SOLUTIONS BRIEF

FOUR WAYS TEST ANALYTICS CAN IMPROVE AUTOMATED TESTING

As a testing organization, it is important to continuously review testing trends and improve the process. To date, analyzing trends in tests has been limited to homegrown or third party apps. However, with the recently released Sauce Labs Test Analytics, we have reviewed years of anonymous data to uncover common test trends that can help modern development teams test more successfully. We have identified four key areas that most testing organizations can benefit from to improve their testing practices. If you are an executive looking to improve your team's test velocity, or an engineer interested in writing massively parallel tests, our insights can help you accelerate your tests so you can release better software faster.

SAUCELABS

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This paper focuses on four areas that many teams do not optimize for in their testing. All the test practices recommendations are based on the data Sauce Labs has gathered over the years and our experience working with thousands of customers:

1. Test job queuing: Minimize queuing to help teams test faster

2. Small, atomic and autonomous tests: The fastest testers use shorter tests.

3. Fix failures quickly: Enable teams to get maximum benefit from Agile development practices

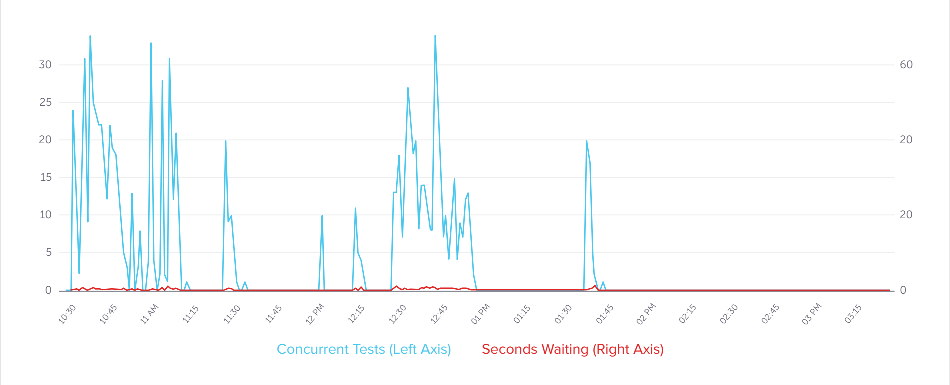
4. Annotations: Use test annotations to help teams troubleshoot faster.

LITTLE TO NO WAIT TIME HELPS TEAMS TEST FASTER

Many organizations are constrained by the number of concurrent sessions* with which they test. Due to budget, limited machine availability or large teams sharing testing resources, we have found that high wait times are one of the biggest impediments to testing faster. In general, it is a good testing practice for Sauce Labs testers to have their virtual machine concurrency spike and reach its peak multiple times a day. That shows a healthy DevOps pipeline, where

- Developers are able to run tests whenever they are ready
- Developers are not blocked by lower concurrency and do NOT have to wait for resources
- Teams within an organization are not blocked by each other

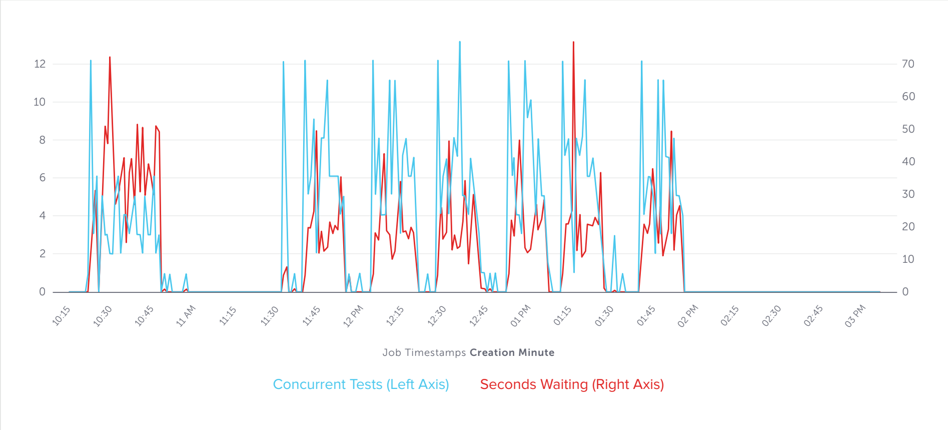
The chart below shows how constraints in capacity slow down the testing process. The blue line indicates the number of tests starting. The red line indicates the average time these tests spend waiting for capacity to free up. When an organization is not constrained and has adequate infrastructure capacity, average wait time is very low to almost insignificant. Since tests do not have to wait for resources to be freed up, developers are able to test faster and testing is not a bottleneck to their release cycles.



Example of test wait times with sufficient test concurrency assets - tests run and complete quickly

*We refer to each parallel test as a concurrent session which includes a single use Virtual Machine (VM) along and analysis tools (videos, screenshots, logs, metadata and Test Analytics).

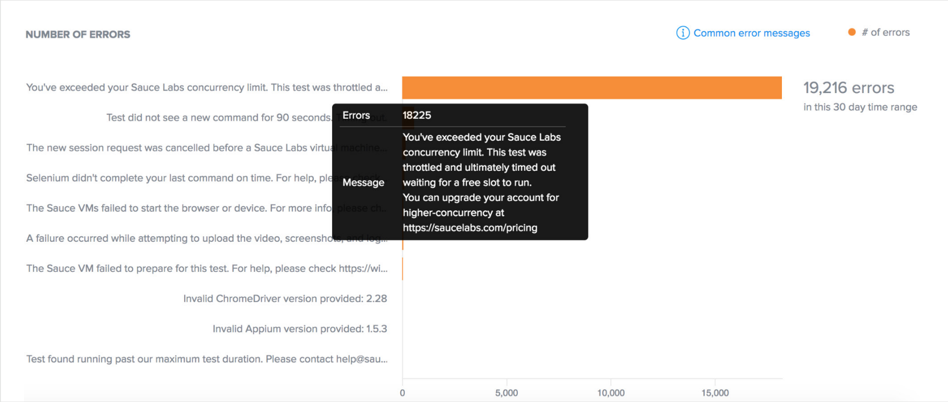
On the other hand, below is an example of how wait times spike when an organization is running without enough capacity to support their testing needs. Whenever they need to start tests, their capacity constraints incur a wait time while capacity slowly frees up. All of these wait times add up over the course of a build and slow everything down. This leads to teams battling for priority in the testing cycle and introducing needless overhead in the release cycle.



Example of test wait times with insufficient test concurrency assets - tests have high wait times and overall, testing is slowed

Obtaining this information using Sauce Analytics

Using Sauce Test Analytics, you can review concurrency utilization and error information to gauge how often your teams hit their concurrency limit and their tests are bumped off the queue. That's a good indicator of your team's capacity utilization.



NUMBER OF ERRORS

ⓘ Common error messages ● # of errors

You've exceeded your Sauce Labs concurrency limit. This test was throttled a...

Test did not see a new command for 90 seconds. T... out.

The new session request was cancelled before a Sauce Labs virtual machine...

Selenium didn't complete your last command on time. For help, please check...

The Sauce VMs failed to start the browser or device. For more info please ch...

A failure occurred while attempting to upload the video, screenshots, and log...

The Sauce VM failed to prepare for this test. For help, please check https://wi...

Invalid ChromeDriver version provided: 2.28

Invalid Appium version provided: 1.5.3

Test found running past our maximum test duration. Please contact help@sau...

19,216 errors
in this 30 day time range

Errors	18225
Message	You've exceeded your Sauce Labs concurrency limit. This test was throttled and ultimately timed out waiting for a free slot to run. You can upgrade your account for higher-concurrency at https://saucelabs.com/pricing

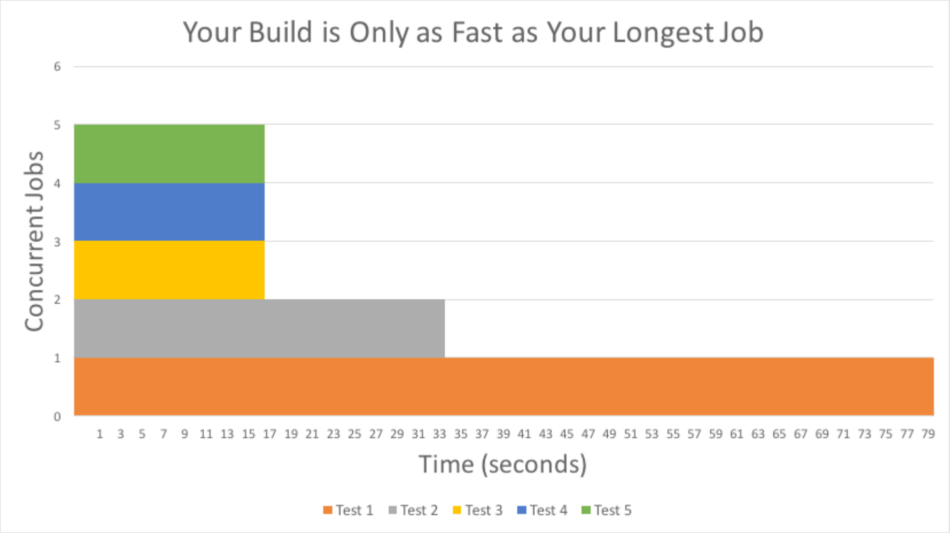
Screenshot of Test Analytics showing errors due to insufficient concurrency. Ultimately, this slows your testing and increases the time to release new features.

BEST WRITTEN TESTS ARE SHORTER

Sauce Labs has always advocated for atomic tests that validate a single specific function in an application. Atomic tests are short and test a single function making it easier and faster to identify a failure. The shorter the tests, the more efficiently they can be parallelized for execution and make use of the maximum available VM concurrency.

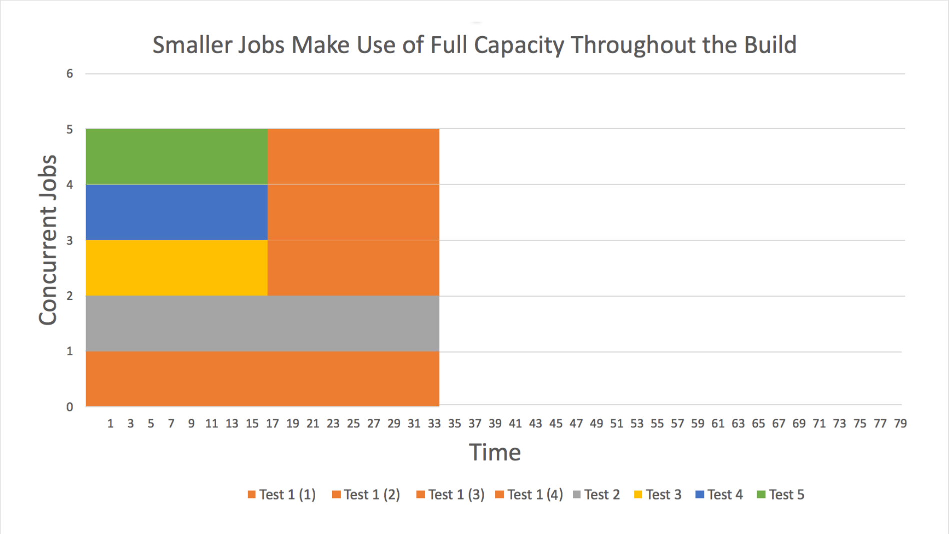
A build can only be as short as its longest test. By breaking down long-running tests into multiple smaller tests, more tests can run in parallel and builds can finish faster. Long tests bottleneck the whole build and consume the concurrency available to the whole organization, resulting in less efficient use of available resources and slower testing times. In addition, longer running tests are brittle and tend to fail more often. The failures leave the user with a long debugging process, which slows down testing even more.

The chart below shows concurrency utilization through time during a build that has a “long tail” - a single long running test which makes the entire build take much longer than it needs to, slowing down both the build time and the testing cycle.



Comparison of test times. If your tests are not similar in length, the longest test will determine total test time.

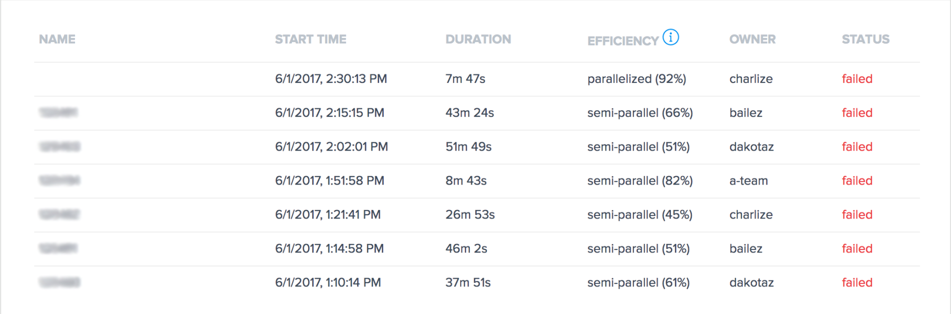
This practice of having long-running tests within a build is not recommended if it can be avoided. Below is a good example of the same build but with the single long test being broken down into four separate, much shorter tests. In this model, throughout the run time of the build, all tests run in parallel and make maximum use of the available concurrency. Because of the increased parallelism, the build runs in half the time-- thereby speeding up the entire testing process. The smaller tests are able to utilize the capacity that was previously wasted while waiting for other tests to end.



Tests that are similar in length help optimize test times so your test suite finishes quickly.

Obtaining this information using Sauce Analytics

Within the Sauce Test Analytics ‘Build and Test Statistics’, review the efficiency percentage of your builds to understand if your builds are sequential, semi-parallel or parallelized. If they are semi-parallel, review long-running tests and try to break them down into smaller atomic tests using these best practices.



NAME	START TIME	DURATION	EFFICIENCY ⓘ	OWNER	STATUS
	6/1/2017, 2:30:13 PM	7m 47s	parallelized (92%)	charlize	failed
[redacted]	6/1/2017, 2:15:15 PM	43m 24s	semi-parallel (66%)	bailez	failed
[redacted]	6/1/2017, 2:02:01 PM	51m 49s	semi-parallel (51%)	dakotaz	failed
[redacted]	6/1/2017, 1:51:58 PM	8m 43s	semi-parallel (82%)	a-team	failed
[redacted]	6/1/2017, 1:21:41 PM	26m 53s	semi-parallel (45%)	charlize	failed
[redacted]	6/1/2017, 1:14:58 PM	46m 2s	semi-parallel (51%)	bailez	failed
[redacted]	6/1/2017, 1:10:14 PM	37m 51s	semi-parallel (61%)	dakotaz	failed

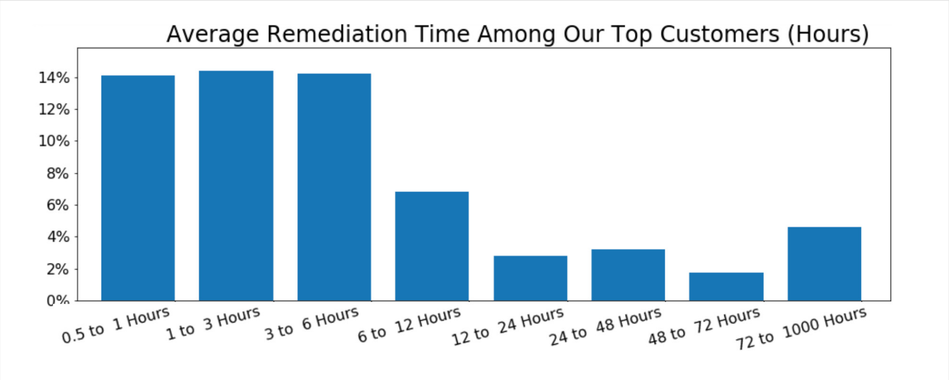
Not only can you view test duration and status, but the Efficiency column shows how parallelized your tests are to help you identify which tests are not running as quickly as possible.

HOW QUICKLY SHOULD A TEST FAILURE BE FIXED?

This subject is often debated - how quickly should a test be fixed in order for a team to continue to be agile and test quickly? The correct answer is ASAP, but in reality, automated test failures are often misconstrued as an impediment to release and teams end up finding work arounds. Often times, testers manually verify test cases to push the release forward. But the best way to continue testing and releasing faster is to determine if a failure is a release blocker or not, then fix the issue and run automated tests again to determine release readiness.

Time to fix a failure varies from team to team. Based on anonymized data, we have determined that tests are most effective when failures are fixed within the same day of their occurrence. Below is a graph that shows how quickly a test goes from red -> green.

The x-axis is represented in hours and the first bucket starts from 30 minutes to eliminate any re-tries made for flakes. As you can see, this model shows most of our users fixing failures within the first 12 hours of their occurrence. A failure starts getting less relevant if it takes more time to address it.



Most Sauce users fix test issues in under 12 hours.

OTHER TESTING BEST PRACTICES

Adding relevant test metadata

It is always good practice among testers to add metadata to tests so that they can troubleshoot faster. Here are some relevant metadata that helps in troubleshooting:

- Tagging tests as pass/fail using sauce labs REST API to make better use of analytics data.
- Adding contextual information to a test eases the troubleshooting and debugging process through Selenium logs and commands. This also helps users identify areas where tests are slowing down.

  Here is a snippet from Sauce Labs' test framework that shows how to add context information to tests written in Java using TestNG:

```
WebDriver driver = this.getWebDriver();
String commentInputText = UUID.randomUUID().toString();
this.annotate(“Visiting GuineaPig page...”);
GuineaPigPage page = GuineaPigPage.visitPage(driver);
this.annotate(String.format(“Submitting comment: \”%s\””,
commentInputText));
page.submitComment(commentInputText);
this.annotate(String.format(“Asserting submitted comment is: \”%s\””,
commentInputText));
```

- Adding build information in order to group tests together. This allows users to review test run times and help improve testing times.
- Using tags to identify different features and projects useful for analysis.

CONCLUSION

Using Sauce Test Analytics offers a number of best practices that can help any test team become more efficient and optimize the use of their testing resources. Small testing improvements go a long way - be it run time or reducing failures - and can make a big difference in shortening your release cycles. Using Sauce Test Analytics and your own data, you should be able to quickly identify bottlenecks in your testing processes that you can address to speed up your testing time. By running shorter tests, optimizing resources to minimize queuing time and addressing failures quickly, your team can reduce testing time and accelerate your releases.

SAUCELABS

Learn more at saucelabs.com

ABOUT SAUCE LABS

Sauce Labs ensures the world’s leading apps and websites work flawlessly on every browser, OS and device. Its award-winning Continuous Testing Cloud provides development and quality teams with instant access to the test coverage, scalability, and analytics they need to deliver a flawless digital experience. Sauce Labs is a privately held company funded by Toba Capital, Salesforce Ventures, Centerview Capital Technology, IVP and Adams Street Partners. For more information, please visit saucelabs.com.

SAUCE LABS

SAUCE LABS INC. - HQ
116 NEW MONTGOMERY STREET, 3RD FL
SAN FRANCISCO, CA 94105 USA

SAUCE LABS EUROPE GMBH
NEUENDORFSTR. 18B
16761 HENNIGSDORF GERMANY

SAUCE LABS INC. - CANADA
134 ABBOTT ST #501
VANCOUVER, BC V6B 2K4 CANADA