Ignite Tests How To

This page describes hits for developing and researching failures of Apache Ignite tests

- Ignite Specific Test Frameworks
  - Test compatibility with older versions
  - Test of .NET API parity with Java API
  - Test configurations
  - Misc
    - Locate run configuration which runs test case
    - Enable Test Debug
    - Test timeout
      - Fast run config timed out
      - Timed out suite with sufficient timeout
      - Normal thread execution examples

Ignite Specific Test Frameworks

Test compatibility with older versions

Ignite test has built in Framework to test compatibility. This framework provides an opportunity to start working with Ignite instances of previously released versions.

The entire module is built on top of the Ignite Testing Framework, especially on the MultiJVM-mode classes. There is a class IgniteCompatibilityAbstractTest that provides methods to start Ignite nodes with versions which have been previously released in the Maven repository in separate JVM and allows them to join topology.

The framework is looking for artifacts of a specific version in the Maven local repository, and if they don’t exist there, they will be downloaded and stored via Maven.

The main implemented API:

```java
startGrid(name, version, configurationClosure);
startGrid(name, version, configurationClosure, postStartupClosure);
```

You can specify a version of Ignite, which you want to start, define the configuration in the configurationClosure and set the actions on the started node in the postStartupClosure.

It’s straightforward to use it for writing unit tests, here is a simple example which demonstrates main functions.

Test of .NET API parity with Java API

This test checks that everything that is on the public API in the configuration, is there in the .NET, unless specified otherwise. Exceptions are:

- "it’s not necessary in .NET"
- "it’s not yet supported in .NET".

If there is a public API, but it is not in the .NET class, or in the list of unnecessary, or in the list of known unsupported, then the test fails. This fix explicitly marks the property as yet unimplemented in class

modules/platforms/dotnet/Apache.Ignite.Core.Tests/ApiParity/IgniteConfigurationParityTest.cs

there is String array MissingProperties. This array stores properties that are missing on .NET side. Adding property to this list disable Parity test fails, but it is reasonable to add properties only with corresponding issue creation first. Issue number can be added as comment

Test configurations

Since OptimizedMarshaller was removed in Ignite 2.0 from the PublicAPI, several unnecessary test suites were removed from the build plan from Ignite 2.0.

Please use for Ignite 2.0+ tests appropriate run configs from Ignite 2.0 project, which is 14 test suites shorter than the previous plan.

Use -> Run All to run all suites for changes. Select your PR in branch selection.
Misc

Locate run configuration which runs test case

Usually it is clear from test suite naming to which run config it belongs.

But it is not clear where test is executed on teamcity it is possible to do the following.

Way 1: Using code

- Step 1) for TestCase it is possible to find usages in idea. Some TestSuite including this case may be found. If it is corresponds to some Run
  Config name, suite found.
- Step 2) find usage may be repeated to find grouping TestSuite.
- Double check of search result: TEST_SUITE parameter in run configuration includes full class name found suite from step 2.

Way 2: Use search in top right corner in teamcity

Make sure to select 'ignite 2.0 Tests' group if 2+ tests are required

Enable Test Debug

To enable debug messages for test it is possible to set in

incubator-ignite/modules/core/src/test/config/log4j-test.xml

This XML contains commented out examples of enable debug for particular packages

```
<category name="org.apache.ignite.cache.query"> <!-- Uncomment to enable Ignite query execution debugging. -->
    <level value="DEBUG"/>
</category>
```

For example for debugging Exchange messages following XML may be inserted test config:

```
<category name="org.apache.ignite.internal.processors.cache.distributed.dht">
    <level value="DEBUG"/>
</category>
```

Be careful with committing log with debug enabled, it may generate huge amount of messages at continious integration.

Test timeout

Fast run config timed out

If relatively fast run configuration timed out

Check required time test was timed out (or timeout set on run configuration). If it is relatively low (e.g. 10 minutes) and other successful runs required 3-9
minutes consider timeout increase.

Check agent type - some windows agents works slower than linux.

Check thread dump, if build is still running (tests even not started), consider timeout increase.

```
"main" prio=6 tid=0x0000000001798000 nid=0x188c runnable [0x000000000168d000]
  java.lang.Thread.State: RUNNABLE
    at java.io.WinNTFileSystem.getBooleanAttributes(Native Method)
    at java.io.File.exists(File.java:813)
    at org.apache.maven.plugin.compiler.AbstractCompilerMojo.hasNewFile(AbstractCompilerMojo.java:1185)
```
Timed out suite with sufficient timeout

If timeout is already high, e.g. 2h or more, timeout probably indicates problem in code. To find out reason

1) download full build log from TC (it is faster to download compressed build log).

2) search ‘timed out’ or ‘Test has been timed out’ to find out which test was failed

```
[19:24:43]W:                 [org.apache.ignite:ignite-core] [2017-06-19 16:24:43,353][ERROR][main][root] Test has been timed out and will be interrupted (threads dump will be taken before interruption) [test=testPutAllAsyncFailover, timeout=120000]
```

This line is logged at the end of test execution.

3) Search backwards ‘Starting test’

```
```

This line is logged at the beginning of test execution.

Most likely there is some exception, assertion error between these 2 logged messages.

Also it is possible now to run test locally if hang up or not.

4) Thread dump analysis

After timed out tests there is also thread dump is logged. To find out abnormal activity in this dump it is useful to take into account following information
- pool type (included into pool name)
- node name (for test may include test name)

Normal thread execution examples

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Normal trace</th>
</tr>
</thead>
</table>
| sys                | System execution pool, responsible for processing internal system messages.  
<p>|                    | See also message flow section from Ignite Tests How To     | Waiting for task to execute                                                  |
| ttl-cleanup-worker | Entry cleanup worker. Provides functionality of expiration for cache entries   | Periodic sleep and wakeup                                                    |
|                    |                                                           | state=TIMED_WAITING                                                         |
|                    |                                                           | at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:226)    |
|                    |                                                           | at java.util.concurrent.LinkedBlockingQueue.poll                            |
|                    |                                                           | (LinkedBlockingQueue.java:467)                                              |
|                    |                                                           | at java.util.concurrent.ThreadPoolExecutor.getTask                        |
|                    |                                                           | (ThreadPoolExecutor.java:1068)                                              |
| exchang-worker     | One thread per node. Partition maps exchange. Usage of one thread for exchange provides strict actions order.   |
|                    | See also &quot;Partition Map Exchange&quot; section from Ignite Tests How To | If there is no exchange waits on the queue                                  |
| grid-nio-worker-tcp-comm |                                                            |                                                                              |</p>
<table>
<thead>
<tr>
<th>nio-acceptor</th>
<th>See also 'Striped pool' section from Part 2</th>
<th>Waiting on queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys-stripe</td>
<td>state=WAITING</td>
<td>state=WAITING</td>
</tr>
<tr>
<td></td>
<td>at o.a.i.i.util.StripedExecutor$StripeConcurrentQueue.take(StripedExecutor.java:581)</td>
<td>at o.a.i.i.util.StripedExecutor$StripeConcurrentQueue.take(StripedExecutor.java:581)</td>
</tr>
<tr>
<td>tcp-disco-sock-reader</td>
<td>Reads socket</td>
<td>Reads socket</td>
</tr>
<tr>
<td>tcp-disco-ip-finder</td>
<td>Waiting on queue</td>
<td>Waiting on queue</td>
</tr>
<tr>
<td>tcp-disco-msg-worker</td>
<td>Waiting on queue</td>
<td>Waiting on queue</td>
</tr>
<tr>
<td>update-thread</td>
<td></td>
<td></td>
</tr>
<tr>
<td>restart-thread</td>
<td></td>
<td></td>
</tr>
<tr>
<td>test-runner</td>
<td>Runs test itself</td>
<td>Test method e.g. CacheAsyncOperationsFailoverAbstractTest.testPutAllAsyncFailover()</td>
</tr>
<tr>
<td>disco-event-worker</td>
<td>Waiting on queue</td>
<td>Waiting on queue</td>
</tr>
<tr>
<td>main</td>
<td>Start up test runner thread and waits to complete within getTestTimeout()</td>
<td>ThreadImpl.dumpThreads0 - this thread checks timeout occurred and initializes thread dump</td>
</tr>
</tbody>
</table>