExecutionTime().getAverage(), + opName)); + } + + sb.setLength(sb.length() - 1); + + LOGGER.info(sb.toString()); + } +} diff --git a/demo/perf/src/main/java/io/servicecomb/demo/perf/RedisClientUtils.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/RedisClientUtils.java new file mode 100644 index 000000000..b1be93526 --- /dev/null +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/RedisClientUtils.java @@ -0,0 +1,73 @@ +/* + * 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. + */ +package io.servicecomb.demo.perf; + +import java.util.concurrent.CompletableFuture; +import java.util.concurrent.ExecutionException; + +import javax.ws.rs.core.Response.Status; + +import io.servicecomb.foundation.vertx.VertxUtils; +import io.servicecomb.foundation.vertx.client.ClientPoolFactory; +import io.servicecomb.foundation.vertx.client.ClientPoolManager; +import io.servicecomb.foundation.vertx.client.ClientVerticle; +import io.servicecomb.swagger.invocation.exception.InvocationException; +import io.vertx.core.DeploymentOptions; +import io.vertx.core.Vertx; +import io.vertx.redis.RedisClient; +import io.vertx.redis.RedisOptions; + +public class RedisClientUtils { + private static ClientPoolManager<RedisClient> clientMgr; + + public static void init(Vertx vertx) throws InterruptedException { + RedisOptions redisOptions = new RedisOptions() + .setHost(PerfConfiguration.redisHost) + .setPort(PerfConfiguration.redisPort) + .setAuth(PerfConfiguration.redisPassword); + ClientPoolFactory<RedisClient> factory = () -> { + return RedisClient.create(vertx, redisOptions); + }; + clientMgr = new ClientPoolManager<>(vertx, factory); + + DeploymentOptions deployOptions = VertxUtils.createClientDeployOptions(clientMgr, + PerfConfiguration.redisClientCount); + VertxUtils.blockDeploy(vertx, ClientVerticle.class, deployOptions); + } + + public static String syncQuery(String id) { + CompletableFuture<String> future = doQuery(id, true); + try { + return future.get(); + } catch (InterruptedException | ExecutionException e) { + throw new InvocationException(Status.INTERNAL_SERVER_ERROR.getStatusCode(), + Status.INTERNAL_SERVER_ERROR.getReasonPhrase(), (Object) "Failed to query from redis.", e); + } + } + + public static CompletableFuture<String> asyncQuery(String id) { + return doQuery(id, false); + } + + private static CompletableFuture<String> doQuery(String id, boolean sync) { + CompletableFuture<String> future = new CompletableFuture<>(); + RedisClient redisClient = clientMgr.findClientPool(sync); + RedisSession session = new RedisSession(redisClient, id, future); + session.query(); + return future; + } +} diff --git a/demo/perf/src/main/java/io/servicecomb/demo/perf/RedisSession.java b/demo/perf/src/main/java/io/servicecomb/demo/perf/RedisSession.java new file mode 100644 index 000000000..613176a26 --- /dev/null +++ b/demo/perf/src/main/java/io/servicecomb/demo/perf/RedisSession.java @@ -0,0 +1,73 @@ +/* + * Copyright 2017 Huawei Technologies Co., Ltd + * + * 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. + */ +package io.servicecomb.demo.perf; + +import java.util.concurrent.CompletableFuture; + +import io.vertx.core.AsyncResult; +import io.vertx.redis.RedisClient; + +public class RedisSession { + RedisClient redis; + + String id; + + CompletableFuture<String> future; + + String createResult; + + public RedisSession(RedisClient redis, String id, CompletableFuture<String> future) { + this.redis = redis; + this.id = id; + this.future = future; + } + + public void query() { + redis.get(id, this::onGetResponse); + } + + private void onGetResponse(AsyncResult<String> ar) { + if (ar.succeeded()) { + if (ar.result() == null) { + createCache(); + return; + } + future.complete(ar.result()); + return; + } + future.completeExceptionally(ar.cause()); + } + + private void createCache() { + createResult = new StringBuilder(64 + PerfConfiguration.responseData.length()).append(id) + .append(" from redis: ") + .append(PerfConfiguration.responseData) + .toString(); + redis.set(id, createResult, this::onCreateCacheResponse); + } + + private void onCreateCacheResponse(AsyncResult<Void> ar) { + if (ar.succeeded()) { + future.complete(createResult); + return; + } + future.completeExceptionally(ar.cause()); + } +} diff --git a/demo/demo-perf-client/src/main/resources/META-INF/spring/perf.client.bean.xml b/demo/perf/src/main/resources/logback.xml similarity index 56% rename from demo/demo-perf-client/src/main/resources/META-INF/spring/perf.client.bean.xml rename to demo/perf/src/main/resources/logback.xml index 6b3e6b737..f18c5bb06 100644 --- a/demo/demo-perf-client/src/main/resources/META-INF/spring/perf.client.bean.xml +++ b/demo/perf/src/main/resources/logback.xml @@ -16,15 +16,14 @@ ~ limitations under the License. --> -<beans xmlns="http://www.springframework.org/schema/beans" - xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p" - xmlns:util="http://www.springframework.org/schema/util" - xsi:schemaLocation=" - http://www.springframework.org/schema/beans classpath:org/springframework/beans/factory/xml/spring-beans-3.0.xsd"> - - <bean class="io.servicecomb.demo.client.perf.Config"> - <property name="clientThread" value="${cse.demo.client.thread}"></property> - <property name="transport" value="${cse.demo.client.transport}"></property> - <property name="mode" value="${cse.demo.client.mode}"></property> - </bean> -</beans> +<configuration scan="true"> + <jmxConfigurator /> + <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder> + <pattern>%d [%level] [%thread] - %msg (%F:%L\)%n</pattern> + </encoder> + </appender> + <root level="INFO"> + <appender-ref ref="STDOUT" /> + </root> +</configuration> \ No newline at end of file diff --git a/demo/demo-perf-client/src/main/resources/microservice.yaml b/demo/perf/src/main/resources/microservice.yaml similarity index 67% rename from demo/demo-perf-client/src/main/resources/microservice.yaml rename to demo/perf/src/main/resources/microservice.yaml index e6df45dcf..7d9b5561f 100644 --- a/demo/demo-perf-client/src/main/resources/microservice.yaml +++ b/demo/perf/src/main/resources/microservice.yaml @@ -15,28 +15,47 @@ ## limitations under the License. ## ---------- -cse-config-order: 100 -APPLICATION_ID: pojotest +APPLICATION_ID: perfTest service_description: - name: perfClient + name: perf1 version: 0.0.1 cse: service: registry: address: http://127.0.0.1:30100 rest: - client: - thread-count: 10 - connection-pool-per-thread: 1 + address: 0.0.0.0:8080?sslEnabled=false server: - thread-count: 10 + thread-count: 8 + client: + thread-count: 8 + connection: + maxPoolSize: 30 highway: - thread-count: 1 - connection-pool-per-thread: 10 - handler: - chain: - Consumer: - default: loadbalance + address: 0.0.0.0:7070?sslEnabled=false + server: + thread-count: 8 + client: + thread-count: 8 + references: + transport: rest +servicecomb: metrics: - cycle: - ms: 1000 + window_time: 1000 + +sync-count: 10 +async-count: 20 +sync: false +producer: perf1 +id: 1 +step: 1 +all: 1 +fromDB: false +response-size: 1 + +redis: + client: + count: 8 + host: localhost + port: 6379 +# password: diff --git a/demo/pom.xml b/demo/pom.xml index a940ed24b..e2296d79b 100644 --- a/demo/pom.xml +++ b/demo/pom.xml @@ -35,7 +35,7 @@ <module>demo-server-servlet</module> <module>demo-pojo</module> <module>demo-jaxrs</module> - <module>demo-perf-client</module> + <module>demo-springmvc</module> <module>demo-schema</module> <module>demo-crossapp</module> @@ -45,6 +45,8 @@ <module>demo-multiple</module> <module>demo-signature</module> <module>demo-edge</module> + + <module>perf</module> </modules> <dependencyManagement> diff --git a/foundations/foundation-config-cc/src/main/java/io/servicecomb/config/client/ConfigCenterClient.java b/foundations/foundation-config-cc/src/main/java/io/servicecomb/config/client/ConfigCenterClient.java index 8dcd078a3..0170f929c 100644 --- a/foundations/foundation-config-cc/src/main/java/io/servicecomb/config/client/ConfigCenterClient.java +++ b/foundations/foundation-config-cc/src/main/java/io/servicecomb/config/client/ConfigCenterClient.java @@ -54,7 +54,8 @@ import io.servicecomb.foundation.vertx.VertxTLSBuilder; import io.servicecomb.foundation.vertx.VertxUtils; import io.servicecomb.foundation.vertx.client.ClientPoolManager; -import io.servicecomb.foundation.vertx.client.http.HttpClientVerticle; +import io.servicecomb.foundation.vertx.client.ClientVerticle; +import io.servicecomb.foundation.vertx.client.http.HttpClientPoolFactory; import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; import io.vertx.core.DeploymentOptions; import io.vertx.core.Vertx; @@ -99,7 +100,7 @@ private ConfigCenterConfigurationSourceImpl.UpdateHandler updateHandler; - private static ClientPoolManager<HttpClientWithContext> clientMgr = new ClientPoolManager<>(); + private static ClientPoolManager<HttpClientWithContext> clientMgr; private boolean isWatching = false; @@ -152,9 +153,12 @@ private void refreshMembers(MemberDiscovery memberDiscovery) { private void deployConfigClient() throws InterruptedException { Vertx vertx = VertxUtils.getOrCreateVertxByName("config-center", null); + HttpClientOptions httpClientOptions = createHttpClientOptions(); - DeploymentOptions deployOptions = VertxUtils.createClientDeployOptions(clientMgr, 1, 1, httpClientOptions); - VertxUtils.blockDeploy(vertx, HttpClientVerticle.class, deployOptions); + clientMgr = new ClientPoolManager<>(vertx, new HttpClientPoolFactory(httpClientOptions)); + + DeploymentOptions deployOptions = VertxUtils.createClientDeployOptions(clientMgr, 1); + VertxUtils.blockDeploy(vertx, ClientVerticle.class, deployOptions); } private HttpClientOptions createHttpClientOptions() { diff --git a/foundations/foundation-ssl/src/main/java/io/servicecomb/foundation/ssl/SSLOptionFactory.java b/foundations/foundation-ssl/src/main/java/io/servicecomb/foundation/ssl/SSLOptionFactory.java index 8f8f0e97c..60888ece5 100644 --- a/foundations/foundation-ssl/src/main/java/io/servicecomb/foundation/ssl/SSLOptionFactory.java +++ b/foundations/foundation-ssl/src/main/java/io/servicecomb/foundation/ssl/SSLOptionFactory.java @@ -33,7 +33,7 @@ static SSLOptionFactory createSSLOptionFactory(String className) { try { return (SSLOptionFactory) Class.forName(className).newInstance(); } catch (InstantiationException | IllegalAccessException | ClassNotFoundException e) { - return null; + throw new IllegalStateException("Failed to create SSLOptionFactory.", e); } } return null; diff --git a/foundations/foundation-ssl/src/test/java/io/servicecomb/foundation/ssl/TestSSLOptionFactory.java b/foundations/foundation-ssl/src/test/java/io/servicecomb/foundation/ssl/TestSSLOptionFactory.java index 832f33cda..cde0b3ff3 100644 --- a/foundations/foundation-ssl/src/test/java/io/servicecomb/foundation/ssl/TestSSLOptionFactory.java +++ b/foundations/foundation-ssl/src/test/java/io/servicecomb/foundation/ssl/TestSSLOptionFactory.java @@ -17,8 +17,11 @@ package io.servicecomb.foundation.ssl; +import org.hamcrest.Matchers; import org.junit.Assert; +import org.junit.Rule; import org.junit.Test; +import org.junit.rules.ExpectedException; import com.netflix.config.ConcurrentCompositeConfiguration; @@ -28,6 +31,9 @@ import mockit.Mocked; public class TestSSLOptionFactory { + @Rule + public ExpectedException expectedException = ExpectedException.none(); + @Test public void testSSLOptionFactory() { SSLOptionFactory factory = SSLOptionFactory.createSSLOptionFactory("cc", null); @@ -42,8 +48,10 @@ public void testSSLOptionFactoryWrong(@Mocked SSLOption option) { result = "wrong"; } }; - SSLOptionFactory factory = SSLOptionFactory.createSSLOptionFactory("cc", null); - Assert.assertEquals(factory, null); + + expectedException.expect(IllegalStateException.class); + expectedException.expectMessage(Matchers.is("Failed to create SSLOptionFactory.")); + SSLOptionFactory.createSSLOptionFactory("cc", null); } @Test diff --git a/foundations/foundation-test-scaffolding/pom.xml b/foundations/foundation-test-scaffolding/pom.xml index e687819cc..fb4f46dc4 100644 --- a/foundations/foundation-test-scaffolding/pom.xml +++ b/foundations/foundation-test-scaffolding/pom.xml @@ -34,6 +34,11 @@ <groupId>com.netflix.archaius</groupId> <artifactId>archaius-core</artifactId> </dependency> + <dependency> + <groupId>commons-configuration</groupId> + <artifactId>commons-configuration</artifactId> + <scope>compile</scope> + </dependency> <dependency> <groupId>io.vertx</groupId> <artifactId>vertx-web</artifactId> diff --git a/foundations/foundation-test-scaffolding/src/main/java/io/servicecomb/foundation/test/scaffolding/config/ArchaiusUtils.java b/foundations/foundation-

test-scaffolding/src/main/java/io/servicecomb/foundation/test/scaffolding/config/ArchaiusUtils.java index 34891b861..5d58f9135 100644 --- a/foundations/foundation-test-scaffolding/src/main/java/io/servicecomb/foundation/test/scaffolding/config/ArchaiusUtils.java +++ b/foundations/foundation-test-scaffolding/src/main/java/io/servicecomb/foundation/test/scaffolding/config/ArchaiusUtils.java @@ -19,6 +19,7 @@ import java.lang.reflect.Field; +import org.apache.commons.configuration.Configuration; import org.springframework.util.ReflectionUtils; import com.netflix.config.ConfigurationManager; @@ -57,4 +58,11 @@ public static void resetConfig() { ReflectionUtils.setField(FIELD_INITIALIZED_WITH_DEFAULT_CONFIG, null, false); ReflectionUtils.setField(FIELD_DYNAMIC_PROPERTY_SUPPORTIMPL, null, null); } + + public static void setProperty(String key, Object value) { + // ensure have instance + DynamicPropertyFactory.getInstance(); + Configuration config = (Configuration) DynamicPropertyFactory.getBackingConfigurationSource(); + config.addProperty(key, value); + } } diff --git a/foundations/foundation-vertx/pom.xml b/foundations/foundation-vertx/pom.xml index cc9b6cc9e..7fc74bd09 100644 --- a/foundations/foundation-vertx/pom.xml +++ b/foundations/foundation-vertx/pom.xml @@ -47,5 +47,9 @@ <artifactId>slf4j-log4j12</artifactId> <scope>test</scope> </dependency> + <dependency> + <groupId>io.servicecomb</groupId> + <artifactId>foundation-test-scaffolding</artifactId> + </dependency> </dependencies> </project> diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxTLSBuilder.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxTLSBuilder.java index 6fcf1e282..44c65f68b 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxTLSBuilder.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxTLSBuilder.java @@ -22,6 +22,7 @@ import io.servicecomb.foundation.ssl.SSLCustom; import io.servicecomb.foundation.ssl.SSLManager; import io.servicecomb.foundation.ssl.SSLOption; +import io.servicecomb.foundation.ssl.SSLOptionFactory; import io.vertx.core.http.ClientAuth; import io.vertx.core.http.HttpClientOptions; import io.vertx.core.net.ClientOptionsBase; @@ -50,6 +51,18 @@ public static NetServerOptions buildNetServerOptions(SSLOption sslOption, SSLCus return netServerOptions; } + public static void buildHttpClientOptions(String sslKey, HttpClientOptions httpClientOptions) { + SSLOptionFactory factory = SSLOptionFactory.createSSLOptionFactory(sslKey, null); + SSLOption sslOption; + if (factory == null) { + sslOption = SSLOption.buildFromYaml(sslKey); + } else { + sslOption = factory.createSSLOption(); + } + SSLCustom sslCustom = SSLCustom.createSSLCustom(sslOption.getSslCustomClass()); + buildHttpClientOptions(sslOption, sslCustom, httpClientOptions); + } + public static HttpClientOptions buildHttpClientOptions(SSLOption sslOption, SSLCustom sslCustom, HttpClientOptions httpClientOptions) { buildClientOptionsBase(sslOption, sslCustom, httpClientOptions); @@ -70,19 +83,19 @@ public static ClientOptionsBase buildClientOptionsBase(SSLOption sslOption, SSLC } private static TCPSSLOptions buildTCPSSLOptions(SSLOption sslOption, SSLCustom sslCustom, - TCPSSLOptions httpClientOptions) { - httpClientOptions.setSsl(true); + TCPSSLOptions tcpClientOptions) { + tcpClientOptions.setSsl(true); if (isFileExists(sslCustom.getFullPath(sslOption.getKeyStore()))) { if (STORE_PKCS12.equalsIgnoreCase(sslOption.getKeyStoreType())) { PfxOptions keyPfxOptions = new PfxOptions(); keyPfxOptions.setPath(sslCustom.getFullPath(sslOption.getKeyStore())); keyPfxOptions.setPassword(new String(sslCustom.decode(sslOption.getKeyStoreValue().toCharArray()))); - httpClientOptions.setPfxKeyCertOptions(keyPfxOptions); + tcpClientOptions.setPfxKeyCertOptions(keyPfxOptions); } else if (STORE_JKS.equalsIgnoreCase(sslOption.getKeyStoreType())) { JksOptions keyJksOptions = new JksOptions(); keyJksOptions.setPath(sslCustom.getFullPath(sslOption.getKeyStore())); keyJksOptions.setPassword(new String(sslCustom.decode(sslOption.getKeyStoreValue().toCharArray()))); - httpClientOptions.setKeyStoreOptions(keyJksOptions); + tcpClientOptions.setKeyStoreOptions(keyJksOptions); } else { throw new IllegalArgumentException("invalid key store type."); } @@ -94,29 +107,29 @@ private static TCPSSLOptions buildTCPSSLOptions(SSLOption sslOption, SSLCustom s trustPfxOptions.setPath(sslCustom.getFullPath(sslOption.getTrustStore())); trustPfxOptions .setPassword(new String(sslCustom.decode(sslOption.getTrustStoreValue().toCharArray()))); - httpClientOptions.setPfxTrustOptions(trustPfxOptions); + tcpClientOptions.setPfxTrustOptions(trustPfxOptions); } else if (STORE_JKS.equalsIgnoreCase(sslOption.getTrustStoreType())) { JksOptions trustJksOptions = new JksOptions(); trustJksOptions.setPath(sslCustom.getFullPath(sslOption.getTrustStore())); trustJksOptions .setPassword(new String(sslCustom.decode(sslOption.getTrustStoreValue().toCharArray()))); - httpClientOptions.setTrustStoreOptions(trustJksOptions); + tcpClientOptions.setTrustStoreOptions(trustJksOptions); } else { throw new IllegalArgumentException("invalid trust store type."); } } for (String protocol : sslOption.getProtocols().split(",")) { - httpClientOptions.addEnabledSecureTransportProtocol(protocol); + tcpClientOptions.addEnabledSecureTransportProtocol(protocol); } for (String cipher : SSLManager.getEnalbedCiphers(sslOption.getCiphers())) { - httpClientOptions.addEnabledCipherSuite(cipher); + tcpClientOptions.addEnabledCipherSuite(cipher); } if (isFileExists(sslCustom.getFullPath(sslOption.getCrl()))) { - httpClientOptions.addCrlPath(sslCustom.getFullPath(sslOption.getCrl())); + tcpClientOptions.addCrlPath(sslCustom.getFullPath(sslOption.getCrl())); } - return httpClientOptions; + return tcpClientOptions; } private static boolean isFileExists(String name) { diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxUtils.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxUtils.java index e98fc0998..361773530 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxUtils.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/VertxUtils.java @@ -31,8 +31,8 @@ import org.slf4j.LoggerFactory; import io.netty.buffer.ByteBuf; -import io.servicecomb.foundation.vertx.client.AbstractClientVerticle; import io.servicecomb.foundation.vertx.client.ClientPoolManager; +import io.servicecomb.foundation.vertx.client.ClientVerticle; import io.servicecomb.foundation.vertx.stream.BufferInputStream; import io.vertx.core.AbstractVerticle; import io.vertx.core.Context; @@ -72,15 +72,12 @@ private VertxUtils() { vertx.deployVerticle(cls.getName(), options); } - public

static <CLIENT_POOL, CLIENT_OPTIONS> DeploymentOptions createClientDeployOptions(+ public static <CLIENT_POOL> DeploymentOptions createClientDeployOptions(ClientPoolManager<CLIENT_POOL> clientMgr, - int instanceCount, - int poolCountPerVerticle, CLIENT_OPTIONS clientOptions) { + int instanceCount) { DeploymentOptions options = new DeploymentOptions().setInstances(instanceCount); SimpleJsonObject config = new SimpleJsonObject(); - config.put(AbstractClientVerticle.CLIENT_MGR, clientMgr); - config.put(AbstractClientVerticle.POOL_COUNT, poolCountPerVerticle); - config.put(AbstractClientVerticle.CLIENT_OPTIONS, clientOptions); + config.put(ClientVerticle.CLIENT_MGR, clientMgr); options.setConfig(config); return options; diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/ClientPoolManager.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/ClientPoolManager.java index 386c66ac6..004b13ab8 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/ClientPoolManager.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/ClientPoolManager.java @@ -17,21 +17,37 @@ package io.servicecomb.foundation.vertx.client; -import java.util.ArrayList; import java.util.List; import java.util.Map; +import java.util.UUID; import java.util.concurrent.ConcurrentHashMap; +import java.util.concurrent.CopyOnWriteArrayList; import java.util.concurrent.atomic.AtomicInteger; +import io.vertx.core.Context; +import io.vertx.core.Vertx; + /** * CLIENT_POOL是一个完备的连接池，支持向同一个目标建立一个或多个连接 * 之所以再包装一层，是因为多个线程使用一个连接池的场景下 * 会导致多个线程抢连接池的同一把锁 - * 包装之后，允许使用m个网络线程，每个线程中有n个连接池 + * 包装之后，允许使用m个网络线程，每个线程中有1个连接池 + * + * support both sync and reactive invoke. + * 1.sync invoke, bind to a net thread + * 2.async but not in eventloop, select but not bind to a net thread + * 3.async and in eventloop, use clientPool in self thread + * + * sync/async is not about net operation, just about consumer invoke mode. */ public class ClientPoolManager<CLIENT_POOL> { - // 多个网络线程 - private List<NetThreadData<CLIENT_POOL>> netThreads = new ArrayList<>(); + private Vertx vertx; + + private String id = UUID.randomUUID().toString(); + + private ClientPoolFactory<CLIENT_POOL> factory; + + private List<CLIENT_POOL> pools = new CopyOnWriteArrayList<>(); private AtomicInteger bindIndex = new AtomicInteger(); @@ -40,29 +56,64 @@ // TODO:要不要考虑已经绑定的线程消失了的场景？ private Map<Long, CLIENT_POOL> threadBindMap = new ConcurrentHashMap<>(); - private static final Object LOCK = new Object(); + public ClientPoolManager(Vertx vertx, ClientPoolFactory<CLIENT_POOL> factory) { + this.vertx = vertx; + this.factory = factory; + } + + public CLIENT_POOL createClientPool() { + CLIENT_POOL pool = factory.createClientPool(); + addPool(pool); + return pool; + } - public void addNetThread(NetThreadData<CLIENT_POOL> netThread) { - synchronized (LOCK) { - netThreads.add(netThread); + protected void addPool(CLIENT_POOL pool) { + Vertx.currentContext().put(id, pool); + pools.add(pool); + } + + public CLIENT_POOL findClientPool(boolean sync) { + if (sync) { + return findThreadBindClientPool(); } + + // reactive mode + return findByContext(); } public CLIENT_POOL findThreadBindClientPool() { - long threadId = Thread.currentThread().getId(); - CLIENT_POOL clientPool = threadBindMap.get(threadId); - if (clientPool == null) { - synchronized (LOCK) { - clientPool = threadBindMap.get(threadId); - if (clientPool == null) { - int idx = bindIndex.getAndIncrement() % netThreads.size(); - NetThreadData<CLIENT_POOL> netThread = netThreads.get(idx); - clientPool = netThread.selectClientPool(); - threadBindMap.put(threadId, clientPool); - } + protected CLIENT_POOL findByContext() { + Context currentContext = Vertx.currentContext(); + if (currentContext != null && currentContext.owner() == vertx && currentContext.isEventLoopContext()) { + // standard reactive mode + CLIENT_POOL clientPool = currentContext.get(id); + if (clientPool != null) { + return clientPool; } + + // this will make "client.thread-count" bigger than which in microservice.yaml + // maybe it's better to remove "client.thread-count", just use "rest/highway.thread-count" + return createClientPool(); } - return clientPool; + + // not in correct context: + // 1.normal thread + // 2.vertx worker thread + // 3.other vertx thread + // select a existing context + return nextPool(); + } + + public CLIENT_POOL findThreadBindClientPool() { + long threadId = Thread.currentThread().getId(); + return threadBindMap.computeIfAbsent(threadId, tid -> { + return nextPool(); + }); + } + + protected CLIENT_POOL nextPool() { + int idx = bindIndex.getAndIncrement() % pools.size(); + return pools.get(idx); } } diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/AbstractClientVerticle.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/ClientVerticle.java similarity index 72% rename from foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/AbstractClientVerticle.java rename to foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/ClientVerticle.java index 9d9704057..43e777201 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/AbstractClientVerticle.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/ClientVerticle.java @@ -19,21 +19,13 @@ import io.vertx.core.AbstractVerticle; -public abstract class AbstractClientVerticle<CLIENT_POOL> extends AbstractVerticle - implements ClientPoolFactory<CLIENT_POOL> { +public class ClientVerticle<CLIENT_POOL> extends AbstractVerticle { public static final String CLIENT_MGR = "clientMgr"; - public static final String POOL_COUNT = "poolCount"; - - public static final String CLIENT_OPTIONS = "clientOptions"; - @SuppressWarnings("unchecked") @Override public void start() throws Exception { ClientPoolManager<CLIENT_POOL> clientMgr = (ClientPoolManager<CLIENT_POOL>) config().getValue(CLIENT_MGR); - Integer poolCount = config().getInteger(POOL_COUNT); - - NetThreadData<CLIENT_POOL> netThreadData = new NetThreadData<>(this, poolCount); - clientMgr.addNetThread(netThreadData); + clientMgr.createClientPool(); } } diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/NetThreadData.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/NetThreadData.java deleted file mode 100644 index 33a972dc3..000000000 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/NetThreadData.java +++ /dev/null @@ -1,61 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.foundation.vertx.client; - -import java.util.concurrent.atomic.AtomicInteger; - -public class NetThreadData<CLIENT_POOL> { - // 每个网络线程提供一个factory - private ClientPoolFactory<CLIENT_POOL> factory; - - // 每个网络线程有多个连接池 - private CLIENT_POOL[] pools; - - private AtomicInteger bindIndex = new AtomicInteger(); - - @SuppressWarnings("unchecked") - public NetThreadData(ClientPoolFactory<CLIENT_POOL> factory, int poolCount) { - this.factory = factory; - pools = (CLIENT_POOL[]) new Object[poolCount]; - } - - public ClientPoolFactory<CLIENT_POOL> getFactory() { - return factory; - } - - public CLIENT_POOL[] getPools() { - return pools; - } - - public AtomicInteger getBindIndex() { - return bindIndex; - } - /** - * 在ClientPoolManager中被调用，是被锁保护的 - */ - public CLIENT_POOL selectClientPool() { - int idx = bindIndex.getAndIncrement() % pools.length; - CLIENT_POOL clientPool = pools[idx]; - if (clientPool == null) { - clientPool = factory.createClientPool(); - pools[idx] = clientPool; - } - return clientPool; - } -} diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/http/HttpClientVerticle.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/http/HttpClientPoolFactory.java similarity index 66% rename from foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/http/HttpClientVerticle.java rename to foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/http/HttpClientPoolFactory.java index c88cb975b..9cb589e23 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/http/HttpClientVerticle.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/http/HttpClientPoolFactory.java @@ -17,15 +17,24 @@ package io.servicecomb.foundation.vertx.client.http; -import io.servicecomb.foundation.vertx.client.AbstractClientVerticle; +import io.servicecomb.foundation.vertx.client.ClientPoolFactory; +import io.vertx.core.Context; +import io.vertx.core.Vertx; import io.vertx.core.http.HttpClient; import io.vertx.core.http.HttpClientOptions; -public class HttpClientVerticle extends AbstractClientVerticle<HttpClientWithContext> { +// execute in vertx context +public class HttpClientPoolFactory implements ClientPoolFactory<HttpClientWithContext> { + private HttpClientOptions httpClientOptions; + + public HttpClientPoolFactory(HttpClientOptions httpClientOptions) { + this.httpClientOptions = httpClientOptions; + } + @Override public HttpClientWithContext createClientPool() { - HttpClientOptions httpClientOptions = (HttpClientOptions) config().getValue(CLIENT_OPTIONS); - HttpClient httpClient = vertx.createHttpClient(httpClientOptions); + Context context = Vertx.currentContext(); + HttpClient httpClient = context.owner().createHttpClient(httpClientOptions); return new HttpClientWithContext(httpClient, context); } diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientConnectionPool.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientConnectionPool.java index 91f4846ad..dd888c0a0 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientConnectionPool.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientConnectionPool.java @@ -19,31 +19,29 @@ import java.util.Map; import java.util.concurrent.ConcurrentHashMap; +import java.util.concurrent.TimeUnit; import io.vertx.core.Context; -import io.vertx.core.net.NetClient; public abstract class AbstractTcpClientConnectionPool<T extends TcpClientConnection> { // 是在哪个context中创建的 protected Context context; - protected TcpClientConfig clientConfig; - - protected NetClient netClient; + protected NetClientWrapper netClientWrapper; // key为address protected Map<String, T> tcpClientMap = new ConcurrentHashMap<>(); - public AbstractTcpClientConnectionPool(TcpClientConfig clientConfig, Context context, NetClient netClient) { - this.clientConfig = clientConfig; + public AbstractTcpClientConnectionPool(Context context, NetClientWrapper netClientWrapper) { this.context = context; - this.netClient = netClient; - startCheckTimeout(clientConfig, context); + this.netClientWrapper = netClientWrapper; + startCheckTimeout(context); } - protected void startCheckTimeout(TcpClientConfig clientConfig, Context context) { - context.owner().setPeriodic(clientConfig.getRequestTimeoutMillis(), this::onCheckTimeout); + protected void startCheckTimeout(Context context) { + context.owner().setPeriodic(TimeUnit.SECONDS.toMillis(1), this::onCheckTimeout); } private void onCheckTimeout(Long event) { @@ -52,20 +50,8 @@ private void onCheckTimeout(Long event) { } } - public void send(TcpClientConnection tcpClient, AbstractTcpClientPackage tcpClientPackage, - TcpResponseCallback callback) { - tcpClient.send(tcpClientPackage, clientConfig.getRequestTimeoutMillis(), callback); - } - public T findOrCreateClient(String endpoint) { - T tcpClient = tcpClientMap.get(endpoint); - if (tcpClient == null) { - synchronized (this) { - tcpClient = tcpClientMap.computeIfAbsent(endpoint, this::create); - } - } - return tcpClient; + return tcpClientMap.computeIfAbsent(endpoint, this::create); } protected abstract T create(String endpoint); diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientPoolFactory.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientPoolFactory.java new file mode 100644 index 000000000..8a93e55a5 --- /dev/null +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientPoolFactory.java @@ -0,0 +1,43 @@ +/* + * 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. + */ +package io.servicecomb.foundation.vertx.client.tcp; + +import io.servicecomb.foundation.vertx.client.ClientPoolFactory; +import io.vertx.core.Context; +import io.vertx.core.Vertx; + +public abstract class AbstractTcpClientPoolFactory<CLIENT_POOL> implements ClientPoolFactory<CLIENT_POOL> { + protected TcpClientConfig normalClientConfig; + + protected TcpClientConfig sslClientConfig; + + public AbstractTcpClientPoolFactory(TcpClientConfig normalClientConfig, TcpClientConfig sslClientConfig) { + this.normalClientConfig = normalClientConfig; + this.sslClientConfig = sslClientConfig; + } + + @Override + public CLIENT_POOL createClientPool() { + Context context = Vertx.currentContext(); + Vertx vertx = context.owner(); + + NetClientWrapper netClientWrapper = new NetClientWrapper(vertx, normalClientConfig, sslClientConfig); + return doCreateClientPool(context, netClientWrapper); + } + + protected abstract CLIENT_POOL doCreateClientPool(Context context, NetClientWrapper netClientWrapper); +} diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientVerticle.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientVerticle.java deleted file mode 100644 index aed5a064c..000000000 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/AbstractTcpClientVerticle.java +++ /dev/null @@ -1,40 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.foundation.vertx.client.tcp; - -import io.servicecomb.foundation.vertx.client.AbstractClientVerticle; -import io.vertx.core.impl.VertxInternal; -import io.vertx.core.net.NetClient; -import io.vertx.core.net.impl.NetClientImpl; - -public abstract class AbstractTcpClientVerticle<T extends TcpClientConnection, P extends AbstractTcpClientConnectionPool<T>> - extends AbstractClientVerticle<P> { - protected TcpClientConfig clientConfig; - - // 每线程一个实例即可 - protected NetClient netClient; - - @Override - public void start() throws Exception { - super.start(); - clientConfig = (TcpClientConfig) config().getValue(CLIENT_OPTIONS); - - // vertx.createNetClient()创建出来的netClient不支持跨线程调用 - netClient = new NetClientImpl((VertxInternal) vertx, clientConfig, false); - } -} diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/NetClientWrapper.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/NetClientWrapper.java new file mode 100644 index 000000000..36d08aea0 --- /dev/null +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/NetClientWrapper.java @@ -0,0 +1,61 @@ +/* + * 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. + */ +package io.servicecomb.foundation.vertx.client.tcp; + +import io.vertx.core.AsyncResult; +import io.vertx.core.Handler; +import io.vertx.core.Vertx; +import io.vertx.core.net.NetClient; +import io.vertx.core.net.NetSocket; + +// netClient do not like httpClient +// can not support normal and ssl by the same instance +// so we do this wrap +public class NetClientWrapper { + private TcpClientConfig normalClientConfig; + + private NetClient normalNetClient; + + private TcpClientConfig sslClientConfig; + + private NetClient sslNetClient; + + public NetClientWrapper(Vertx vertx, TcpClientConfig normalClientConfig, TcpClientConfig sslClientConfig) { + this.normalClientConfig = normalClientConfig; + this.normalNetClient = vertx.createNetClient(normalClientConfig); + + this.sslClientConfig = sslClientConfig; + this.sslNetClient = vertx.createNetClient(sslClientConfig); + } + + public TcpClientConfig getClientConfig(boolean ssl) { + if (ssl) { + return sslClientConfig; + } + return normalClientConfig; + } + + public void connect(boolean ssl, int port, String host, Handler<AsyncResult<NetSocket>> connectHandler) { + if (ssl) { + sslNetClient.connect(port, host, connectHandler); + return; + } + normalNetClient.connect(port, host, connectHandler); + } +} diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnection.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnection.java index 9639c1663..ad27d42ff 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnection.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnection.java @@ -29,6 +29,8 @@ import org.slf4j.Logger; import org.slf4j.LoggerFactory; +import com.google.common.annotations.VisibleForTesting; + import io.servicecomb.foundation.common.net.URIEndpointObject; import io.servicecomb.foundation.vertx.server.TcpParser; import io.servicecomb.foundation.vertx.tcp.TcpConnection; @@ -37,7 +39,6 @@ import io.vertx.core.AsyncResult; import io.vertx.core.Context; import io.vertx.core.buffer.Buffer; -import io.vertx.core.net.NetClient; import io.vertx.core.net.NetSocket; import io.vertx.core.net.impl.NetSocketImpl; @@ -51,10 +52,12 @@ WORKING } - private NetClient netClient; + private NetClientWrapper netClientWrapper; private TcpClientConfig clientConfig; + private URIEndpointObject endpoint; + private InetSocketAddress socketAddress; private boolean localSupportLogin = false; @@ -70,14 +73,14 @@ private volatile Map<Long, TcpRequest> requestMap = new ConcurrentHashMap<>(); - public TcpClientConnection(Context context, NetClient netClient, String endpoint, TcpClientConfig clientConfig) { + public TcpClientConnection(Context context, NetClientWrapper netClientWrapper, String strEndpoint) { this.setContext(context); - this.netClient = netClient; - URIEndpointObject ipPort = new URIEndpointObject(endpoint); - this.socketAddress = ipPort.getSocketAddress(); - this.remoteSupportLogin = Boolean.parseBoolean(ipPort.getFirst(TcpConst.LOGIN)); -

this.clientConfig = clientConfig; + this.netClientWrapper = netClientWrapper; + endpoint = new URIEndpointObject(strEndpoint); + this.socketAddress = endpoint.getSocketAddress(); + this.remoteSupportLogin = Boolean.parseBoolean(endpoint.getFirst(TcpConst.LOGIN)); + this.clientConfig = netClientWrapper.getClientConfig(endpoint.isSslEnabled()); } public boolean isLocalSupportLogin() { @@ -96,9 +99,8 @@ protected boolean onLoginResponse(Buffer bodyBuffer) { return true; } - public void send(AbstractTcpClientPackage tcpClientPackage, long msTimeout, - TcpResponseCallback callback) { - requestMap.put(tcpClientPackage.getMsgId(), new TcpRequest(msTimeout, callback)); + public void send(AbstractTcpClientPackage tcpClientPackage, TcpResponseCallback callback) { + requestMap.put(tcpClientPackage.getMsgId(), new TcpRequest(clientConfig.getRequestTimeoutMillis(), callback)); if (writeToBufferQueue(tcpClientPackage)) { return; @@ -154,18 +156,22 @@ private void writePackageInContext() { } } - private void connect() { + @VisibleForTesting + protected void connect() { this.status = Status.CONNECTING; LOGGER.info("connecting to address {}", socketAddress.toString()); - netClient.connect(socketAddress.getPort(), socketAddress.getHostString(), ar -> { - if (ar.succeeded()) { - onConnectSuccess(ar.result()); - return; - } - onConnectFailed(ar.cause()); - }); + netClientWrapper.connect(endpoint.isSslEnabled(), + socketAddress.getPort(), + socketAddress.getHostString(), + ar -> { + if (ar.succeeded()) { + onConnectSuccess(ar.result()); + return; + } + onConnectFailed(ar.cause()); + }); } private void onConnectSuccess(NetSocket socket) { diff --git a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnectionPool.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnectionPool.java index 21f5afcd4..14c2e0668 100644 --- a/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnectionPool.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientConnectionPool.java @@ -17,18 +17,14 @@ package io.servicecomb.foundation.vertx.client.tcp; import io.vertx.core.Context; -import io.vertx.core.net.NetClient; public class TcpClientConnectionPool extends AbstractTcpClientConnectionPool<TcpClientConnection> { - public TcpClientConnectionPool(TcpClientConfig clientConfig, Context context, NetClient netClient) { - super(clientConfig, context, netClient); + public TcpClientConnectionPool(Context context, NetClientWrapper netClientWrapper) { + super(context, netClientWrapper); } - /** - * {@inheritDoc} - */ @Override protected TcpClientConnection create(String endpoint) { - return new TcpClientConnection(context, netClient, endpoint, clientConfig); + return new TcpClientConnection(context, netClientWrapper, endpoint); } } diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientVerticle.java b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientPoolFactory.java similarity index 61% rename from transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientVerticle.java rename to foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientPoolFactory.java index d9fcacd4e..343df70bf 100644 --- a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientVerticle.java +++ b/foundations/foundation-vertx/src/main/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientPoolFactory.java @@ -14,14 +14,17 @@ * See the License for the specific language governing permissions and * limitations under the License. */ -package io.servicecomb.transport.highway; +package io.servicecomb.foundation.vertx.client.tcp; -import io.servicecomb.foundation.vertx.client.tcp.AbstractTcpClientVerticle; +import io.vertx.core.Context; + +public class TcpClientPoolFactory extends AbstractTcpClientPoolFactory<TcpClientConnectionPool> { + public TcpClientPoolFactory(TcpClientConfig normalClientConfig, TcpClientConfig sslClientConfig) { + super(normalClientConfig, sslClientConfig); + } -public class HighwayClientVerticle - extends AbstractTcpClientVerticle<HighwayClientConnection, HighwayClientConnectionPool> { @Override - public HighwayClientConnectionPool createClientPool() { - return new HighwayClientConnectionPool(clientConfig, context, netClient); + protected TcpClientConnectionPool doCreateClientPool(Context context, NetClientWrapper netClientWrapper) { + return new TcpClientConnectionPool(context, netClientWrapper); } } diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestClient.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestClient.java deleted file mode 100644 index 4bedac383..000000000 --- a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestClient.java +++ /dev/null @@ -1,43 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.foundation.vertx; - -import org.junit.Assert; -import org.junit.Test; - -import io.servicecomb.foundation.vertx.client.http.HttpClientVerticle; -import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; -import io.vertx.core.Vertx; - -public class TestClient { - @Test - public void testHttpClientVerticle() throws Exception { - Vertx vertx = VertxUtils.init(null); - - HttpClientVerticle oVerticle = new HttpClientVerticle(); - oVerticle.init(vertx, null); - Assert.assertEquals("clientMgr", HttpClientVerticle.CLIENT_MGR); - Assert.assertEquals("poolCount", HttpClientVerticle.POOL_COUNT); - Assert.assertEquals("clientOptions", HttpClientVerticle.CLIENT_OPTIONS); - HttpClientWithContext oContextClient = new HttpClientWithContext(null, null); - Assert.assertEquals(null, oContextClient.getHttpClient()); - - vertx.close(); - } -} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestTcp.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestTcp.java deleted file mode 100644 index e9c2ada5c..000000000 --- a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestTcp.java +++ /dev/null @@ -1,189 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.foundation.vertx; - -import org.junit.Assert; -import org.junit.Test; import org.mockito.Mockito; - -import io.servicecomb.foundation.common.net.URIEndpointObject; -import io.servicecomb.foundation.vertx.client.tcp.TcpClientConfig; -import io.servicecomb.foundation.vertx.client.tcp.TcpClientConnection; -import io.servicecomb.foundation.vertx.client.tcp.TcpClientConnectionPool; -import io.servicecomb.foundation.vertx.client.tcp.TcpClientPackage; -import io.servicecomb.foundation.vertx.client.tcp.TcpClientVerticle; -import io.servicecomb.foundation.vertx.client.tcp.TcpData; -import io.servicecomb.foundation.vertx.client.tcp.TcpRequest; -import io.servicecomb.foundation.vertx.client.tcp.TcpResponseCallback; -import io.servicecomb.foundation.vertx.server.TcpParser; -import io.servicecomb.foundation.vertx.server.TcpServer; -import io.servicecomb.foundation.vertx.tcp.TcpOutputStream; -import io.vertx.core.AsyncResult; -import io.vertx.core.Context; -import io.vertx.core.Handler; -import io.vertx.core.Vertx; -import io.vertx.core.buffer.Buffer; -import io.vertx.core.net.NetClient; -import io.vertx.core.net.NetSocket; -import io.vertx.core.net.SocketAddress; -import io.vertx.core.net.impl.NetSocketImpl; -import io.vertx.core.net.impl.SocketAddressImpl; -import mockit.Deencapsulation; -import mockit.Mock; -import mockit.MockUp; - -public class TestTcp { - - enum ParseStatus { - MSG_ID_AND_LEN, - // len: total len/header len - HEADER, - BODY - } - - @Test - public void testTcpClient() throws Exception { - NetClient oNetClient = new NetClient() { - - @Override - public boolean isMetricsEnabled() { - return true; - } - - @Override - public NetClient connect(int port, String host, Handler<AsyncResult<NetSocket>> connectHandler) { - return Mockito.mock(NetClient.class); - } - - @Override - public void close() { - } - - @Override - public NetClient connect(int port, String host, String serverName, - Handler<AsyncResult<NetSocket>> connectHandler) { - return null; - } - - @Override - public NetClient connect(SocketAddress remoteAddress, Handler<AsyncResult<NetSocket>> connectHandler) { - return null; - } - - @Override - public NetClient connect(SocketAddress remoteAddress, String serverName, - Handler<AsyncResult<NetSocket>> connectHandler) { - return null; - } - }; - TcpClientConnection oTcpClient = - new TcpClientConnection(Mockito.mock(Context.class), oNetClient, "highway://127.2.0.1:8080", - new TcpClientConfig()); - oTcpClient.checkTimeout(); - oTcpClient.send(new TcpClientPackage(null), 123, Mockito.mock(TcpResponseCallback.class)); - oTcpClient.send(new TcpClientPackage(null), 123, Mockito.mock(TcpResponseCallback.class)); - - new MockUp<TcpClientConnectionPool>() { - @Mock - protected void startCheckTimeout(TcpClientConfig clientConfig, Context context) { - } - }; - Vertx vertx = VertxUtils.init(null); - TcpClientConfig config = new TcpClientConfig(); - TcpClientConnectionPool oClientPool = - new TcpClientConnectionPool(config, vertx.getOrCreateContext(), oNetClient); - oClientPool.send(oTcpClient, new TcpClientPackage(null), Mockito.mock(TcpResponseCallback.class)); - oClientPool.send(oTcpClient, new TcpClientPackage(null), Mockito.mock(TcpResponseCallback.class)); - Assert.assertNotNull(oClientPool); - - TcpRequest oTcpRequest = new TcpRequest(1234, Mockito.mock(TcpResponseCallback.class)); - oTcpRequest.isTimeout(); - oTcpRequest.onReply(Buffer.buffer(), Buffer.buffer(("test").getBytes())); - oTcpRequest.onSendError(new Throwable("test Errorsss")); - Assert.assertNotNull(oTcpRequest); - - TcpClientVerticle oTcpClientVerticle = new TcpClientVerticle(); - oTcpClientVerticle.init(vertx, vertx.getOrCreateContext()); - oTcpClientVerticle.createClientPool(); - oTcpClientVerticle.createClientPool(); - Assert.assertNotNull(oTcpClientVerticle.getVertx()); - - NetSocket socket = Mockito.mock(NetSocketImpl.class); - Throwable e = Mockito.mock(Throwable.class); - Buffer hBuffer = Mockito.mock(Buffer.class); - Buffer bBuffer = Mockito.mock(Buffer.class); - - Deencapsulation.invoke(oTcpClient, "connect"); - Deencapsulation.invoke(oTcpClient, "onConnectSuccess", socket); - Mockito.when(socket.localAddress()).thenReturn(new SocketAddressImpl(0, "127.0.0.1")); - Deencapsulation.setField(oTcpClient, "netSocket", socket); - Deencapsulation.invoke(oTcpClient, "onDisconnected", e); - Deencapsulation.invoke(oTcpClient, "tryLogin"); - Deencapsulation.invoke(oTcpClient, "onLoginSuccess"); - Deencapsulation.invoke(oTcpClient, "onConnectFailed", e); - long l = 10; - Deencapsulation.invoke(oTcpClient, "onReply", l, hBuffer, bBuffer); - oTcpClient.checkTimeout(); - Assert.assertNotNull(oTcpClient); - - vertx.close(); - } - - @Test - public void testTcpOutputStream() { - TcpOutputStream oStream = new TcpOutputStream(0); - oStream.close(); - Buffer buffer = oStream.getBuffer(); - Assert.assertArrayEquals(TcpParser.TCP_MAGIC, buffer.getBytes(0, TcpParser.TCP_MAGIC.length)); - Assert.assertEquals(oStream.getMsgId(), buffer.getLong(TcpParser.TCP_MAGIC.length)); - } - - @Test - public void testTcpServerStarter() { - Vertx vertx = VertxUtils.init(null); - - URIEndpointObject endpiont = new URIEndpointObject("highway://127.0.0.1:9900"); - TcpServer oStarter = new TcpServer(endpiont); - oStarter.init(vertx, "", null); - Assert.assertNotNull(oStarter); - //TODO Need to find a way to Assert TcpServerStarter as this obbject does not return any values. - - vertx.close(); - } - - @Test - public void testTcpClientConfig() { - TcpClientConfig tcpClientConfig = new TcpClientConfig(); - tcpClientConfig.getRequestTimeoutMillis(); - tcpClientConfig.setRequestTimeoutMillis(1); - Assert.assertNotNull(tcpClientConfig); - } - - @Test - public void testTcpData() { - Buffer hBuffer = Mockito.mock(Buffer.class); - Buffer bBuffer = Mockito.mock(Buffer.class); - TcpData tcpData = new TcpData(hBuffer, bBuffer); - tcpData.getBodyBuffer(); - tcpData.setBodyBuffer(bBuffer); - tcpData.getHeaderBuffer(); - tcpData.setBodyBuffer(hBuffer); - Assert.assertNotNull(tcpData.getBodyBuffer()); - Assert.assertNotNull(tcpData.getHeaderBuffer()); - } -} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestVertxTLSBuilder.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestVertxTLSBuilder.java index cd0cbb6c8..b00ace987 100644 --- a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestVertxTLSBuilder.java +++ b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/TestVertxTLSBuilder.java @@ -17,11 +17,15 @@ @@ package io.servicecomb.foundation.vertx; +import org.junit.AfterClass; import org.junit.Assert; +import

org.junit.BeforeClass; import org.junit.Test; import io.servicecomb.foundation.ssl.SSLCustom; import io.servicecomb.foundation.ssl.SSLOption; +import io.servicecomb.foundation.ssl.SSLOptionFactory; +import io.servicecomb.foundation.test.scaffolding.config.ArchaiusUtils; import io.vertx.core.http.ClientAuth; import io.vertx.core.http.HttpClientOptions; import io.vertx.core.http.HttpServerOptions; @@ -29,6 +33,15 @@ import mockit.MockUp; public class TestVertxTLSBuilder { + @BeforeClass + public static void classSetup() { + ArchaiusUtils.resetConfig(); + } + + @AfterClass + public static void classTeardown() { + ArchaiusUtils.resetConfig(); + } @Test public void testbuildHttpServerOptions() { @@ -40,6 +53,36 @@ public void testbuildHttpServerOptions() { Assert.assertEquals(serverOptions.getClientAuth(), ClientAuth.REQUEST); } + @Test + public void testbuildHttpClientOptions_sslKey_noFactory() { + HttpClientOptions clientOptions = new HttpClientOptions(); + VertxTLSBuilder.buildHttpClientOptions("notExist", clientOptions); + Assert.assertTrue(clientOptions.isSsl()); + } + + public static class SSLOptionFactoryForTest implements SSLOptionFactory { + static SSLOption sslOption = new SSLOption(); + static { + sslOption.setProtocols(""); + sslOption.setCiphers(SSLOption.DEFAUL_CIPHERS); + sslOption.setCheckCNHost(true); + } + + @Override + public SSLOption createSSLOption() { + return sslOption; + } + } + + @Test + public void testbuildHttpClientOptions_ssl_withFactory() { + ArchaiusUtils.setProperty("ssl.exist.sslOptionFactory", SSLOptionFactoryForTest.class.getName()); + HttpClientOptions clientOptions = new HttpClientOptions(); + VertxTLSBuilder.buildHttpClientOptions("exist", clientOptions); + Assert.assertTrue(clientOptions.isSsl()); + Assert.assertTrue(clientOptions.isVerifyHost()); + } @Test public void testbuildHttpClientOptions() { SSLOption option = SSLOption.buildFromYaml("rest.consumer"); diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/TestClientPoolManager.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/TestClientPoolManager.java new file mode 100644 index 000000000..46a55ac81 --- /dev/null +++ b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/TestClientPoolManager.java @@ -0,0 +1,195 @@ +/* + * 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. + */ +package io.servicecomb.foundation.vertx.client; + +import java.util.HashMap; +import java.util.List; +import java.util.Map; + +import org.hamcrest.Matchers; +import org.junit.Assert; +import org.junit.Before; +import org.junit.Test; + +import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; +import io.vertx.core.Context; +import io.vertx.core.Vertx; +import io.vertx.core.impl.VertxImpl; +import mockit.Deencapsulation; +import mockit.Expectations; +import mockit.Mock; +import mockit.MockUp; +import mockit.Mocked; + +public class TestClientPoolManager { + @Mocked + Vertx vertx; + + @Mocked + ClientPoolFactory<HttpClientWithContext> factory; + + ClientPoolManager<HttpClientWithContext> poolMgr; + + String id; + + List<HttpClientWithContext> pools; + + Map<String, Object> contextMap = new HashMap<>(); + + Context context; + + @Before + public void setup() { + poolMgr = new ClientPoolManager<>(vertx, factory); + id = Deencapsulation.getField(poolMgr, "id"); + pools = Deencapsulation.getField(poolMgr, "pools"); + context = new MockUp<Context>() { + @Mock + void put(String key, Object value) { + contextMap.put(key, value); + } + + @SuppressWarnings("unchecked") + @Mock + <T> T get(String key) { + return (T) contextMap.get(key); + } + + @Mock + Vertx owner() { + return vertx; + } + + @Mock + boolean isEventLoopContext() { + return true; + } + }.getMockInstance(); + } + + @Test + public void createClientPool(@Mocked HttpClientWithContext pool) { + new Expectations(VertxImpl.class) { + { + factory.createClientPool(); + result = pool; + VertxImpl.context(); + result = context; + } + }; + + Assert.assertSame(pool, poolMgr.createClientPool()); + Assert.assertSame(pool, context.get(id)); + Assert.assertThat(pools, Matchers.contains(pool)); + } + + @Test + public void findClientPool_sync(@Mocked HttpClientWithContext pool1, @Mocked HttpClientWithContext pool2) { + new Expectations(poolMgr) { + { + poolMgr.findThreadBindClientPool(); + result = pool1; + poolMgr.findByContext(); + result = pool2; + } + }; + + Assert.assertSame(pool1, poolMgr.findClientPool(true)); + Assert.assertSame(pool2, poolMgr.findClientPool(false)); + } + + @Test + public void findThreadBindClientPool(@Mocked HttpClientWithContext pool1, @Mocked HttpClientWithContext pool2) { + pools.add(pool1); + pools.add(pool2); + + Assert.assertSame(pool1, poolMgr.findThreadBindClientPool()); + // find again, get the same result + Assert.assertSame(pool1, poolMgr.findThreadBindClientPool()); + } + + @Test + public void findByContext_reactive() { + HttpClientWithContext notMatchPool = new HttpClientWithContext(null, null); + pools.add(notMatchPool); + + new Expectations(VertxImpl.class) { + { + factory.createClientPool(); + result = new HttpClientWithContext(null, null); + VertxImpl.context(); + result = context; + } + }; + + HttpClientWithContext result = poolMgr.findByContext(); + Assert.assertNotSame(notMatchPool, result); + // find again, get the same result + Assert.assertSame(result, poolMgr.findByContext()); + } + + @Test + public void findByContext_normalThread() { + HttpClientWithContext pool = new HttpClientWithContext(null, null); + pools.add(pool); + + new Expectations(VertxImpl.class) { + { + VertxImpl.context(); + result = null; + } + }; + + Assert.assertSame(pool, poolMgr.findByContext()); + } + + @Test + public void findByContext_otherVertx(@Mocked Vertx otherVertx, @Mocked Context otherContext) { + HttpClientWithContext pool = new HttpClientWithContext(null, null); + pools.add(pool); + + new Expectations(VertxImpl.class) { + { + VertxImpl.context(); + result = otherContext; + otherContext.owner(); + result = otherVertx; + } + }; + + Assert.assertSame(pool, poolMgr.findByContext()); + } + + @Test + public void findByContext_woker(@Mocked Context workerContext) { + HttpClientWithContext pool = new HttpClientWithContext(null, null); + pools.add(pool); + + new Expectations(VertxImpl.class) { + { + VertxImpl.context(); + result = workerContext; + workerContext.owner(); + result = vertx; + workerContext.isEventLoopContext(); + result = false; + } + }; + + Assert.assertSame(pool, poolMgr.findByContext()); + } +} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/TestClientVerticle.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/TestClientVerticle.java new file mode 100644 index 000000000..38f1c9462 --- /dev/null +++ b/foundations/foundation-

vertx/src/test/java/io/servicecomb/foundation/vertx/client/TestClientVerticle.java @@ -0,0 +1,61 @@ +/* + * 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. + */ +package io.servicecomb.foundation.vertx.client; + +import java.util.concurrent.atomic.AtomicInteger; + +import org.junit.Assert; +import org.junit.Test; + +import io.servicecomb.foundation.vertx.SimpleJsonObject; +import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; +import io.vertx.core.Context; +import io.vertx.core.json.JsonObject; +import mockit.Expectations; +import mockit.Mock; +import mockit.MockUp; +import mockit.Mocked; + +public class TestClientVerticle { + ClientVerticle<HttpClientWithContext> clientVerticle = new ClientVerticle<>(); + + @Test + public void start(@Mocked Context context) throws Exception { + AtomicInteger count = new AtomicInteger(); + ClientPoolManager<HttpClientWithContext> clientMgr = new MockUp<ClientPoolManager<HttpClientWithContext>>() { + @Mock + HttpClientWithContext createClientPool() { + count.incrementAndGet(); + return null; + } + }.getMockInstance(); + clientVerticle.init(null, context); + + JsonObject config = new SimpleJsonObject(); + config.put(ClientVerticle.CLIENT_MGR, clientMgr); + new Expectations() { + { + context.config(); + result = config; + } + }; + + clientVerticle.start(); + + Assert.assertEquals(1, count.get()); + } +} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/http/TestHttpClientPoolFactory.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/http/TestHttpClientPoolFactory.java new file mode 100644 index 000000000..94d1dd644 --- /dev/null +++ b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/http/TestHttpClientPoolFactory.java @@ -0,0 +1,52 @@ +/* + * 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. + */ +package io.servicecomb.foundation.vertx.client.http; + +import org.junit.Assert; +import org.junit.Test; + +import io.vertx.core.Context; +import io.vertx.core.Vertx; +import io.vertx.core.http.HttpClient; +import io.vertx.core.http.HttpClientOptions; +import io.vertx.core.impl.VertxImpl; +import mockit.Expectations; +import mockit.Mocked; + +public class TestHttpClientPoolFactory { + HttpClientOptions httpClientOptions = new HttpClientOptions(); + + HttpClientPoolFactory factory = new HttpClientPoolFactory(httpClientOptions); + + @Test + public void createClientPool(@Mocked Vertx vertx, @Mocked Context context, @Mocked HttpClient httpClient) { + new Expectations(VertxImpl.class) { + { + VertxImpl.context(); + result = context; + context.owner(); + result = vertx; + vertx.createHttpClient(httpClientOptions); + result = httpClient; + } + }; + HttpClientWithContext pool = factory.createClientPool(); + + Assert.assertSame(context, pool.context()); + Assert.assertSame(httpClient, pool.getHttpClient()); + } +} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientTest.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientTest.java deleted file mode 100644 index 83734a301..000000000 --- a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TcpClientTest.java +++ /dev/null @@ -1,79 +0,0 @@ -/* - * 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. - */ -package io.servicecomb.foundation.vertx.client.tcp; - -import java.net.InetSocketAddress; - -import org.junit.After; -import org.junit.Assert; -import org.junit.Before; -import org.junit.Test; -import org.mockito.InjectMocks; -import org.mockito.Mockito; - -import io.servicecomb.foundation.common.net.IpPort; -import io.servicecomb.foundation.common.net.NetUtils; -import io.vertx.core.Context; -import io.vertx.core.net.NetClient; -import mockit.Mock; -import mockit.MockUp; - -public class TcpClientTest { - - private TcpClientConnection instance = null; - - @InjectMocks - private NetUtils netUtils; - - private void mockTestCases() { - new MockUp<NetUtils>() { - @Mock - public IpPort parseIpPort(String address) { - return Mockito.mock(IpPort.class); - } - }; - } - - @Before - public void setUp() throws Exception { - Context context = Mockito.mock(Context.class); - NetClient netClient = Mockito.mock(NetClient.class); - InetSocketAddress socketAddress = Mockito.mock(InetSocketAddress.class); - mockTestCases(); - Mockito.when(NetUtils.parseIpPort("sss").getSocketAddress()).thenReturn(socketAddress); - instance = new TcpClientConnection(context, netClient, "highway://127.0.0.1:80", new TcpClientConfig()); - } - - @After - public void tearDown() throws Exception { - instance = null; - } - - @Test - public void testCallBack() { - instance.getClass(); - TcpResponseCallback callback = Mockito.mock(TcpResponseCallback.class); - try { - instance.send(new TcpClientPackage(null), 1, callback); - Assert.assertNotNull(callback); - } catch (Exception e) { - Assert.assertEquals("java.lang.NullPointerException", e.getClass().getName()); - } - } -} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientVerticle.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestAbstractTcpClientPoolFactory.java similarity index 54% rename from foundations/foundation-

vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientVerticle.java rename to foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestAbstractTcpClientPoolFactory.java index 5cf6344f5..0d0e24b53 100644 --- a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientVerticle.java +++ b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestAbstractTcpClientPoolFactory.java @@ -14,42 +14,37 @@ @@ * See the License for the specific language governing permissions and * limitations under the License. */ - package io.servicecomb.foundation.vertx.client.tcp; import org.junit.Assert; import org.junit.Test; -import io.servicecomb.foundation.vertx.client.ClientPoolManager; import io.vertx.core.Context; -import io.vertx.core.impl.VertxInternal; -import io.vertx.core.json.JsonObject; +import io.vertx.core.Vertx; +import io.vertx.core.impl.VertxImpl; import mockit.Expectations; import mockit.Mocked; -public class TestTcpClientVerticle { - private ClientPoolManager<TcpClientConnectionPool> clientMgr = new ClientPoolManager<>(); +public class TestAbstractTcpClientPoolFactory { + private TcpClientConfig normalClientConfig = new TcpClientConfig(); - @Test - public void testTcpClientVerticle(@Mocked TcpClientConfig config, @Mocked VertxInternal vertx, - @Mocked Context context, - @Mocked JsonObject json) throws Exception { + private TcpClientConfig sslClientConfig = new TcpClientConfig(); - new Expectations() { + TcpClientPoolFactory factory = new TcpClientPoolFactory(normalClientConfig, sslClientConfig); + + @Test + public void createClientPool(@Mocked Vertx vertx, @Mocked Context context) { + new Expectations(VertxImpl.class) { { - context.config(); - result = json; - json.getValue("clientOptions"); - result = config; - json.getValue("clientMgr"); - result = clientMgr; + VertxImpl.context(); + result = context; + context.owner(); + result = vertx; } }; + TcpClientConnectionPool pool = factory.createClientPool(); - TcpClientVerticle client = new TcpClientVerticle(); - client.init(vertx, context); - client.start(); - TcpClientConnectionPool pool = client.createClientPool(); - Assert.assertNotNull(pool); + Assert.assertSame(normalClientConfig, pool.netClientWrapper.getClientConfig(false)); + Assert.assertSame(sslClientConfig, pool.netClientWrapper.getClientConfig(true)); } } diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestClientPoolManager.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestClientPoolManager.java deleted file mode 100644 index 4114595af..000000000 --- a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestClientPoolManager.java +++ /dev/null @@ -1,47 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.foundation.vertx.client.tcp; - -import org.junit.Assert; -import org.junit.Before; -import org.junit.Test; -import org.mockito.Mockito; - -import io.servicecomb.foundation.vertx.client.ClientPoolManager; -import io.servicecomb.foundation.vertx.client.NetThreadData; -import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; - -public class TestClientPoolManager { - private ClientPoolManager<HttpClientWithContext> instance; - - @Before - public void setUp() throws Exception { - instance = new ClientPoolManager<>(); - } - - @Test - public void testAddNetThread() { - @SuppressWarnings("unchecked") - NetThreadData<HttpClientWithContext> netThread = Mockito.mock(NetThreadData.class); - instance.addNetThread(netThread); - HttpClientWithContext context = Mockito.mock(HttpClientWithContext.class); - Mockito.when(netThread.selectClientPool()).thenReturn(context); - HttpClientWithContext netThreadValue = instance.findThreadBindClientPool(); - Assert.assertNotNull(netThreadValue); - } -} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestNetClientWrapper.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestNetClientWrapper.java new file mode 100644 index 000000000..689559667 --- /dev/null +++ b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestNetClientWrapper.java @@ -0,0 +1,107 @@ +/* + * 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. + */ +package io.servicecomb.foundation.vertx.client.tcp; + +import java.util.ArrayList; +import java.util.List; + +import org.hamcrest.Matchers; +import org.junit.Assert; +import org.junit.Before; +import org.junit.Test; + +import io.vertx.core.AsyncResult; +import io.vertx.core.Handler; +import io.vertx.core.Vertx; +import io.vertx.core.impl.FutureFactoryImpl; +import io.vertx.core.net.NetClient; +import io.vertx.core.net.NetSocket; +import mockit.Expectations; +import mockit.Mock; +import mockit.MockUp; +import mockit.Mocked; + +public class TestNetClientWrapper { + @Mocked + Vertx vertx; + + @Mocked + TcpClientConfig normalClientConfig; + + @Mocked + NetClient normalNetClient; + + @Mocked + TcpClientConfig sslClientConfig; + + @Mocked + NetClient sslNetClient; + + NetClientWrapper netClientWrapper; + + @Before + public void setup() { + new Expectations() { + { + vertx.createNetClient(normalClientConfig); + result = normalNetClient; + vertx.createNetClient(sslClientConfig); + result = sslNetClient; + } + }; + netClientWrapper = new NetClientWrapper(vertx, normalClientConfig, sslClientConfig); + } + + @Test + public void getClientConfig() { + Assert.assertSame(normalClientConfig, netClientWrapper.getClientConfig(false)); + Assert.assertSame(sslClientConfig, netClientWrapper.getClientConfig(true)); + } + + @Test + public void connect(@Mocked NetSocket normalSocket, @Mocked NetSocket sslSocket) { + int port = 8000; + String host = "localhost"; + + FutureFactoryImpl futureFactory = new FutureFactoryImpl(); + new MockUp<NetClient>

(normalNetClient) { + @Mock + NetClient connect(int port, String host, Handler<AsyncResult<NetSocket>> connectHandler) { + connectHandler.handle(futureFactory.succeededFuture(normalSocket)); + return null; + } + }; + new MockUp<NetClient>(sslNetClient) { + @Mock + NetClient connect(int port, String host, Handler<AsyncResult<NetSocket>> connectHandler) { + connectHandler.handle(futureFactory.succeededFuture(sslSocket)); + return null; + } + }; + + List<NetSocket> socks = new ArrayList<>(); + netClientWrapper.connect(false, port, host, asyncSocket -> { + socks.add(asyncSocket.result()); + }); + netClientWrapper.connect(true, port, host, asyncSocket -> { + socks.add(asyncSocket.result()); + }); + + Assert.assertThat(socks, Matchers.contains(normalSocket, sslSocket)); + } +} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestNetThreadData.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestNetThreadData.java deleted file mode 100644 index 6ecf50ac4..000000000 --- a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestNetThreadData.java +++ /dev/null @@ -1,75 +0,0 @@ -/* - * 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. - */ -package io.servicecomb.foundation.vertx.client.tcp; - -import java.util.concurrent.atomic.AtomicInteger; - -import org.junit.After; -import org.junit.Assert; -import org.junit.Before; -import org.junit.Test; -import org.mockito.Mockito; - -import io.servicecomb.foundation.vertx.client.ClientPoolFactory; -import io.servicecomb.foundation.vertx.client.NetThreadData; -import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; - -/** - * - * - */ -public class TestNetThreadData { - private NetThreadData<HttpClientWithContext> instance; - - @Before - public void setUp() throws Exception { - @SuppressWarnings("unchecked") - ClientPoolFactory<HttpClientWithContext> factory = Mockito.mock(ClientPoolFactory.class); - instance = new NetThreadData<>(factory, 1); - Assert.assertNotNull(instance); - } - - @After - public void tearDown() throws Exception { - instance = null; - } - - @Test - public void testGetFactory() { - ClientPoolFactory<HttpClientWithContext> factory = instance.getFactory(); - Assert.assertNotNull(factory); - } - - @Test - public void testGetPools() { - instance.getPools(); - Assert.assertNotNull(instance.getPools()); - } - - @Test - public void testGetBindIndex() { - AtomicInteger count = instance.getBindIndex(); - Assert.assertNotNull(count); - } - - @Test - public void testSelectClientPool() { - Assert.assertNull(instance.selectClientPool()); - } -} diff --git a/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientConnection.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientConnection.java new file mode 100644 index 000000000..42bd88464 --- /dev/null +++ b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientConnection.java @@ -0,0 +1,249 @@ +/* + * 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. + */ +package io.servicecomb.foundation.vertx.client.tcp; + +import java.util.Map; +import java.util.Queue; +import java.util.concurrent.atomic.AtomicInteger; + +import org.junit.Assert; +import org.junit.Before; +import org.junit.Test; + +import io.netty.buffer.ByteBuf; +import io.netty.buffer.Unpooled; +import io.servicecomb.foundation.vertx.client.tcp.TcpClientConnection.Status; +import io.servicecomb.foundation.vertx.tcp.TcpOutputStream; +import io.vertx.core.AsyncResult; +import io.vertx.core.Context; +import io.vertx.core.Handler; +import io.vertx.core.impl.FutureFactoryImpl; +import io.vertx.core.net.NetSocket; +import io.vertx.core.net.impl.NetSocketImpl; +import mockit.Deencapsulation; +import mockit.Expectations; +import mockit.Mock; +import mockit.MockUp; +import mockit.Mocked; + +public class TestTcpClientConnection { + @Mocked + Context context; + + @Mocked + NetClientWrapper netClientWrapper; + + String strEndpoint = "rest://localhost:8080"; + + TcpClientConnection tcpClientConnection; + + Map<Long, TcpRequest> requestMap; + + Queue<ByteBuf> writeQueue; + + Queue<AbstractTcpClientPackage> packageQueue; + + @Before + public void setup() { + tcpClientConnection = new TcpClientConnection(context, netClientWrapper, strEndpoint); + requestMap = Deencapsulation.getField(tcpClientConnection, "requestMap"); + packageQueue = Deencapsulation.getField(tcpClientConnection, "packageQueue"); + writeQueue = Deencapsulation.getField(tcpClientConnection, "writeQueue"); + } + + @Test + public void localSupportLogin() { + Assert.assertFalse(tcpClientConnection.isLocalSupportLogin()); + + tcpClientConnection.setLocalSupportLogin(true); + Assert.assertTrue(tcpClientConnection.isLocalSupportLogin()); + } + + @Test + public void createLogin() { + Assert.assertNull(tcpClientConnection.createLogin()); + } + + @Test + public void onLoginResponse_buffer() { + Assert.assertTrue(tcpClientConnection.onLoginResponse(null)); + } + + @Test + public void send_inWorkingStatus(@Mocked AbstractTcpClientPackage tcpClientPackage, + @Mocked TcpOutputStream tcpOutputStream) { + Deencapsulation.setField(tcpClientConnection, "status", Status.WORKING); + + long msgId = 1; + ByteBuf byteBuf = Unpooled.buffer(); + new Expectations(tcpClientConnection) { + { + tcpClientPackage.getMsgId(); + result = msgId; + tcpClientPackage.createStream(); + result = tcpOutputStream; + tcpOutputStream.getByteBuf(); + result = byteBuf; + } + }; + tcpClientConnection.send(tcpClientPackage, ar -> { + }); + + Assert.assertSame(byteBuf, writeQueue.poll()); + Assert.assertNull(writeQueue.poll()); + Assert.assertEquals(Status.WORKING, Deencapsulation.getField(tcpClientConnection, "status")); + } + + @Test + public void send_inDisconnectedStatus(@Mocked AbstractTcpClientPackage tcpClientPackage, + @Mocked TcpOutputStream

tcpOutputStream) { + long msgId = 1; + new Expectations(tcpClientConnection) { + { + tcpClientPackage.getMsgId(); + result = msgId; + } + }; + new MockUp<Context>(context) { + @Mock + void runOnContext(Handler<Void> action) { + action.handle(null); + } + }; + tcpClientConnection.send(tcpClientPackage, ar -> { + }); + + Assert.assertSame(tcpClientPackage, packageQueue.poll()); + Assert.assertNull(packageQueue.poll()); + Assert.assertEquals(Status.CONNECTING, Deencapsulation.getField(tcpClientConnection, "status")); + } + + @Test + public void send_disconnectedToTryLogin(@Mocked AbstractTcpClientPackage tcpClientPackage, + @Mocked TcpOutputStream tcpOutputStream) { + long msgId = 1; + new Expectations(tcpClientConnection) { + { + tcpClientPackage.getMsgId(); + result = msgId; + } + }; + new MockUp<Context>(context) { + @Mock + void runOnContext(Handler<Void> action) { + Deencapsulation.setField(tcpClientConnection, "status", Status.TRY_LOGIN); + action.handle(null); + } + }; + tcpClientConnection.send(tcpClientPackage, ar -> { + }); + + Assert.assertSame(tcpClientPackage, packageQueue.poll()); + Assert.assertNull(packageQueue.poll()); + Assert.assertEquals(Status.TRY_LOGIN, Deencapsulation.getField(tcpClientConnection, "status")); + } + + @Test + public void send_disconnectedToWorking(@Mocked AbstractTcpClientPackage tcpClientPackage, + @Mocked TcpOutputStream tcpOutputStream) { + long msgId = 1; + new Expectations(tcpClientConnection) { + { + tcpClientPackage.getMsgId(); + result = msgId; + } + }; + new MockUp<Context>(context) { + @Mock + void runOnContext(Handler<Void> action) { + Deencapsulation.setField(tcpClientConnection, "status", Status.WORKING); + action.handle(null); + } + }; + tcpClientConnection.send(tcpClientPackage, ar -> { + }); + + Assert.assertNull(writeQueue.poll()); + Assert.assertNull(packageQueue.poll()); + Assert.assertEquals(Status.WORKING, Deencapsulation.getField(tcpClientConnection, "status")); + } + + @Test + public void connect_success(@Mocked NetSocketImpl netSocket) { + FutureFactoryImpl futureFactory = new FutureFactoryImpl(); + new MockUp<NetClientWrapper>(netClientWrapper) { + @Mock + void connect(boolean ssl, int port, String host, Handler<AsyncResult<NetSocket>> connectHandler) { + connectHandler.handle(futureFactory.succeededFuture(netSocket)); + } + }; + + tcpClientConnection.connect(); + + Assert.assertSame(netSocket, tcpClientConnection.getNetSocket()); + Assert.assertEquals(Status.WORKING, Deencapsulation.getField(tcpClientConnection, "status")); + } + + @Test + public void connect_failed() { + requestMap.put(10L, new TcpRequest(10, ar -> { + })); + + FutureFactoryImpl futureFactory = new FutureFactoryImpl(); + Error error = new Error(); + new MockUp<NetClientWrapper>(netClientWrapper) { + @Mock + void connect(boolean ssl, int port, String host, Handler<AsyncResult<NetSocket>> connectHandler) { + connectHandler.handle(futureFactory.failedFuture(error)); + } + }; + + tcpClientConnection.connect(); + + Assert.assertEquals(Status.DISCONNECTED, Deencapsulation.getField(tcpClientConnection, "status")); + Assert.assertEquals(0, requestMap.size()); + } + + @Test + public void onClosed(@Mocked NetSocketImpl netSocket) { + requestMap.put(10L, new TcpRequest(10, ar -> { + })); + tcpClientConnection.initNetSocket(netSocket); + + Deencapsulation.invoke(tcpClientConnection, "onClosed", new Class<?>[] {Void.class}, new Object[] {null}); + Assert.assertEquals(Status.DISCONNECTED, Deencapsulation.getField(tcpClientConnection, "status")); + Assert.assertEquals(0, requestMap.size()); + } + + @Test + public void onReply_notExist() { + // should not throw exception + tcpClientConnection.onReply(1, null, null); + } + + @Test + public void on_exist() { + long msgId = 1L; + AtomicInteger count = new AtomicInteger(); + requestMap.put(msgId, new TcpRequest(10, ar -> { + count.incrementAndGet(); + })); + + tcpClientConnection.onReply(msgId, null, null); + Assert.assertEquals(1, count.get()); + } +} diff --git a/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayClientManager.java b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientConnectionPool.java similarity index 61% rename from transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayClientManager.java rename to foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientConnectionPool.java index d1781a95b..930c0fc2b 100644 --- a/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayClientManager.java +++ b/foundations/foundation-vertx/src/test/java/io/servicecomb/foundation/vertx/client/tcp/TestTcpClientConnectionPool.java @@ -14,21 +14,32 @@ * See the License for the specific language governing permissions and * limitations under the License. */ +package io.servicecomb.foundation.vertx.client.tcp; -package io.servicecomb.transport.highway; - +import org.hamcrest.Matchers; import org.junit.Assert; +import org.junit.Before; import org.junit.Test; - -public class TestHighwayClientManager { - @Test - public void testRestTransportClientManager() { - HighwayClient client1 = HighwayClientManager.INSTANCE.getHighwayClient(false); - HighwayClient client2 = HighwayClientManager.INSTANCE.getHighwayClient(false); - Assert.assertEquals(client1, client2); +import io.vertx.core.Context; +import mockit.Mocked; + +public class TestTcpClientConnectionPool { + @Mocked + Context context; + + @Mocked + NetClientWrapper netClientWrapper; - HighwayClient client3 = HighwayClientManager.INSTANCE.getHighwayClient(true); - HighwayClient client4 = HighwayClientManager.INSTANCE.getHighwayClient(true); - Assert.assertEquals(client3, client4); + TcpClientConnectionPool pool; + + @Before + public void setup() { + pool = new TcpClientConnectionPool(context, netClientWrapper); + } + + @Test + public void create() { + Assert.assertThat(pool.create("rest://localhost:8765"), Matchers.instanceOf(TcpClientConnection.class)); } } diff --git a/integration-tests/tracing-tests/src/test/resources/microservice.yaml b/integration-tests/tracing-tests/src/test/resources/microservice.yaml index 775e908d6..ab8a3535b 100644 --- a/integration-tests/tracing-tests/src/test/resources/microservice.yaml +++ b/integration-tests/tracing-tests/src/test/resources/microservice.yaml @@ -38,3 +38,6 @@ servicecomb: tracing: collector: address: http://localhost:9411 + executor: + default: + thread-per-group: 10 \ No newline at end of file diff --git a/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/AbstractClientPool.java b/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/AbstractClientPool.java index eb064c9c6..e93ba5455 100644 --- a/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/AbstractClientPool.java +++ b/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/AbstractClientPool.java @@ -20,13 +20,10 @@ import org.slf4j.Logger; import org.slf4j.LoggerFactory; -import io.servicecomb.foundation.ssl.SSLCustom; -import io.servicecomb.foundation.ssl.SSLOption; -import io.servicecomb.foundation.ssl.SSLOptionFactory; import

io.servicecomb.foundation.vertx.VertxTLSBuilder; import io.servicecomb.foundation.vertx.VertxUtils; import io.servicecomb.foundation.vertx.client.ClientPoolManager; -import io.servicecomb.foundation.vertx.client.http.HttpClientVerticle; +import io.servicecomb.foundation.vertx.client.ClientVerticle; +import io.servicecomb.foundation.vertx.client.http.HttpClientPoolFactory; import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; import io.servicecomb.serviceregistry.config.ServiceRegistryConfig; import io.vertx.core.DeploymentOptions; @@ -37,12 +34,11 @@ * Created by on 2017/4/28. */ public abstract class AbstractClientPool implements ClientPool { - private static final Logger LOGGER = LoggerFactory.getLogger(AbstractClientPool.class); - private ClientPoolManager<HttpClientWithContext> clientMgr = new ClientPoolManager<>(); + protected static final String SSL_KEY = "sc.consumer"; - private static final String SSL_KEY = "sc.consumer"; + private ClientPoolManager<HttpClientWithContext> clientMgr; public AbstractClientPool() { create(); @@ -56,28 +52,15 @@ public AbstractClientPool() { public void create() { // 这里面是同步接口，且好像直接在事件线程中用，保险起见，先使用独立的vertx实例 Vertx vertx = VertxUtils.getOrCreateVertxByName("registry", null); HttpClientOptions httpClientOptions = createHttpClientOptions(); + clientMgr = new ClientPoolManager<>(vertx, new HttpClientPoolFactory(httpClientOptions)); + DeploymentOptions deployOptions = VertxUtils.createClientDeployOptions(this.clientMgr, - ServiceRegistryConfig.INSTANCE.getWorkerPoolSize(), - 1, - httpClientOptions); + ServiceRegistryConfig.INSTANCE.getWorkerPoolSize()); try { - VertxUtils.blockDeploy(vertx, HttpClientVerticle.class, deployOptions); + VertxUtils.blockDeploy(vertx, ClientVerticle.class, deployOptions); } catch (InterruptedException e) { LOGGER.error("deploy a registry verticle failed, {}", e.getMessage()); } } - - protected void buildSecureClientOptions(HttpClientOptions httpClientOptions) { - SSLOptionFactory factory = - SSLOptionFactory.createSSLOptionFactory(SSL_KEY, null); - SSLOption sslOption; - if (factory == null) { - sslOption = SSLOption.buildFromYaml(SSL_KEY); - } else { - sslOption = factory.createSSLOption(); - } - SSLCustom sslCustom = SSLCustom.createSSLCustom(sslOption.getSslCustomClass()); - VertxTLSBuilder.buildHttpClientOptions(sslOption, sslCustom, httpClientOptions); - } } diff --git a/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/HttpClientPool.java b/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/HttpClientPool.java index d63eeeda5..d573dd724 100644 --- a/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/HttpClientPool.java +++ b/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/HttpClientPool.java @@ -20,6 +20,7 @@ import org.slf4j.Logger; import org.slf4j.LoggerFactory; +import io.servicecomb.foundation.vertx.VertxTLSBuilder; import io.servicecomb.serviceregistry.config.ServiceRegistryConfig; import io.vertx.core.http.HttpClientOptions; import io.vertx.core.http.HttpVersion; @@ -58,7 +59,7 @@ public HttpClientOptions createHttpClientOptions() { } if (ServiceRegistryConfig.INSTANCE.isSsl()) { LOGGER.debug("service center client performs requests over TLS"); - buildSecureClientOptions(httpClientOptions); + VertxTLSBuilder.buildHttpClientOptions(SSL_KEY, httpClientOptions); } return httpClientOptions; } diff --git a/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/WebsocketClientPool.java b/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/WebsocketClientPool.java index 4e7df62ec..5c974eb1c 100644 --- a/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/WebsocketClientPool.java +++ b/service-registry/src/main/java/io/servicecomb/serviceregistry/client/http/WebsocketClientPool.java @@ -20,6 +20,7 @@ import org.slf4j.Logger; import org.slf4j.LoggerFactory; +import io.servicecomb.foundation.vertx.VertxTLSBuilder; import io.servicecomb.serviceregistry.config.ServiceRegistryConfig; import io.vertx.core.http.HttpClientOptions; import io.vertx.core.http.HttpVersion; @@ -49,7 +50,7 @@ public HttpClientOptions createHttpClientOptions() { } if (ServiceRegistryConfig.INSTANCE.isSsl()) { LOGGER.debug("service center ws client performs requests over TLS"); - buildSecureClientOptions(httpClientOptions); + VertxTLSBuilder.buildHttpClientOptions(SSL_KEY, httpClientOptions); } return httpClientOptions; } diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClient.java b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClient.java index a2be8f2cb..42f3b30d6 100644 --- a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClient.java +++ b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClient.java @@ -33,6 +33,7 @@ import io.servicecomb.foundation.vertx.VertxTLSBuilder; import io.servicecomb.foundation.vertx.VertxUtils; import io.servicecomb.foundation.vertx.client.ClientPoolManager; +import io.servicecomb.foundation.vertx.client.ClientVerticle; import io.servicecomb.foundation.vertx.client.tcp.TcpClientConfig; import io.servicecomb.swagger.invocation.AsyncResponse; import io.servicecomb.swagger.invocation.Response; @@ -40,27 +41,24 @@ import io.vertx.core.Vertx; public class HighwayClient { - private static final Logger log = LoggerFactory.getLogger(HighwayClient.class); + private static final Logger LOGGER = LoggerFactory.getLogger(HighwayClient.class); private static final String SSL_KEY = "highway.consumer"; - private ClientPoolManager<HighwayClientConnectionPool> clientMgr = new ClientPoolManager<>(); + private ClientPoolManager<HighwayClientConnectionPool> clientMgr; - private final boolean sslEnabled; + public void init(Vertx vertx) throws Exception { + TcpClientConfig normalConfig = createTcpClientConfig(); + normalConfig.setSsl(false); - public HighwayClient(boolean sslEnabled) { - this.sslEnabled = sslEnabled; - } + TcpClientConfig sslConfig = createTcpClientConfig(); + sslConfig.setSsl(true); - public void init(Vertx vertx) throws Exception { - TcpClientConfig config = createTcpClientConfig(); + clientMgr = new ClientPoolManager<>(vertx, new HighwayClientPoolFactory(normalConfig, sslConfig)); DeploymentOptions deployOptions = VertxUtils.createClientDeployOptions(clientMgr, - HighwayConfig.getClientThreadCount(), - HighwayConfig.getClientConnectionPoolPerThread(), - config); - - VertxUtils.blockDeploy(vertx, HighwayClientVerticle.class, deployOptions); + HighwayConfig.getClientThreadCount()); + VertxUtils.blockDeploy(vertx, ClientVerticle.class, deployOptions); } private TcpClientConfig createTcpClientConfig() { @@ -73,31 +71,33 @@ public void run() { }); tcpClientConfig.setRequestTimeoutMillis(prop.get()); - if (this.sslEnabled) { - SSLOptionFactory factory = -

SSLOptionFactory.createSSLOptionFactory(SSL_KEY, null); - SSLOption sslOption; - if (factory == null) { - sslOption = SSLOption.buildFromYaml(SSL_KEY); - } else { - sslOption = factory.createSSLOption(); - } - SSLCustom sslCustom = SSLCustom.createSSLCustom(sslOption.getSslCustomClass()); - VertxTLSBuilder.buildClientOptionsBase(sslOption, sslCustom, tcpClientConfig); + SSLOptionFactory factory = + SSLOptionFactory.createSSLOptionFactory(SSL_KEY, null); + SSLOption sslOption; + if (factory == null) { + sslOption = SSLOption.buildFromYaml(SSL_KEY); + } else { + sslOption = factory.createSSLOption(); } + SSLCustom sslCustom = SSLCustom.createSSLCustom(sslOption.getSslCustomClass()); + VertxTLSBuilder.buildClientOptionsBase(sslOption, sslCustom, tcpClientConfig); + return tcpClientConfig; } public void send(Invocation invocation, AsyncResponse asyncResp) throws Exception { - HighwayClientConnectionPool tcpClientPool = clientMgr.findThreadBindClientPool(); + HighwayClientConnectionPool tcpClientPool = clientMgr.findClientPool(invocation.isSync()); OperationMeta operationMeta = invocation.getOperationMeta(); OperationProtobuf operationProtobuf = ProtobufManager.getOrCreateOperation(operationMeta); - HighwayClientConnection tcpClient = tcpClientPool.findOrCreateClient(invocation.getEndpoint().getEndpoint()); + HighwayClientConnection tcpClient = + tcpClientPool.findOrCreateClient(invocation.getEndpoint().getEndpoint()); HighwayClientPackage clientPackage = new HighwayClientPackage(invocation, operationProtobuf, tcpClient); - log.debug("Calling method {} of {} by highway", operationMeta.getMethod(), invocation.getMicroserviceName()); - tcpClientPool.send(tcpClient, clientPackage, ar -> { + LOGGER.debug("Sending request by highway, qualifiedName={}, endpoint={}.", + invocation.getMicroserviceQualifiedName(), + invocation.getEndpoint().getEndpoint()); + tcpClient.send(clientPackage, ar -> { // 此时是在网络线程中，转换线程 invocation.getResponseExecutor().execute(() -> { if (ar.failed()) { diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnection.java b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnection.java index 9a495fad3..af58ccade 100644 --- a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnection.java +++ b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnection.java @@ -21,7 +21,7 @@ import io.protostuff.runtime.ProtobufFeature; import io.servicecomb.foundation.vertx.client.tcp.AbstractTcpClientPackage; -import io.servicecomb.foundation.vertx.client.tcp.TcpClientConfig; +import io.servicecomb.foundation.vertx.client.tcp.NetClientWrapper; import io.servicecomb.foundation.vertx.client.tcp.TcpClientConnection; import io.servicecomb.foundation.vertx.tcp.TcpOutputStream; import io.servicecomb.transport.highway.message.LoginRequest; @@ -29,16 +29,14 @@ import io.servicecomb.transport.highway.message.RequestHeader; import io.vertx.core.Context; import io.vertx.core.buffer.Buffer; -import io.vertx.core.net.NetClient; public class HighwayClientConnection extends TcpClientConnection { private static final Logger LOGGER = LoggerFactory.getLogger(HighwayClientConnection.class); private ProtobufFeature protobufFeature = new ProtobufFeature(); - public HighwayClientConnection(Context context, NetClient netClient, String endpoint, - TcpClientConfig clientConfig) { - super(context, netClient, endpoint, clientConfig); + public HighwayClientConnection(Context context, NetClientWrapper netClientWrapper, String endpoint) { + super(context, netClientWrapper, endpoint); setLocalSupportLogin(true); } diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnectionPool.java b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnectionPool.java index 7405a0760..93d2d0bae 100644 --- a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnectionPool.java +++ b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientConnectionPool.java @@ -17,17 +17,16 @@ package io.servicecomb.transport.highway; import io.servicecomb.foundation.vertx.client.tcp.AbstractTcpClientConnectionPool; -import io.servicecomb.foundation.vertx.client.tcp.TcpClientConfig; +import io.servicecomb.foundation.vertx.client.tcp.NetClientWrapper; import io.vertx.core.Context; -import io.vertx.core.net.NetClient; public class HighwayClientConnectionPool extends AbstractTcpClientConnectionPool<HighwayClientConnection> { - public HighwayClientConnectionPool(TcpClientConfig clientConfig, Context context, NetClient netClient) { - super(clientConfig, context, netClient); + public HighwayClientConnectionPool(Context context, NetClientWrapper netClientWrapper) { + super(context, netClientWrapper); } @Override protected HighwayClientConnection create(String endpoint) { - return new HighwayClientConnection(context, netClient, endpoint, clientConfig); + return new HighwayClientConnection(context, netClientWrapper, endpoint); } } diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientManager.java b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientManager.java deleted file mode 100644 index a379d6506..000000000 --- a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientManager.java +++ /dev/null @@ -1,73 +0,0 @@ -/* - * 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. - */ - -package io.servicecomb.transport.highway; - -import org.slf4j.Logger; -import org.slf4j.LoggerFactory; - -import io.servicecomb.foundation.vertx.VertxUtils; -import io.vertx.core.Vertx; - -public final class HighwayClientManager { - private static final Logger LOGGER = LoggerFactory.getLogger(HighwayClientManager.class); - - public static final HighwayClientManager INSTANCE = new HighwayClientManager(); - - // same instance in AbstractTranport. need refactor in future. - private final Vertx transportVertx = VertxUtils.getOrCreateVertxByName("transport", null); - - private static final

Object LOCK = new Object(); - - private volatile HighwayClient sslClient = null; - - private volatile HighwayClient nonSslCient = null; - - private HighwayClientManager() { - } - - public HighwayClient getHighwayClient(boolean sslEnabled) { - try { - if (sslEnabled) { - if (sslClient == null) { - synchronized (LOCK) { - if (sslClient == null) { - HighwayClient client = new HighwayClient(true); - client.init(transportVertx); - sslClient = client; - } - } - } - return sslClient; - } else { - if (nonSslCient == null) { - synchronized (LOCK) { - if (nonSslCient == null) { - HighwayClient client = new HighwayClient(false); - client.init(transportVertx); - nonSslCient = client; - } - } - } - return nonSslCient; - } - } catch (Exception e) { - LOGGER.error(""); - throw new IllegalStateException("init rest client transport failed."); - } - } -} diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientPoolFactory.java b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientPoolFactory.java new file mode 100644 index 000000000..389fd1df3 --- /dev/null +++ b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayClientPoolFactory.java @@ -0,0 +1,33 @@ +/* + * 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. + */ +package io.servicecomb.transport.highway; + +import io.servicecomb.foundation.vertx.client.tcp.AbstractTcpClientPoolFactory; +import io.servicecomb.foundation.vertx.client.tcp.NetClientWrapper; +import io.servicecomb.foundation.vertx.client.tcp.TcpClientConfig; +import io.vertx.core.Context; + +public class HighwayClientPoolFactory extends AbstractTcpClientPoolFactory<HighwayClientConnectionPool> { + public HighwayClientPoolFactory(TcpClientConfig normalClientConfig, TcpClientConfig sslClientConfig) { + super(normalClientConfig, sslClientConfig); + } + + @Override + protected HighwayClientConnectionPool doCreateClientPool(Context context, NetClientWrapper netClientWrapper) { + return new HighwayClientConnectionPool(context, netClientWrapper); + } +} diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayConfig.java b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayConfig.java index 93bfff342..ba1197072 100644 --- a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayConfig.java +++ b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayConfig.java @@ -42,10 +42,4 @@ public static int getClientThreadCount() { DynamicPropertyFactory.getInstance().getIntProperty("cse.highway.client.thread-count", 1); return address.get(); } - - public static int getClientConnectionPoolPerThread() { - DynamicIntProperty address = - DynamicPropertyFactory.getInstance().getIntProperty("cse.highway.client.connection-pool-per-thread", 1); - return address.get(); - } } diff --git a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayTransport.java b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayTransport.java index b72e52d46..a34e3c104 100644 --- a/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayTransport.java +++ b/transports/transport-highway/src/main/java/io/servicecomb/transport/highway/HighwayTransport.java @@ -19,13 +19,10 @@ import java.util.Collections; -import org.slf4j.Logger; -import org.slf4j.LoggerFactory; import org.springframework.stereotype.Component; import io.servicecomb.core.Invocation; import io.servicecomb.core.transport.AbstractTransport; -import io.servicecomb.foundation.common.net.URIEndpointObject; import io.servicecomb.foundation.vertx.SimpleJsonObject; import io.servicecomb.foundation.vertx.VertxUtils; import io.servicecomb.foundation.vertx.tcp.TcpConst; @@ -34,17 +31,18 @@ @Component public class HighwayTransport extends AbstractTransport { - private static final Logger log = LoggerFactory.getLogger(HighwayTransport.class); - public static final String NAME = "highway"; + private HighwayClient highwayClient = new HighwayClient(); + @Override public String getName() { return NAME; } public boolean init() throws Exception { + highwayClient.init(transportVertx); HighwayCodec.setHighwayTransport(this); DeploymentOptions deployOptions = new DeploymentOptions().setInstances(HighwayConfig.getServerThreadCount()); @@ -52,20 +50,11 @@ public boolean init() throws Exception { SimpleJsonObject json = new SimpleJsonObject(); json.put(ENDPOINT_KEY, getEndpoint()); deployOptions.setConfig(json); - return VertxUtils.blockDeploy(transportVertx, HighwayServerVerticle.class, deployOptions) && deployClient(); - } - - private boolean deployClient() { - return HighwayClientManager.INSTANCE.getHighwayClient(true) != null && - HighwayClientManager.INSTANCE.getHighwayClient(false) != null; + return VertxUtils.blockDeploy(transportVertx, HighwayServerVerticle.class, deployOptions); } @Override public void send(Invocation invocation, AsyncResponse asyncResp) throws Exception { - URIEndpointObject endpoint = (URIEndpointObject) invocation.getEndpoint().getAddress(); - HighwayClient client = - HighwayClientManager.INSTANCE.getHighwayClient(endpoint.isSslEnabled()); - log.debug("Sending request by highway to endpoint {}:{}", endpoint.getHostOrIp(), endpoint.getPort()); - client.send(invocation, asyncResp); + highwayClient.send(invocation, asyncResp); } } diff --git a/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayClient.java b/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayClient.java index bc7b62677..86986726b 100644 --- a/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayClient.java +++ b/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayClient.java @@ -17,7 +17,8 @@ package io.servicecomb.transport.highway; -import java.util.concurrent.Executor; +import javax.ws.rs.core.Response.Status; +import javax.xml.ws.Holder; import org.apache.commons.configuration.AbstractConfiguration; import org.junit.Assert; @@ -29,48 +30,49 @@ import io.netty.buffer.ByteBuf; import io.protostuff.runtime.ProtobufCompatibleUtils; +import io.protostuff.runtime.ProtobufFeature; import io.servicecomb.codec.protobuf.definition.OperationProtobuf; import io.servicecomb.codec.protobuf.definition.ProtobufManager; import io.servicecomb.config.ConfigUtil; import

io.servicecomb.core.Endpoint; import io.servicecomb.core.Invocation; import
io.servicecomb.core.definition.OperationMeta; +import io.servicecomb.core.executor.ReactiveExecutor; import
io.servicecomb.core.transport.AbstractTransport; import io.servicecomb.foundation.vertx.VertxUtils; import
io.servicecomb.foundation.vertx.client.ClientPoolManager; -import
io.servicecomb.foundation.vertx.client.tcp.TcpClientConfig; +import
io.servicecomb.foundation.vertx.client.tcp.AbstractTcpClientPackage; +import
io.servicecomb.foundation.vertx.client.tcp.NetClientWrapper; +import io.servicecomb.foundation.vertx.client.tcp.TcpData;
+import io.servicecomb.foundation.vertx.client.tcp.TcpResponseCallback; import
io.servicecomb.foundation.vertx.server.TcpParser; import io.servicecomb.foundation.vertx.tcp.TcpOutputStream; -import
io.servicecomb.swagger.invocation.AsyncResponse; +import io.servicecomb.swagger.invocation.Response; +import
io.servicecomb.swagger.invocation.exception.InvocationException; import
io.servicecomb.transport.highway.message.LoginRequest; import
io.servicecomb.transport.highway.message.RequestHeader; import io.vertx.core.AbstractVerticle; import
io.vertx.core.DeploymentOptions; import io.vertx.core.Vertx; import io.vertx.core.buffer.Buffer; +import
mockit.Deencapsulation; import mockit.Mock; import mockit.MockUp; import mockit.Mocked; public class
TestHighwayClient { - private TcpClientConfig options; - private static final String REQUEST_TIMEOUT_KEY =
"cse.request.timeout"; + private static final String REQUEST_TIMEOUT_KEY = "cse.request.timeout"; - HighwayClient
client = new HighwayClient(true); + HighwayClient client = new HighwayClient(); Invocation invocation =
Mockito.mock(Invocation.class); - AsyncResponse asyncResp = Mockito.mock(AsyncResponse.class); - OperationProtobuf
operationProtobuf = Mockito.mock(OperationProtobuf.class); OperationMeta operationMeta =
Mockito.mock(OperationMeta.class); Endpoint endpoint = Mockito.mock(Endpoint.class); - Executor excutor =
Mockito.mock(Executor.class); - - @BeforeClass public static void beforeCls() { ConfigUtil.installDynamicConfig(); @@
-89,14 +91,22 @@ public void testRequestTimeout() { public void testHighwayClientSSL(@Mocked Vertx vertx) throws
Exception { new MockUp<VertxUtils>() { @Mock - <CLIENT_POOL, CLIENT_OPTIONS> DeploymentOptions
createClientDeployOptions(- ClientPoolManager<CLIENT_POOL> clientMgr, - int instanceCount, - int
poolCountPerVerticle, CLIENT_OPTIONS clientOptions) { - options = (TcpClientConfig) clientOptions; - return null; +
<VERTICLE extends AbstractVerticle> boolean blockDeploy(Vertx vertx, + Class<VERTICLE> cls, + DeploymentOptions
options) throws InterruptedException { + return true; } + }; + + client.init(vertx); + +
ClientPoolManager<HighwayClientConnectionPool> clientMgr = Deencapsulation.getField(client, "clientMgr"); +
Assert.assertSame(vertx, Deencapsulation.getField(clientMgr, "vertx")); + } + private Object doTestSend(Vertx vertx,
HighwayClientConnectionPool pool, HighwayClientConnection tcpClient, + Object decodedResponse) throws Exception {
+ new MockUp<VertxUtils>() { @Mock <VERTICLE extends AbstractVerticle> boolean blockDeploy(Vertx vertx,
Class<VERTICLE> cls, @@ -105,34 +115,114 @@ public void testHighwayClientSSL(@Mocked Vertx vertx) throws
Exception { } }; - client.init(vertx); - Assert.assertEquals(options.isSsl(), true); - } + new
MockUp<ClientPoolManager<HighwayClientConnectionPool>>() { + @Mock + public HighwayClientConnectionPool
findClientPool(boolean sync) { + return pool; + } + }; - @Test - public void testSend() { - boolean status = true; -
mockUps(); + new MockUp<ProtobufManager>() { + @Mock + public OperationProtobuf
getOrCreateOperation(OperationMeta operationMeta) throws Exception { + return operationProtobuf; + } + }; + + new
MockUp<HighwayClientConnectionPool>() { + @Mock + HighwayClientConnection findOrCreateClient(String endpoint)
{ + return tcpClient; + } + }; + + new MockUp<HighwayCodec>() { + @Mock + public Buffer encodeRequest(Invocation
invocation, OperationProtobuf operationProtobuf, + long msgId) throws Exception { + return null; + } + + @Mock +
Response decodeResponse(Invocation invocation, OperationProtobuf operationProtobuf, + TcpData tcpData,
ProtobufFeature protobufFeature) throws Throwable { + if (Response.class.isInstance(decodedResponse)) { + return
(Response) decodedResponse; + } + + throw (Throwable) decodedResponse; + } + }; + + client.init(vertx);
Mockito.when(invocation.getOperationMeta()).thenReturn(operationMeta);
Mockito.when(invocation.getEndpoint()).thenReturn(endpoint);
Mockito.when(invocation.getEndpoint().getEndpoint()).thenReturn("endpoint"); -
Mockito.when(invocation.getResponseExecutor()).thenReturn(excutor); - - try { - client.send(invocation, asyncResp); - }
catch (Exception e) { - status = false; - } - Assert.assertTrue(status); +
Mockito.when(invocation.getResponseExecutor()).thenReturn(new ReactiveExecutor()); + + Holder<Object> result = new
Holder<>(); + client.send(invocation, ar -> { + result.value = ar.getResult(); + }); + + return result.value; } @Test - public
void testCreateLogin() throws Exception { + public void testSend_success(@Mocked Vertx vertx, @Mocked
HighwayClientConnectionPool pool, + @Mocked HighwayClientConnection tcpClient) throws Exception { + new
MockUp<HighwayClientConnection>() { + @Mock + void send(AbstractTcpClientPackage tcpClientPackage,
TcpResponseCallback callback) { + callback.success(null); + } + }; + + Object result = doTestSend(vertx, pool, tcpClient,
Response.ok("ok")); + + Assert.assertEquals("ok", result); + } + + @Test + public void
testSend_success_decode_failed(@Mocked Vertx vertx, @Mocked HighwayClientConnectionPool pool, + @Mocked
HighwayClientConnection tcpClient) throws Exception { + new MockUp<HighwayClientConnection>() { + @Mock + void
send(AbstractTcpClientPackage tcpClientPackage, TcpResponseCallback callback) { + callback.success(null); + } + }; + +
Object result = doTestSend(vertx, pool, tcpClient, new InvocationException(Status.BAD_REQUEST, (Object) "failed")); +
+ Assert.assertEquals("failed", ((InvocationException) result).getErrorData()); + } + + @Test + public void
testSend_failed(@Mocked Vertx vertx, @Mocked HighwayClientConnectionPool pool, + @Mocked
HighwayClientConnection tcpClient) throws Exception { + new MockUp<HighwayClientConnection>() { + @Mock + void
send(AbstractTcpClientPackage tcpClientPackage, TcpResponseCallback callback) { + callback.fail(new
InvocationException(Status.BAD_REQUEST, (Object) "failed")); + } + }; + + Object result = doTestSend(vertx, + pool, +
tcpClient, + null); + + Assert.assertEquals("failed", ((InvocationException) result).getErrorData()); + } + + @Test + public
void testCreateLogin(@Mocked NetClientWrapper netClientWrapper) throws Exception { ProtobufCompatibleUtils.init();
HighwayClientConnection connection = - new HighwayClientConnection(null, null, "highway://127.0.0.1:7890", null); +
new HighwayClientConnection(null, netClientWrapper, "highway://127.0.0.1:7890"); TcpOutputStream os =

connection.createLogin(); ByteBuf buf = os.getBuffer().getByteBuf(); @@ -156,30 +246,4 @@ public void testCreateLogin() throws Exception { LoginRequest login = LoginRequest.readObject(bodyBuffer); Assert.assertEquals(HighwayTransport.NAME, login.getProtocol()); } - - private void mockUps() { - - new MockUp<ClientPoolManager<HighwayClientConnectionPool>>() { - - @Mock - public HighwayClientConnectionPool findThreadBindClientPool() { - return Mockito.mock(HighwayClientConnectionPool.class); - } - }; - - new MockUp<ProtobufManager>() { - @Mock - public OperationProtobuf getOrCreateOperation(OperationMeta operationMeta) throws Exception { - return operationProtobuf; - } - }; - - new MockUp<HighwayCodec>() { - @Mock - public Buffer encodeRequest(Invocation invocation, OperationProtobuf operationProtobuf, - long msgId) throws Exception { - return null; - } - }; - } } diff --git a/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayConfig.java b/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayConfig.java index 0dfffea50..26b064399 100644 --- a/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayConfig.java +++ b/transports/transport-highway/src/test/java/io/servicecomb/transport/highway/TestHighwayConfig.java @@ -22,18 +22,17 @@ public class TestHighwayConfig { @Test - public void testGetThreadCount() { + public void getServerThreadCount() { Assert.assertEquals(HighwayConfig.getServerThreadCount(), 1); - Assert.assertEquals(HighwayConfig.getClientThreadCount(), 1); } @Test - public void testGetAddress() { - Assert.assertEquals(HighwayConfig.getAddress(), null); + public void getClientThreadCount() { + Assert.assertEquals(HighwayConfig.getClientThreadCount(), 1); } @Test - public void testGetConnectionPoolPerThread() { - Assert.assertEquals(HighwayConfig.getClientConnectionPoolPerThread(), 1); + public void getAddress() { + Assert.assertEquals(HighwayConfig.getAddress(), null); } } diff --git a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClient.java b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClient.java index 7b88060e5..d239e709d 100644 --- a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClient.java +++ b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClient.java @@ -21,13 +21,11 @@ import org.slf4j.LoggerFactory; import io.servicecomb.core.Invocation; -import io.servicecomb.foundation.ssl.SSLCustom; -import io.servicecomb.foundation.ssl.SSLOption; -import io.servicecomb.foundation.ssl.SSLOptionFactory; import io.servicecomb.foundation.vertx.VertxTLSBuilder; import io.servicecomb.foundation.vertx.VertxUtils; import io.servicecomb.foundation.vertx.client.ClientPoolManager; -import io.servicecomb.foundation.vertx.client.http.HttpClientVerticle; +import io.servicecomb.foundation.vertx.client.ClientVerticle; +import io.servicecomb.foundation.vertx.client.http.HttpClientPoolFactory; import io.servicecomb.foundation.vertx.client.http.HttpClientWithContext; import io.servicecomb.swagger.invocation.AsyncResponse; import io.servicecomb.transport.rest.client.http.VertxHttpMethod; @@ -40,46 +38,30 @@ private static final String SSL_KEY = "rest.consumer"; - private ClientPoolManager<HttpClientWithContext> clientMgr = new ClientPoolManager<>(); + private ClientPoolManager<HttpClientWithContext> clientMgr; - private final boolean sslEnabled; - - public RestTransportClient(boolean sslEnabled) { - this.sslEnabled = sslEnabled; - } public void init(Vertx vertx) throws Exception { HttpClientOptions httpClientOptions = createHttpClientOptions(); + clientMgr = new ClientPoolManager<>(vertx, new HttpClientPoolFactory(httpClientOptions)); + DeploymentOptions deployOptions = VertxUtils.createClientDeployOptions(clientMgr, - TransportClientConfig.getThreadCount(), - TransportClientConfig.getConnectionPoolPerThread(), - httpClientOptions); - VertxUtils.blockDeploy(vertx, HttpClientVerticle.class, deployOptions); + TransportClientConfig.getThreadCount()); + VertxUtils.blockDeploy(vertx, ClientVerticle.class, deployOptions); } - private HttpClientOptions createHttpClientOptions() { + private static HttpClientOptions createHttpClientOptions() { HttpClientOptions httpClientOptions = new HttpClientOptions(); httpClientOptions.setMaxPoolSize(TransportClientConfig.getConnectionMaxPoolSize()); httpClientOptions.setIdleTimeout(TransportClientConfig.getConnectionIdleTimeoutInSeconds()); httpClientOptions.setKeepAlive(TransportClientConfig.getConnectionKeepAlive()); - if (this.sslEnabled) { - SSLOptionFactory factory = - SSLOptionFactory.createSSLOptionFactory(SSL_KEY, - null); - SSLOption sslOption; - if (factory == null) { - sslOption = SSLOption.buildFromYaml(SSL_KEY); - } else { - sslOption = factory.createSSLOption(); - } - SSLCustom sslCustom = SSLCustom.createSSLCustom(sslOption.getSslCustomClass()); - VertxTLSBuilder.buildHttpClientOptions(sslOption, sslCustom, httpClientOptions); - } + + VertxTLSBuilder.buildHttpClientOptions(SSL_KEY, httpClientOptions); return httpClientOptions; } public void send(Invocation invocation, AsyncResponse asyncResp) throws Exception { - HttpClientWithContext httpClientWithContext = clientMgr.findThreadBindClientPool(); + HttpClientWithContext httpClientWithContext = clientMgr.findClientPool(invocation.isSync()); try { VertxHttpMethod.INSTANCE.doMethod(httpClientWithContext, invocation, asyncResp); } catch (Exception e) { diff --git a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClientManager.java b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClientManager.java index bf34ee00e..4788b8154 100644 --- a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClientManager.java +++ b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/RestTransportClientManager.java @@ -17,60 +17,26 @@ package io.servicecomb.transport.rest.client; -import org.slf4j.Logger; -import org.slf4j.LoggerFactory; - import io.servicecomb.foundation.vertx.VertxUtils; import io.vertx.core.Vertx; -/** - * REST客户端。只需要两个实例， 一个ssl，一个非ssl. - */ public final class RestTransportClientManager { - private static final Logger LOGGER = LoggerFactory.getLogger(RestTransportClientManager.class); - public static final RestTransportClientManager INSTANCE = new RestTransportClientManager(); // same instance in AbstractTranport. need refactor in future. private final Vertx transportVertx = VertxUtils.getOrCreateVertxByName("transport", null); - private static final Object LOCK = new Object();

- - private volatile RestTransportClient sslClient = null; - - private volatile RestTransportClient nonSslCient = null; + private RestTransportClient restClient = new RestTransportClient(); private RestTransportClientManager() { - } - - public RestTransportClient getRestTransportClient(boolean sslEnabled) { try { - if (sslEnabled) { - if (sslClient == null) { - synchronized (LOCK) { - if (sslClient == null) { - RestTransportClient client = new RestTransportClient(true); - client.init(transportVertx); - sslClient = client; - } - } - } - return sslClient; - } else { - if (nonSslCient == null) { - synchronized (LOCK) { - if (nonSslCient == null) { - RestTransportClient client = new RestTransportClient(false); - client.init(transportVertx); - nonSslCient = client; - } - } - } - return nonSslCient; - } + restClient.init(transportVertx); } catch (Exception e) { - LOGGER.error(""); - throw new IllegalStateException("init rest client transport failed."); + throw new IllegalStateException("Failed to init RestTransportClient.", e); } } + + public RestTransportClient getRestClient() { + return restClient; + } } diff --git a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/TransportClientConfig.java b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/TransportClientConfig.java index dbcf50abe..498a3b1ed 100644 --- a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/TransportClientConfig.java +++ b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/TransportClientConfig.java @@ -28,11 +28,6 @@ public static int getThreadCount() { getIntProperty("cse.rest.client.thread-count", 1).get(); } - public static int getConnectionPoolPerThread() { - return DynamicPropertyFactory.getInstance(). - getIntProperty("cse.rest.client.connection-pool-per-thread", 1).get(); - } - public static int getConnectionMaxPoolSize() { return DynamicPropertyFactory.getInstance(). getIntProperty("cse.rest.client.connection.maxPoolSize", 5).get(); diff --git a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/DefaultHttpClientFilter.java b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/DefaultHttpClientFilter.java index 5ef9d9d95..3241ffff3 100644 --- a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/DefaultHttpClientFilter.java +++ b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/DefaultHttpClientFilter.java @@ -69,7 +69,8 @@ public Response afterReceiveResponse(Invocation invocation, HttpServletResponseE produceProcessor = findProduceProcessor(swaggerRestOperation, responseEx); if (produceProcessor == null) { String msg = - String.format("path %s, statusCode %d, reasonPhrase %s, response content-type %s is not supported", + String.format("method %s, path %s, statusCode %d, reasonPhrase %s, response content-type %s is not supported", + swaggerRestOperation.getHttpMethod(), swaggerRestOperation.getAbsolutePath(), responseEx.getStatus(), responseEx.getStatusType().getReasonPhrase(), diff --git a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/VertxHttpMethod.java b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/VertxHttpMethod.java index b432cc879..a239a9106 100644 --- a/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/VertxHttpMethod.java +++ b/transports/transport-rest/transport-rest-client/src/main/java/io/servicecomb/transport/rest/client/http/VertxHttpMethod.java @@ -48,6 +48,7 @@ import io.vertx.core.http.HttpClientRequest; import io.vertx.core.http.HttpClientResponse; import io.vertx.core.http.HttpMethod; +import io.vertx.core.http.RequestOptions; public class VertxHttpMethod { private static final Logger LOGGER = LoggerFactory.getLogger(VertxHttpMethod.class); @@ -92,14 +93,6 @@ public void doMethod(HttpClientWithContext httpClientWithContext, Invocation inv asyncResp.fail(invocation.getInvocationType(), e); }); - LOGGER.debug(- "Running HTTP method {} on {} at {}:{}{}", - invocation.getOperationMeta().getMethod(), - invocation.getMicroserviceName(), - ipPort.getHostOrIp(), - ipPort.getPort(), - path); - // 从业务线程转移到网络线程中去发送 httpClientWithContext.runOnContext(httpClient -> { this.setCseContext(invocation, clientRequest); @@ -122,9 +115,20 @@ private HttpMethod getMethod(Invocation invocation) { HttpClientRequest createRequest(HttpClient client, Invocation invocation, IpPort ipPort, String path, AsyncResponse asyncResp) { + URIEndpointObject endpoint = (URIEndpointObject) invocation.getEndpoint().getAddress(); + RequestOptions requestOptions = new RequestOptions(); + requestOptions.setHost(ipPort.getHostOrIp()) + .setPort(ipPort.getPort()) + .setSsl(endpoint.isSslEnabled()) + .setURI(path); HttpMethod method = getMethod(invocation); - LOGGER.debug("Calling method {} of {} by rest", method, invocation.getMicroserviceName()); - HttpClientRequest request = client.request(method, ipPort.getPort(), ipPort.getHostOrIp(), path, response -> { + LOGGER.debug("Sending request by rest, method={}, qualifiedName={}, path={}, endpoint={}.", + method, + invocation.getMicroserviceQualifiedName(), + path, + invocation.getEndpoint().getEndpoint()); + HttpClientRequest request = client.request(method, requestOptions, response -> { handleResponse(invocation, response, asyncResp); }); return request; diff --git a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestRestTransportClient.java b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestRestTransportClient.java index 11abd42dd..3ac2b29c2 100644 --- a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestRestTransportClient.java +++ b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestRestTransportClient.java @@ -44,8 +44,6 @@ private RestTransportClient instance = null; - private HttpClientOptions options; - Invocation invocation = Mockito.mock(Invocation.class); AsyncResponse asyncResp = Mockito.mock(AsyncResponse.class); @@ -56,7 +54,7 @@ @Before public void setUp() throws Exception { - instance = new RestTransportClient(false); + instance = new RestTransportClient(); } @After @@ -70,17 +68,8 @@ public void testGetInstance() { } @Test - public void testSSL(@Mocked Vertx vertx, @Mocked VertxUtils utils) throws Exception { + public void init(@Mocked Vertx vertx, @Mocked VertxUtils utils) throws Exception { new MockUp<VertxUtils>() { - @Mock - <CLIENT_POOL, CLIENT_OPTIONS> DeploymentOptions createClientDeployOptions(- ClientPoolManager<CLIENT_POOL> clientMgr, - int instanceCount, - int poolCountPerVerticle, CLIENT_OPTIONS clientOptions) { - options = (HttpClientOptions) clientOptions; - return null; - } - @Mock <VERTICLE extends AbstractVerticle> boolean blockDeploy(Vertx vertx, Class<VERTICLE> cls, @@ -88,12 +77,11 @@ public void testSSL(@Mocked Vertx vertx, @Mocked VertxUtils utils) throws Except return true; } }; - RestTransportClient client = new RestTransportClient(true); + RestTransportClient client = =

new RestTransportClient(); client.init(vertx); - Assert.assertEquals(options.isSsl(), true); - Assert.assertEquals(options.getIdleTimeout(), 30); - Assert.assertEquals(options.isKeepAlive(), true); - Assert.assertEquals(options.getMaxPoolSize(), 5); + + ClientPoolManager<HttpClientWithContext> clientMgr = Deencapsulation.getField(client, "clientMgr"); + Assert.assertSame(vertx, Deencapsulation.getField(clientMgr, "vertx")); } @Test @@ -111,32 +99,7 @@ public void testRestTransportClientException() { @Test public void testCreateHttpClientOptions() { - HttpClientOptions obj = (HttpClientOptions) Deencapsulation.invoke(instance, "createHttpClientOptions"); Assert.assertNotNull(obj); } - - @Test - public void testSend() { - boolean validAssert; - Mockito.when(invocation.getOperationMeta()).thenReturn(operationMeta); - - new MockUp<ClientPoolManager<HttpClientWithContext>>() { - - @Mock - public HttpClientWithContext findThreadBindClientPool() { - return new HttpClientWithContext(null, null); - } - }; - - Mockito.when(operationMeta.getExtData(RestConst.SWAGGER_REST_OPERATION)).thenReturn(swaggerRestOperation); - - try { - validAssert = true; - instance.send(invocation, asyncResp); - } catch (Exception e) { - validAssert = false; - } - Assert.assertTrue(validAssert); - } } diff --git a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestRestTransportClientManager.java b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestRestTransportClientManager.java deleted file mode 100644 index ed4610ff0..000000000 --- a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestRestTransportClientManager.java +++ /dev/null @@ -1,34 +0,0 @@ -/* - * 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. - */ -package io.servicecomb.transport.rest.client; -import org.junit.Assert; -import org.junit.Test; -public class TestRestTransportClientManager { - @Test - public void testRestTransportClientManager() { - RestTransportClient client1 = RestTransportClientManager.INSTANCE.getRestTransportClient(false); - RestTransportClient client2 = RestTransportClientManager.INSTANCE.getRestTransportClient(false); - Assert.assertEquals(client1, client2); - - RestTransportClient client3 = RestTransportClientManager.INSTANCE.getRestTransportClient(true); - RestTransportClient client4 = RestTransportClientManager.INSTANCE.getRestTransportClient(true); - Assert.assertEquals(client3, client4); - } -} diff --git a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestTransportClientConfig.java b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestTransportClientConfig.java index 07e25b669..bfa683a8b 100644 --- a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestTransportClientConfig.java +++ b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/TestTransportClientConfig.java @@ -21,14 +21,23 @@ import org.junit.Test; public class TestTransportClientConfig { - @Test - public void testGetThreadCount() { + public void getThreadCount() { Assert.assertEquals(1, TransportClientConfig.getThreadCount()); } @Test - public void testGetConnectionPoolPerThread() { - Assert.assertEquals(1, TransportClientConfig.getConnectionPoolPerThread()); + public void getConnectionMaxPoolSize() { + Assert.assertEquals(5, TransportClientConfig.getConnectionMaxPoolSize()); + } + + @Test + public void getConnectionIdleTimeoutInSeconds() { + Assert.assertEquals(30, TransportClientConfig.getConnectionIdleTimeoutInSeconds()); + } + + @Test + public void getConnectionKeepAlive() { + Assert.assertTrue(TransportClientConfig.getConnectionKeepAlive()); } } diff --git a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestDefaultHttpClientFilter.java b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestDefaultHttpClientFilter.java index e64d957de..6c054f2d1 100644 --- a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestDefaultHttpClientFilter.java +++ b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestDefaultHttpClientFilter.java @@ -109,7 +109,7 @@ public void testAfterReceiveResponseNullProduceProcessor(@Mocked Invocation invo Response response = filter.afterReceiveResponse(invocation, responseEx); InvocationException exception = response.getResult(); CommonExceptionData data = (CommonExceptionData) exception.getErrorData(); - Assert.assertEquals("path null, statusCode 0, reasonPhrase null, response content-type null is not supported", + Assert.assertEquals("method null, path null, statusCode 0, reasonPhrase null, response content-type null is not supported", data.getMessage()); } diff --git a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestVertxHttpMethod.java b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestVertxHttpMethod.java index 30ab3faaa..aedbc9879 100644 --- a/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestVertxHttpMethod.java +++ b/transports/transport-rest/transport-rest-client/src/test/java/io/servicecomb/transport/rest/client/http/TestVertxHttpMethod.java @@ -120,6 +120,11 @@ public void testCreateRequest() { HttpClient client = mock(HttpClient.class); Invocation invocation = mock(Invocation.class); OperationMeta operationMeta = mock(OperationMeta.class); + Endpoint endpoint = mock(Endpoint.class); + URIEndpointObject address = mock(URIEndpointObject.class); + when(invocation.getEndpoint()).thenReturn(endpoint); + when(endpoint.getAddress()).thenReturn(address); + when(address.isSslEnabled()).thenReturn(false); when(invocation.getOperationMeta()).thenReturn(operationMeta); RestOperationMeta swaggerRestOperation = mock(RestOperationMeta.class); when(operationMeta.getExtData(RestConst.SWAGGER_REST_OPERATION)).thenReturn(swaggerRestOperation); diff --git a/transports/transport-rest/transport-rest-servlet/src/main/java/io/servicecomb/transport/rest/servlet/ServletRestTransport.java b/transports/transport-rest/transport-rest-servlet/src/main/java/io/servicecomb/transport/rest/servlet/ServletRestTransport.java index f7c195a3b..972df7f2c

100644 --- a/transports/transport-rest/transport-rest-servlet/src/main/java/io/servicecomb/transport/rest/servlet/ServletRestTransport.java +++ b/transports/transport-rest/transport-rest-servlet/src/main/java/io/servicecomb/transport/rest/servlet/ServletRestTransport.java @@ -27,7 +27,6 @@ import io.servicecomb.core.Invocation; import io.servicecomb.core.transport.AbstractTransport; -import io.servicecomb.foundation.common.net.URIEndpointObject; import io.servicecomb.serviceregistry.api.Const; import io.servicecomb.swagger.invocation.AsyncResponse; import io.servicecomb.transport.rest.client.RestTransportClient; @@ -37,6 +36,8 @@ public class ServletRestTransport extends AbstractTransport { private static final Logger LOGGER = LoggerFactory.getLogger(ServletRestTransport.class); + private RestTransportClient restClient; + @Override public String getName() { return io.servicecomb.core.Const.RESTFUL; @@ -69,19 +70,12 @@ public boolean init() { String listenAddress = ServletConfig.getLocalServerAddress(); setListenAddressWithoutSchema(listenAddress, queryMap); - return deployClient(); - } - - private boolean deployClient() { - return RestTransportClientManager.INSTANCE.getRestTransportClient(true) != null && - RestTransportClientManager.INSTANCE.getRestTransportClient(false) != null; + restClient = RestTransportClientManager.INSTANCE.getRestClient(); + return true; } @Override public void send(Invocation invocation, AsyncResponse asyncResp) throws Exception { - URIEndpointObject endpoint = (URIEndpointObject) invocation.getEndpoint().getAddress(); - RestTransportClient client = - RestTransportClientManager.INSTANCE.getRestTransportClient(endpoint.isSslEnabled()); - client.send(invocation, asyncResp); + restClient.send(invocation, asyncResp); } } diff --git a/transports/transport-rest/transport-rest-vertx/src/main/java/io/servicecomb/transport/rest/vertx/VertxRestTransport.java b/transports/transport-rest/transport-rest-vertx/src/main/java/io/servicecomb/transport/rest/vertx/VertxRestTransport.java index 17f48a85c..ccf5b485b 100644 --- a/transports/transport-rest/transport-rest-vertx/src/main/java/io/servicecomb/transport/rest/vertx/VertxRestTransport.java +++ b/transports/transport-rest/transport-rest-vertx/src/main/java/io/servicecomb/transport/rest/vertx/VertxRestTransport.java @@ -37,6 +37,8 @@ public class VertxRestTransport extends AbstractTransport { private static final Logger log = LoggerFactory.getLogger(VertxRestTransport.class); + private RestTransportClient restClient; + @Override public String getName() { return Const.RESTFUL; @@ -66,25 +68,18 @@ public boolean canInit() { } @Override public boolean init() throws Exception { + restClient = RestTransportClientManager.INSTANCE.getRestClient(); + // 部署transport server DeploymentOptions options = new DeploymentOptions().setInstances(TransportConfig.getThreadCount()); SimpleJsonObject json = new SimpleJsonObject(); json.put(ENDPOINT_KEY, getEndpoint()); options.setConfig(json); - return VertxUtils.blockDeploy(transportVertx, RestServerVerticle.class, options) && deployClient(); - } - - private boolean deployClient() { - return RestTransportClientManager.INSTANCE.getRestTransportClient(true) != null && - RestTransportClientManager.INSTANCE.getRestTransportClient(false) != null; + return VertxUtils.blockDeploy(transportVertx, RestServerVerticle.class, options) ; } @Override public void send(Invocation invocation, AsyncResponse asyncResp) throws Exception { - URIEndpointObject endpoint = (URIEndpointObject) invocation.getEndpoint().getAddress(); - RestTransportClient client = - RestTransportClientManager.INSTANCE.getRestTransportClient(endpoint.isSslEnabled()); - log.debug("Sending request by rest to endpoint {}:{}", endpoint.getHostOrIp(), endpoint.getPort()); - client.send(invocation, asyncResp); + restClient.send(invocation, asyncResp); } } ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org