**Parallel Testing in TestNG (Brief Overview)**

**What is Parallel Testing?**

Parallel testing in TestNG allows multiple test cases to execute **simultaneously** instead of sequentially. This helps in:  
✅ **Reducing execution time**  
✅ **Utilizing system resources efficiently**  
✅ **Speeding up test automation**

**How to Enable Parallel Testing in TestNG?**

You can configure parallel execution using the **TestNG XML file**.

**1. Parallel Execution at Class Level**

To run multiple test classes in parallel, use parallel="classes" in the <suite> tag.

📌 **Example: testng.xml**

xml

<suite name="ParallelTestSuite" parallel="classes" thread-count="2">

<test name="Test1">

<classes>

<class name="tests.TestClass1"/>

<class name="tests.TestClass2"/>

</classes>

</test>

</suite>

🔹 **Effect:** Both TestClass1 and TestClass2 will run simultaneously in two different threads.

**2. Parallel Execution at Method Level**

To run multiple test methods in parallel, use parallel="methods".

📌 **Example:**

xml

<suite name="ParallelMethodSuite" parallel="methods" thread-count="3">

<test name="Test2">

<classes>

<class name="tests.ParallelTest"/>

</classes>

</test>

</suite>

🔹 **Effect:** All test methods inside ParallelTest class will execute in parallel.

📌 **Example Test Class:**

java

import org.testng.annotations.Test;

public class ParallelTest {

@Test

public void testMethod1() {

System.out.println("Test Method 1 - " + Thread.currentThread().getId());

}

@Test

public void testMethod2() {

System.out.println("Test Method 2 - " + Thread.currentThread().getId());

}

}

🔹 **Output Example:**

mathematica

Test Method 1 - Thread ID: 12

Test Method 2 - Thread ID: 14

**3. Parallel Execution with DataProviders**

TestNG also allows running **data-driven tests in parallel**.

📌 **Example:**

java

import org.testng.annotations.\*;

public class ParallelDataProviderTest {

@Test(dataProvider = "data", parallel = true)

public void test(String value) {

System.out.println("Test executed with: " + value + " | Thread ID: " + Thread.currentThread().getId());

}

@DataProvider(name = "data", parallel = true)

public Object[][] getData() {

return new Object[][] { {"A"}, {"B"}, {"C"} };

}

}

🔹 **Effect:** Each data set (A, B, C) will run in parallel.

**Key Takeaways**

✔ **Parallel testing speeds up execution by running tests simultaneously.**  
✔ **TestNG allows parallel execution at the suite, class, method, and data-provider levels.**  
✔ **Use thread-count to control concurrency and avoid resource exhaustion.**

TestNG Parallel Execution Configuration Options

In the context of test configuration, particularly in tools like TestNG (a popular testing framework for Java), the parallel attribute is used to specify the mode of parallel execution for your tests. The parallel attribute can take several values, each defining a different way to parallelize the test execution. Here are the common values you can assign to the parallel attribute and what they mean:

**1. methods**

* **Meaning**: Test methods within the same class will be run in parallel.
* **Use Case**: Useful when you have multiple independent test methods in a single test class and you want to speed up execution by running them concurrently.

xml

<test name="Test" parallel="methods" thread-count="4">

<classes>

<class name="com.example.TestClass"/>

</classes>

</test>

Run HTML

**2. tests**

* **Meaning**: Different <test> tags in your TestNG XML file will be run in parallel.
* **Use Case**: Useful when you have multiple independent test suites or test groups that can be executed concurrently.

xml

<suite name="Suite" parallel="tests" thread-count="4">

<test name="Test1">

<classes>

<class name="com.example.TestClass1"/>

</classes>

</test>

<test name="Test2">

<classes>

<class name="com.example.TestClass2"/>

</classes>

</test>

</suite>

Run HTML

**3. classes**

* **Meaning**: Test classes will be run in parallel.
* **Use Case**: Useful when you have multiple independent test classes and you want to run them concurrently.

xml

<test name="Test" parallel="classes" thread-count="4">

<classes>

<class name="com.example.TestClass1"/>

<class name="com.example.TestClass2"/>

</classes>

</test>

Run HTML

**4. instances**

* **Meaning**: Test instances will be run in parallel. This is useful when you have multiple instances of the same test class.
* **Use Case**: Useful when you have a test class that is instantiated multiple times (e.g., with different data providers) and you want to run these instances in parallel.

xml

<test name="Test" parallel="instances" thread-count="4">

<classes>

<class name="com.example.TestClass"/>

</classes>

</test>

Run HTML

**5. none**

* **Meaning**: No parallel execution. Tests will run sequentially.
* **Use Case**: Useful when you want to ensure that tests run one after the other, without any concurrency.

xml

<test name="Test" parallel="none">

<classes>

<class name="com.example.TestClass"/>

</classes>

</test>

Run HTML

**6. true**

* **Meaning**: This is a deprecated value and is equivalent to methods. It means that test methods will be run in parallel.
* **Use Case**: Similar to methods, but it's recommended to use methods instead for clarity.

xml

<test name="Test" parallel="true" thread-count="4">

<classes>

<class name="com.example.TestClass"/>

</classes>

</test>

Run HTML

**7. false**

* **Meaning**: This is a deprecated value and is equivalent to none. It means that tests will run sequentially.
* **Use Case**: Similar to none, but it's recommended to use none instead for clarity.

xml

<test name="Test" parallel="false">

<classes>

<class name="com.example.TestClass"/>

</classes>

</test>

Run HTML

**Summary**

* **methods**: Parallelize at the method level.
* **tests**: Parallelize at the test suite level.
* **classes**: Parallelize at the class level.
* **instances**: Parallelize at the instance level.
* **none**: No parallel execution (sequential).
* **true**: Deprecated, equivalent to methods.
* **false**: Deprecated, equivalent to none.

**Example with thread-count**

The thread-count attribute specifies the maximum number of threads to use for parallel execution. For example, if you set thread-count="4", TestNG will use up to 4 threads to run your tests in parallel.

xml

<suite name="Suite" parallel="methods" thread-count="4">

<test name="Test">

<classes>

<class name="com.example.TestClass"/>

</classes>

</test>

</suite>

Run HTML

In this example, up to 4 test methods from TestClass will run in parallel.

**Choosing the Right Parallel Mode**

* Use **methods** if your test methods are independent and can run concurrently.
* Use **tests** if you have multiple independent test suites.
* Use **classes** if you have multiple independent test classes.
* Use **instances** if you have multiple instances of the same test class.
* Use **none** if you need tests to run sequentially.

**Important Considerations**

* **Thread Safety**: Ensure that your test methods are thread-safe if running in parallel.
* **Resource Contention**: Be aware of shared resources (e.g., databases, files) that might cause contention issues when tests run in parallel.
* **Dependencies**: Avoid parallel execution if tests have dependencies on each other.

By carefully selecting the appropriate parallel mode and thread-count, you can significantly reduce the execution time of your test suite while ensuring reliable and consistent test results.