Troubleshooting Parallel Tests

These are good, general practices for investigating problems when parallelising your test suite. Further investigations and assistance will need to come from your own company and language community, because parallelization is a complex topic and we can't hope to cover everything. This topic is specifically aimed at covering issues with Selenium when running tests in parallel. The Selenium protocol is pretty simple, as is Sauce Labs' integration with it: if you send us a correct Selenium command, we'll obey it. Our service doesn't take any action that isn't driven by your test suite, so if we are sure Selenium is working correctly, then it's highly likely something other than Sauce Labs is causing issues.

Add Logging

Why log instead of using trace tools?

Many languages have tools for tracing through code as it runs, monitoring what lines and classes are currently executing. These can be extremely useful for debugging code problems, provided you're already familiar with them. Because many people aren't familiar with these tools, we recommend you start by adding logging to your tests. Lots of logging. **All the Logging.** Logging also makes it easier to reason about multiple threads over a period of time.

How should I do my logging?

You can use your language or test framework's logging tools, your container's logging tools, or even printing to standard out. Printing to standard out is a brute force approach, but can often be the best one to use when trying to debug problems on a continuous integration system, where logging framework output may be hard to access.

Regardless of how you do your logging, you should always include:

1. Timestamps.
2. ID of the current thread or process (where applicable).
3. ID of the Selenium driver currently in use (where applicable).

A good template to log with is:

```
[TIMESTAMP] - [PROCESS ID]: [DRIVER ID] - [Log Message]
```

Where `Driver ID` is only present if relevant.

If you're logging using a tool that allows for log levels, log to the `DEBUG` or `VERBOSE` levels of logging.

What should I actually log?

The purpose of logging is to inform you of concurrency problems that are occurring. So, you should log everything relevant to the problem you're seeing. At a minimum, you should add logging to show:

- Just before creating a Selenium driver, with the desired capabilities
- When the driver is created successfully, with its session ID
- When a driver is going to be quit, with its session ID
- When a driver has been successfully quit, with its session ID

Other logging required depends on your specific problem

- When a test starts
- When a test finishes
- When a test is first about to use a Selenium driver
- Every time a test is using a Selenium driver

- When a test is about to do something that is network intensive for the browser
- When test exceptions occur
- When driver setup exceptions occur
- When any pre-test actions that set up data or browser state, are about to run
- When any pre-test actions that set up data or browser state have run
- When any post-test actions that clean up data or browser state have run
- When any post-test actions that clean up data or browser state have run

You should always include **test names** when talking about specific tests, **Session IDs** when using Selenium drivers, and **Thread** or **Process ID**s.

You should also try, whenever possible, to improve the error messages your test suite throws when problems occur.

Check How Much Parallelisation Your Tests Can Stand

You should also check to see if there's a level of parallelization at which your tests work, and one where. Some Sauce Labs users have problems **only** once they've exceeded a certain number of parallel tests at once. For instance, problems with queuing and network congestion are often exposed when running higher numbers of parallel tests.

First up, check how many parallel tests your account is able to run simultaneously. You can check this by logging in to Sauce Labs. In the left-hand column you'll see the number of Concurrent VMs allowed for your account, which corresponds to the number of tests you can run in parallel. Make sure you're not exceeding this number, and if you are, simply throttle back on the number of parallel tests you're running until the problem resolves.

If your problem continues try slowly lowering the number of tests and see if there's a level, higher than one, at which you're no longer experiencing problems. If you find the level at which this occurs, then you can start investigating your logging to see what your tests do at higher levels, that differs from lower ones. For instance, you might discover that some of your tests rely on running in browsers with other tests, and when your parallelisation goes higher, this no longer happens.

If there's no level of parallelisation where your tests work correctly in parallel, or there is but it's not related to your Sauce Labs concurrency limit **a nd** you're certain it's not a networking problem, leave your parallelisation at **2** and carry on diagnosis.

Run Fewer Tests

Take tests out of your test suite! Similar to the steps above, removing tests from your test suite can surface problematic tests. It also shows if you have a problem with the length of individual thread lifecycles.

If there's a point where you remove some tests and things start functioning, try adding some of the tests you removed earlier back in, and see if they're not working correctly also. If they are, you've identified a problematic test you can check out further.

If you don't experience more stable tests by removing a couple of unstable ones, or if every test above a certain number causes issues, configure your tests to run the lowest number of tests that exhibits a problem, and keep debugging.

Read Your Logs

Time to start cross-checking the logs.

Incorrect driver setup

- "Empty" or incomplete desired capabilities
- "Empty" or missing user authentication

Non-thread-safe code

- Selenium session IDs changing during a single test
- Selenium session IDs being used across threads
- Tests using Selenium sessions that have already been quit
- Tests using Selenium sessions which don't exist

Selenium Lifecycle Problems

- Selenium sessions being created all at once, but only used by tests much later
- Selenium sessions all quit at once
- Selenium session IDs being used across multiple threads
- Selenium sessions being created and never quit
- Tests using Selenium sessions that have already been quit
- Tests using Selenium sessions which don't exist

Problem Still Unsolved

Unfortunately, if all of your tests get unique Selenium sessions created, those sessions are started and stopped correctly, are only ever used by a single thread, and don't get queued, then its likely your problems arise from something else in your test suite. Unfortunately, these are out of scope of Sauce Labs' support. It can be helpful to investigate your database and application server's ability to deal with concurrent tests, shared sessions, and test data. Good Luck!