Tools

#WWDC14

Testing in Xcode 6

爷爷的儿子不能少!

你依旧没有看错!

Session 414

Brooke Callahan
Xcode Software Engineer

Wil Turner
Xcode Software Engineer

What We’ll Cover 我们会讲一些什么?

Benefits of testing 测试的好处

Getting started 如何开始测试

$\boxed{\text{Asynchronous testing}}$ 异步测试

$\boxed{\text{Performance testing}}$ 性能测试

Motivation 动机

Why test? 为什么要测试?

Find bugs 找Bugs

Codify requirements 让你的需求变得更有条理

Workflow 开始的流程

Getting started

Add tests 添加测试

Verify that tests pass 验证测试通过

Or

前方高能：这里是指TDD

Write tests 写一个测试

Write code that passes the tests 写代码，来通过测试！！

AKA “Test-Driven Development” 测试驱动开发

Workflow

Continuous Integration

1.并不完全正确的代码才被集成入库
2.入库合并前的问题代码风险可控

Test Hosting

How tests are run 测试是如何运行的?

Test bundles are executed by a host process 1.测试bundles 被主进程执行

- Injected into your app, or 测试被注入到App
- Hosting process provided by Xcode Xcode提供主进程

Resources for tests are not in the main bundle

- Don’t use +[NSBundle mainBundle]
- Use +[NSBundle bundleForClass:[MyTest class]]

What’s New

APIs and tools

Testing with Xcode 6

Compatibility improvements 兼容性的提升

Instruments integration 仪表的集成

New APIs

Asynchronous Testing 异步测试

More and more APIs are asynchronous 越来越多的异步API

- Block invocations block的调用
- Delegate callbacks 代理的调用
- Network requests 网络请求
- Background processing 后台进程

Unit tests run synchronously 单元测试运行同步

Asynchronous Testing

New APIs in XCTest

NEW

"Expectation" objects describe expected events

```
- (XCTestExpectation *)expectationWithDescription:(NSString *)description;
```

XCTestCase waits for expectations to "fulfill"

```
- (void)waitForExpectationsWithTimeout:(NSTimeInterval)timeout
                               handler:(XCWaitCompletionHandler)handlerOrNil;
```

Asynchronous Testing

Example

```
- (void)testDocumentOpening
{
    XCTestExpectation *expectation = [self expectationWithDescription:@“open doc”];

    UIDocument *doc = …;

    [doc openWithCompletionHandler:^(BOOL success) {
        XCTAssert(success);
        [expectation fulfill];
    }];

    [self waitForExpectationsWithTimeout:5.0 handler:nil];
}
```

5秒内如果执行这个方法，就不报错，如果5秒内expectation没有fullfill就抛出异常

明白代码的执行过程！

Performance Testing

Code changes can introduce performance regressions 代码改变

Catching these regressions is difficult 获取这些回归是困难的 什么是回归?

Performance testing automates this 性能测试自动化

Overview

NEW

New APIs to measure performance 性能测试的新API

New UI to interpret results 新UI解释结果

Profiling tests with Instruments 仪表盘的测试视图

Measuring Performance

New API in XCTestCase

```
- (void)measureBlock:(void (^)(void))block;
```

Takes a block of code and runs it 10 times block内的代码调用10次

Measures time 测量时间

Results show up in Xcode 在Xcode里面的结果

Measuring Performance

Example

```
- (void)testUseFileHandlePerformance
{
    [self measureBlock:^{

        NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
        XCTAssertNotNil(fileHandle);

        UseFileHandle(fileHandle);

        [fileHandle closeFile];
    }];
}
```

该Block被调用10次

Measuring Performance

Wrap-Up

通过调用block来发现性能回归

Call `–measureBlock:` to detect performance regressions

View results in Source Editor and Test Report 视图结果可以在代码编辑器和测试报告中查看

Profile tests with Instruments 仪表盘的测试视图

Performance Testing

Setting Baselines	设置基准线
Standard Deviation	标准偏差
Measuring precisely	测量精度

Setting Baselines

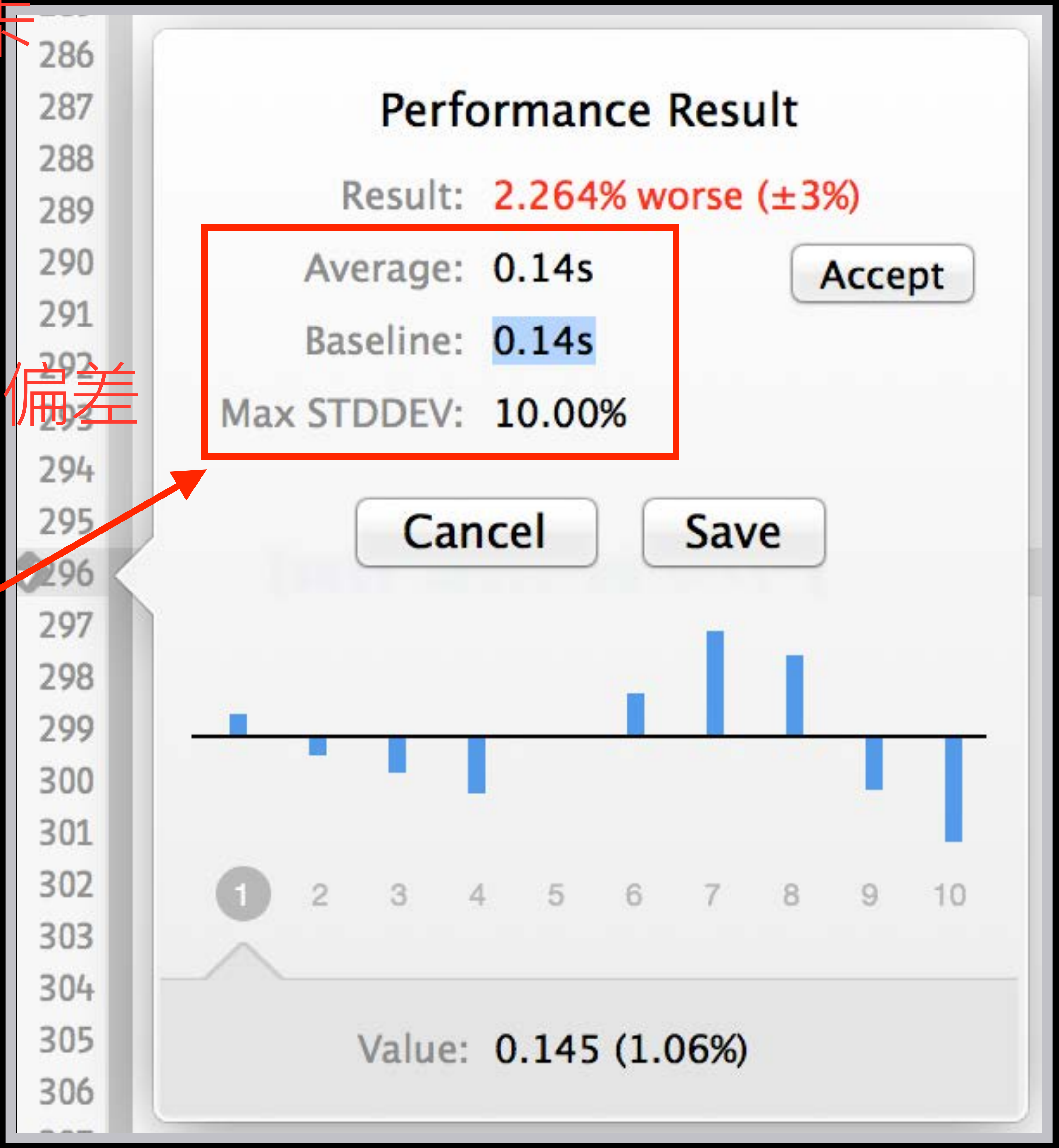
设置基线

Performance Result Popover 性能结果弹出卡

- Source Editor
- Test Report

Edit Baseline and STDDEV 编辑基线和标准偏差

重点理解



Performance Result

Result: 2.264% worse (±3%)

Average: 0.14s

Baseline: 0.14s

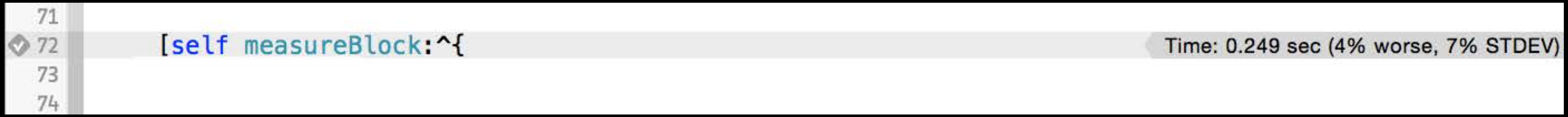
Max STDDEV: 10.00%

Value: 0.145 (1.06%)

Using Baseline Average

Source Editor annotations

- Has Baseline: Passed

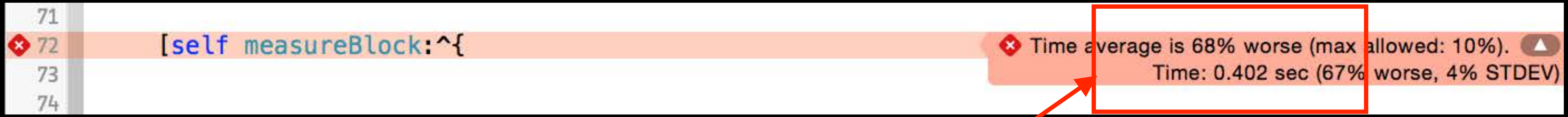


```
    [self measureBlock:^{                                    Time: 0.249 sec (4% worse, 7% STDEV)
```

Using Baseline Average

Source Editor annotations 源码编辑器注解

- Has Baseline: Failed



```
[self measureBlock:^{
```

Time average is 68% worse (max allowed: 10%).
Time: 0.402 sec (67% worse, 4% STDEV)

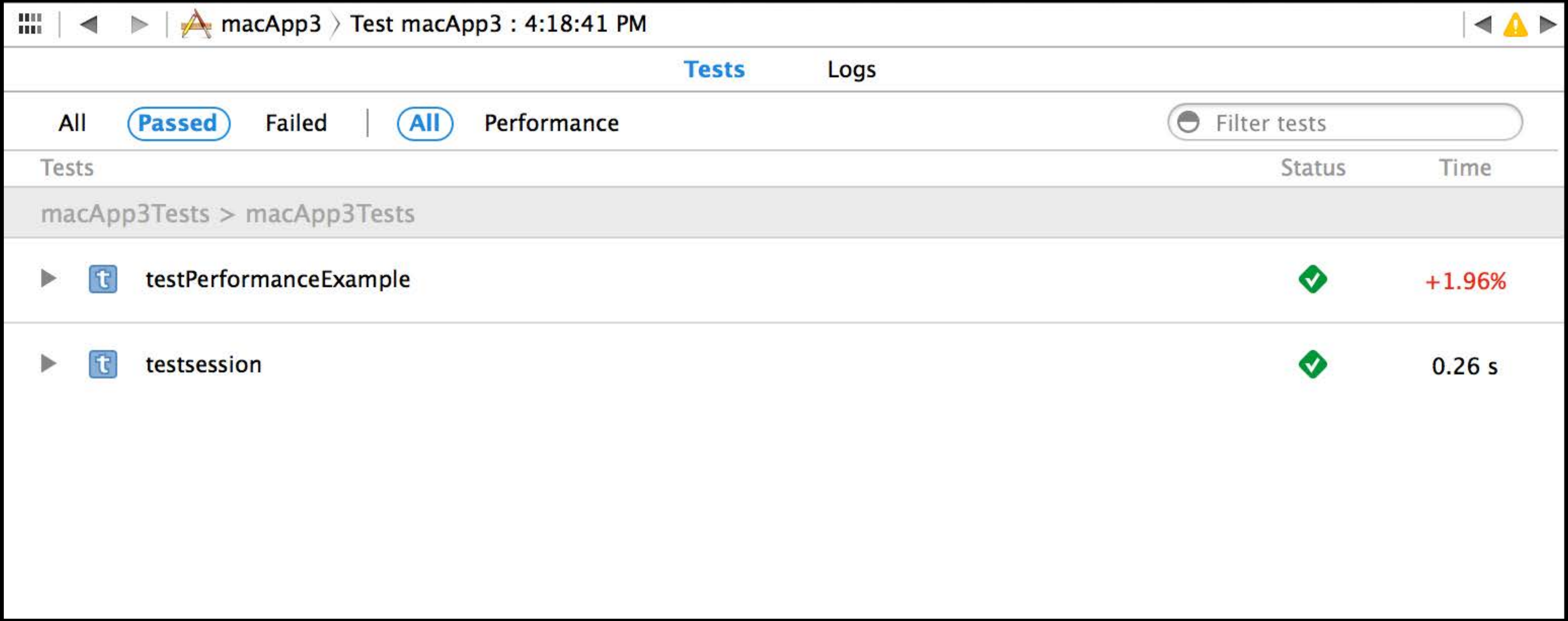
1.判断平均时间是否超过0.1

2.STTDEV是否超过10%

3.进行且运算判断，两者均失败

Using Baseline Average

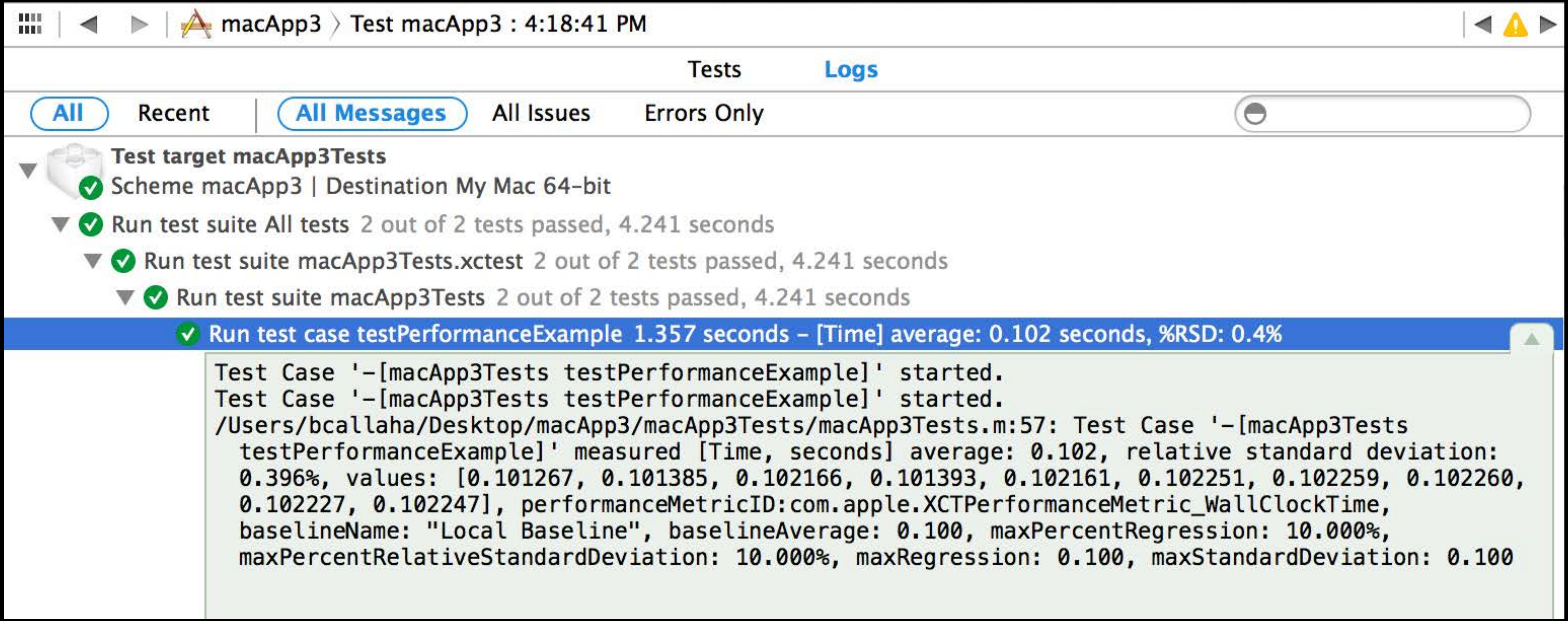
Test Report



Tests	Status	Time
macApp3Tests > macApp3Tests		
testPerformanceExample	✓	+1.96%
testsession	✓	0.26 s

Using Baseline Average

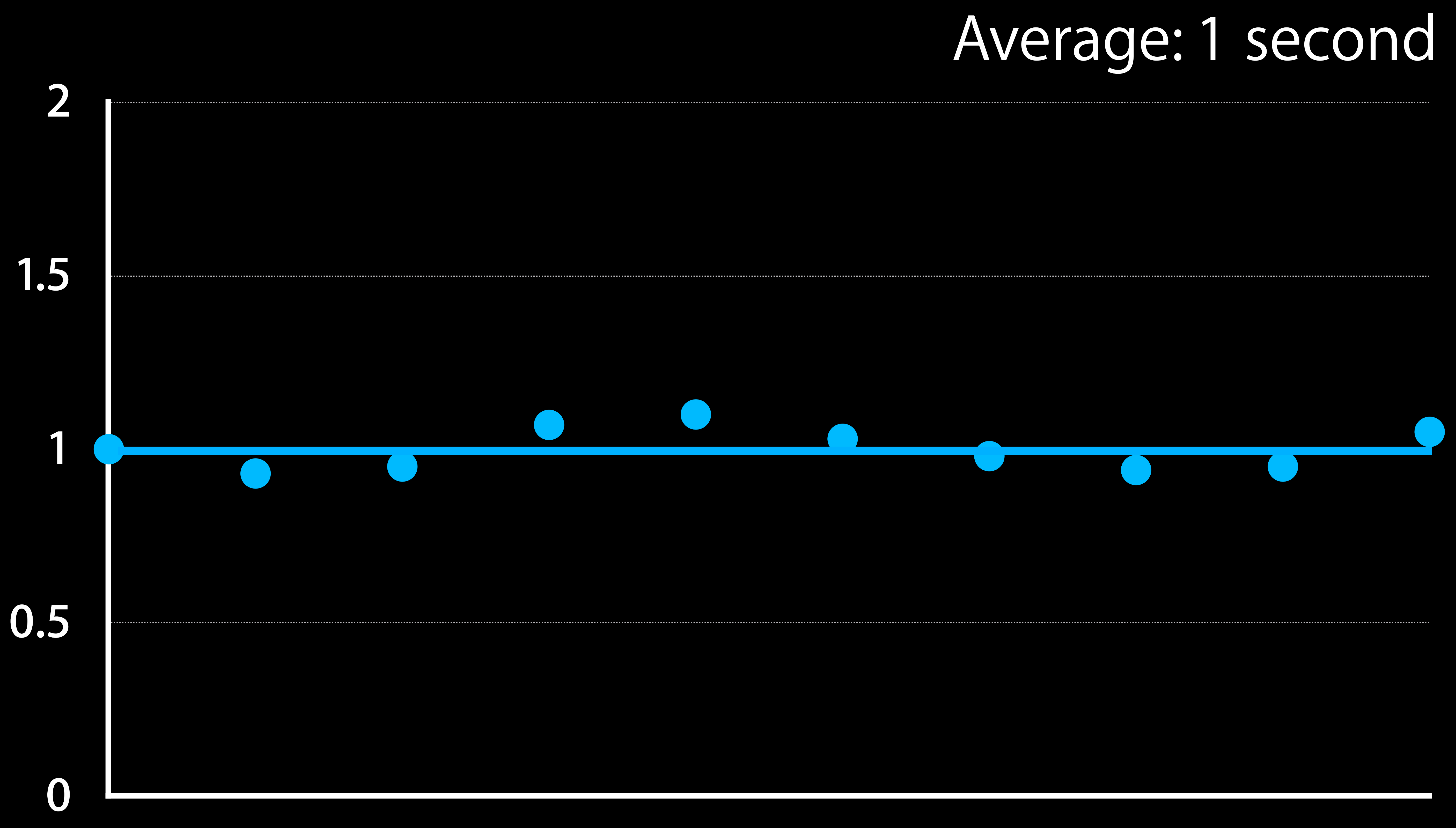
Test Log



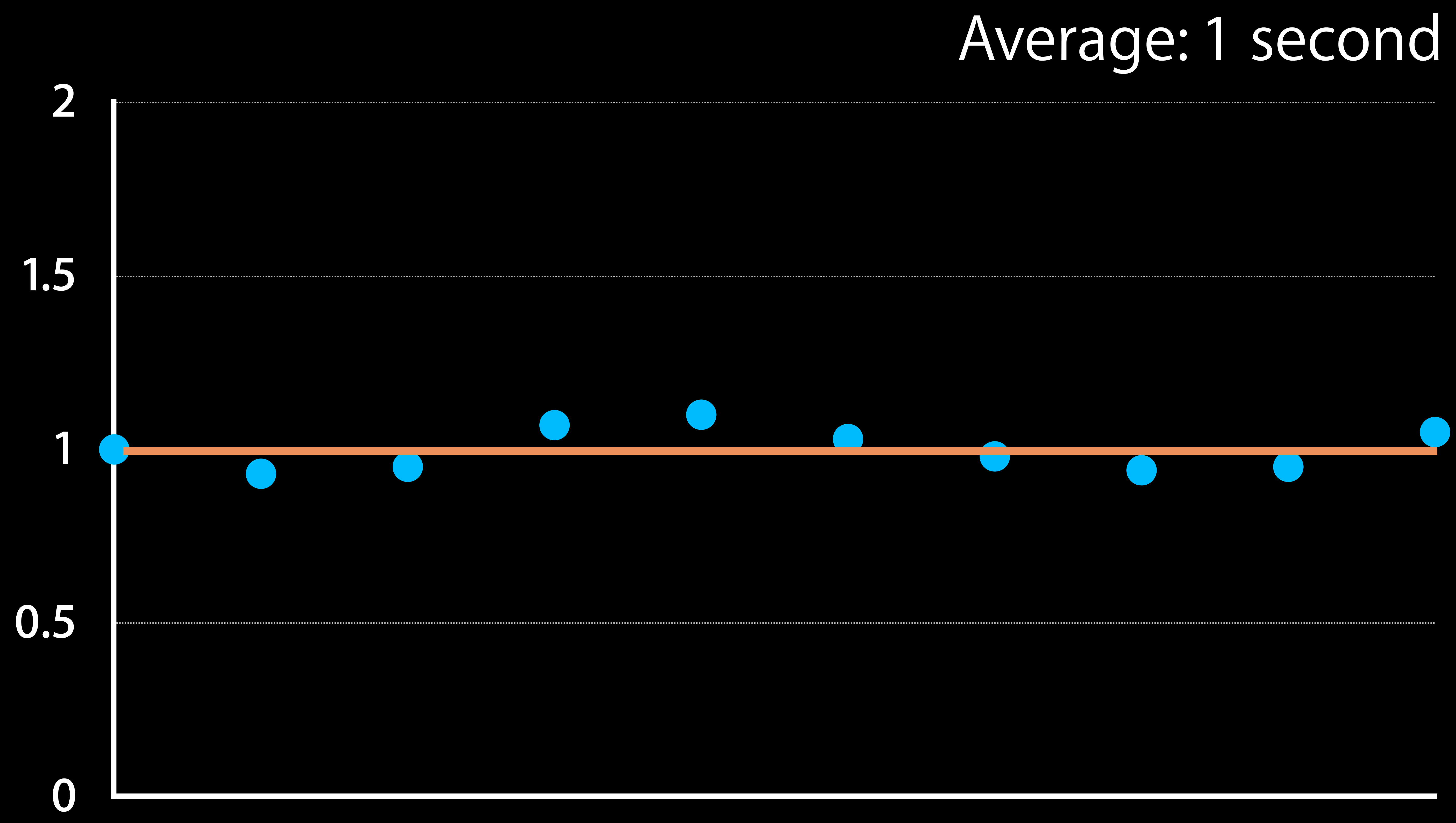
Test target macApp3Tests
✓ Scheme macApp3 | Destination My Mac 64-bit
▼ ✓ Run test suite All tests 2 out of 2 tests passed, 4.241 seconds
▼ ✓ Run test suite macApp3Tests.xctest 2 out of 2 tests passed, 4.241 seconds
▼ ✓ Run test suite macApp3Tests 2 out of 2 tests passed, 4.241 seconds
✓ Run test case testPerformanceExample 1.357 seconds – [Time] average: 0.102 seconds, %RSD: 0.4%

```
Test Case '-[macApp3Tests testPerformanceExample]' started.
Test Case '-[macApp3Tests testPerformanceExample]' started.
/Users/bcallaha/Desktop/macApp3/macApp3Tests/macApp3Tests.m:57: Test Case '-[macApp3Tests
  testPerformanceExample]' measured [Time, seconds] average: 0.102, relative standard deviation:
  0.396%, values: [0.101267, 0.101385, 0.102166, 0.101393, 0.102161, 0.102251, 0.102259, 0.102260,
  0.102227, 0.102247], performanceMetricID:com.apple.XCTPerformanceMetric_WallClockTime,
  baselineName: "Local Baseline", baselineAverage: 0.100, maxPercentRegression: 10.000%,
  maxPercentRelativeStandardDeviation: 10.000%, maxRegression: 0.100, maxStandardDeviation: 0.100
```

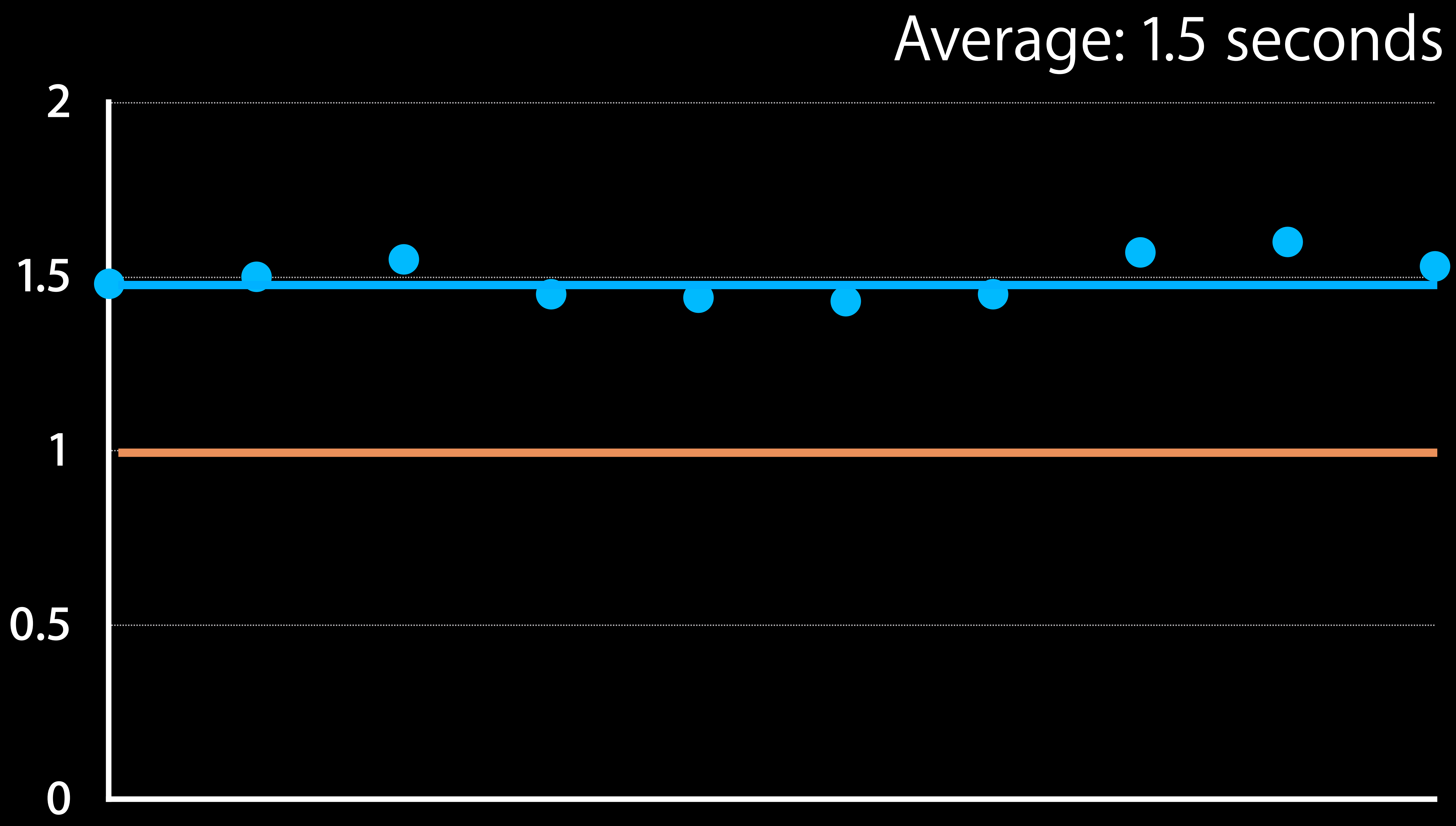
Using Baseline Average



Using Baseline Average



Using Baseline Average



Using Baseline Average

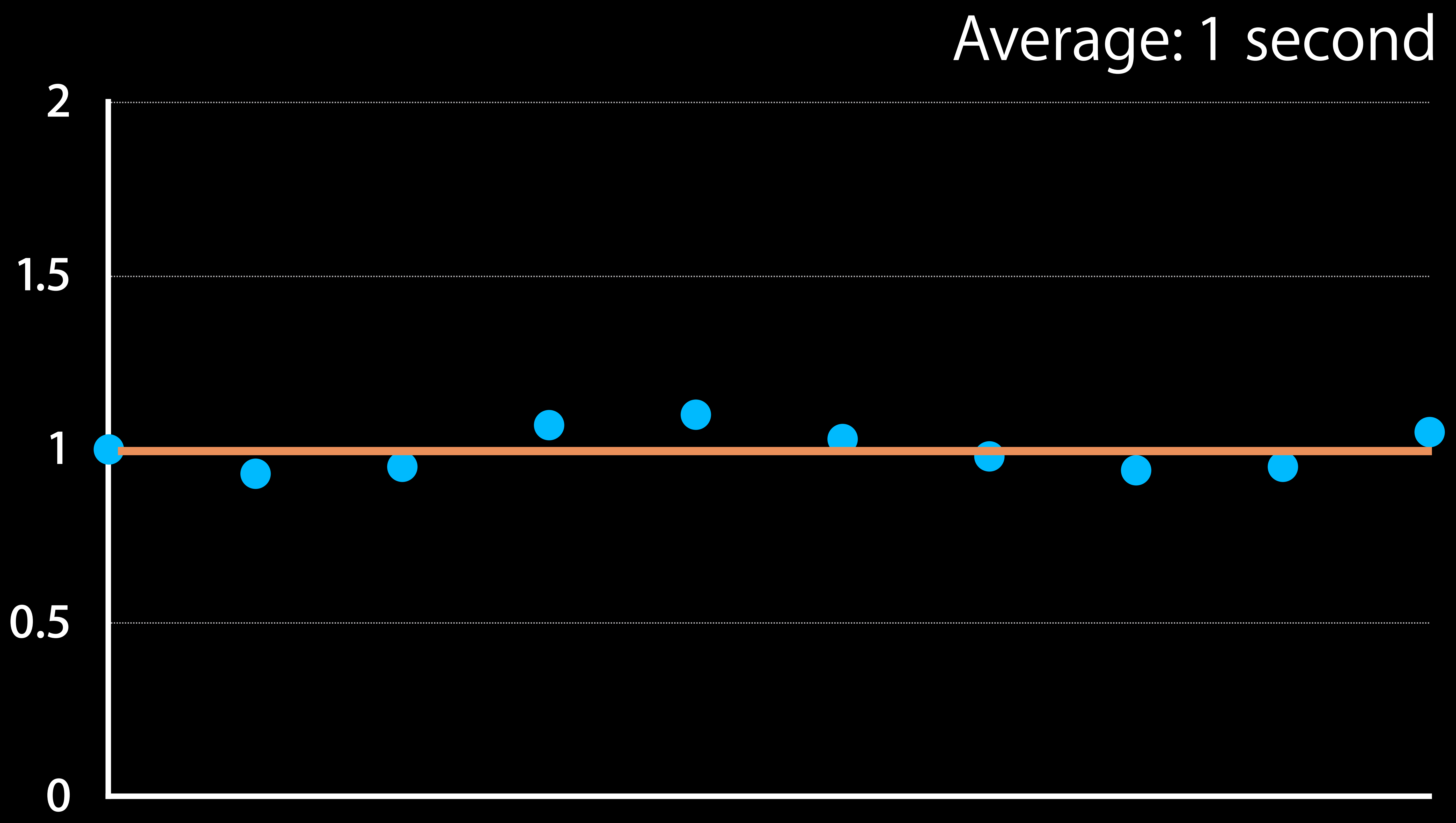
Fail if (Average–Baseline Average) is more than 10% of Baseline Average

Ignore if (Average–Baseline Average) less than 0.1 seconds

基线标准偏差超过百分之十规定为测试不通过

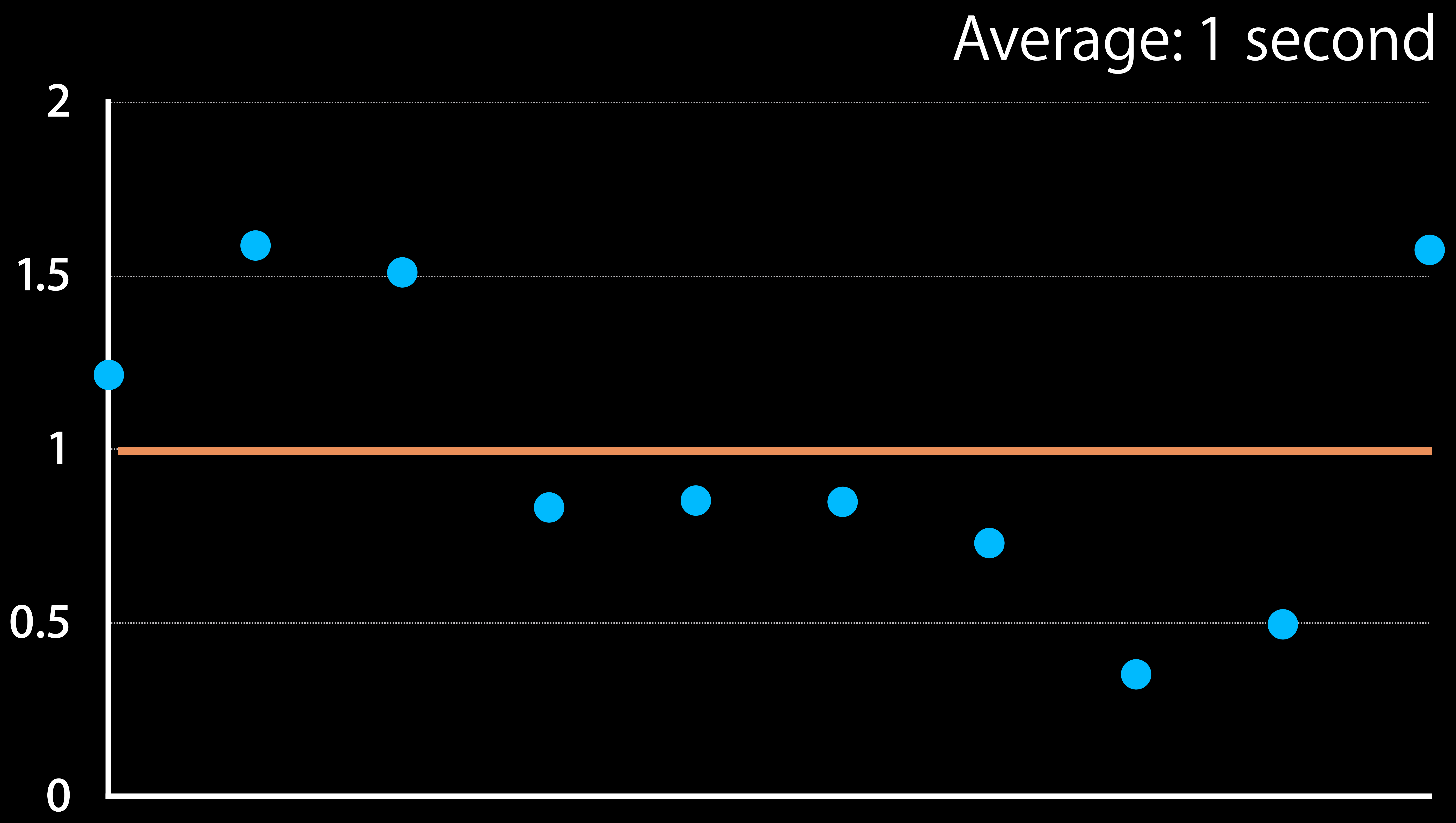
基线值小于0.1秒不与考虑

Is Average Enough?



Is Average Enough?

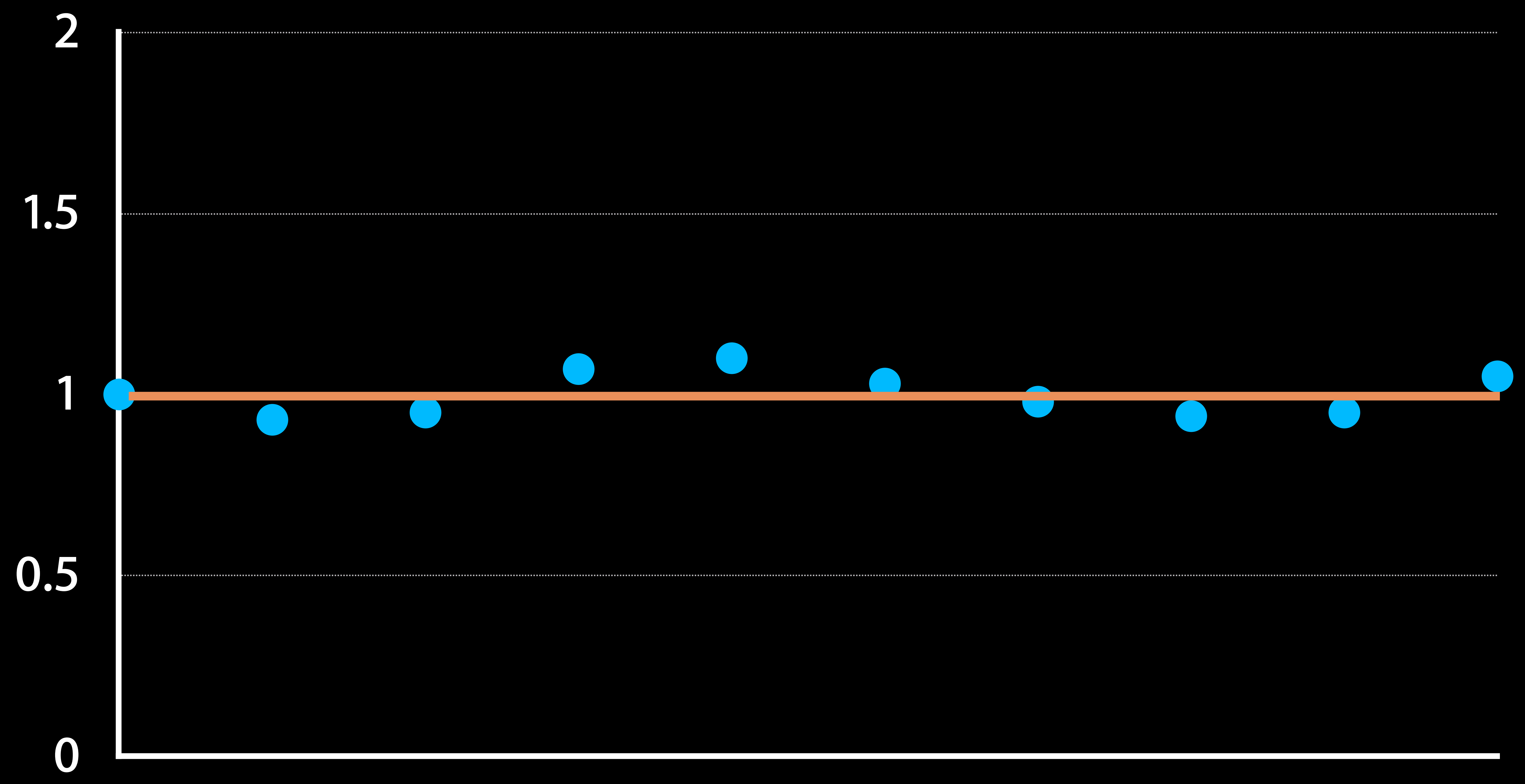
Problem?



Detecting Variance

Average: 1 second

Standard Deviation: 6%

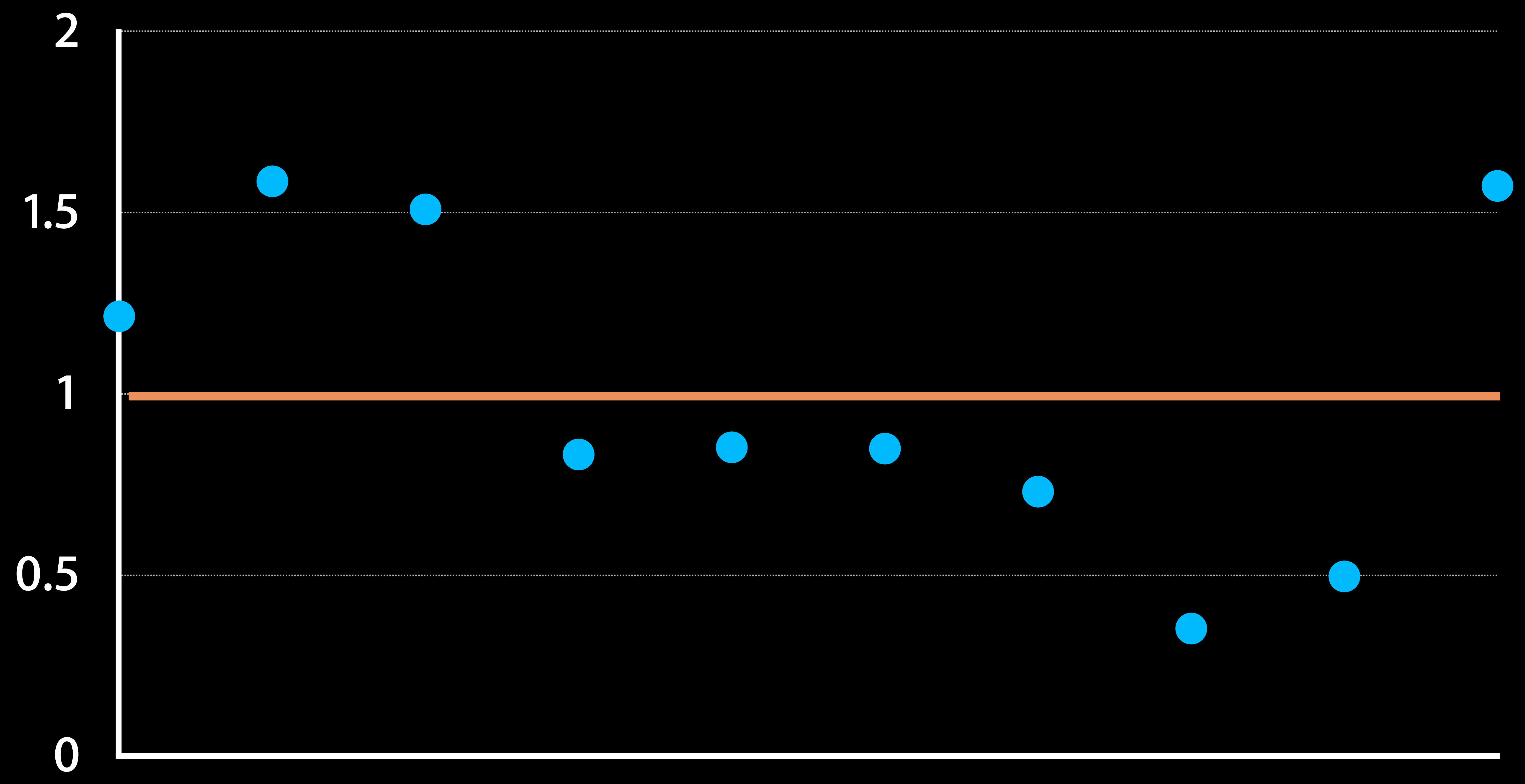


Detecting Variance

Problem?

Average: 1 second

Standard Deviation: 40%



Using Standard Deviation (STDDEV)

Fail if STDDEV is more than 10% of Average (adjustable)

Ignore if STDDEV is less than 0.1 seconds

Excessive STDDEV

Block being measured

- Does file I/O or network I/O
- Doesn’t do the same work each time it’s called

System is busy with other processes

Detecting Regressions

1. If test has no Baseline Average, done
2. If STDDEV >0.1 seconds and >10%, fail
3. If (Average–Baseline Average) >0.1 seconds and >10%, fail
4. Else, pass

Measuring Precisely 测量精度

Only measure code you think that’s important to you
只测试对你重要的代码

Measuring Precisely

Example

```
- (void)testUseFileHandlePerformance
{
    [self measureBlock:^{

        NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
        XCTAssertNotNil(fileHandle);

        UseFileHandle(fileHandle);

        [fileHandle closeFile];
    }];
}
```

这些代码都重要吗?

Measuring Precisely

Example

```
- (void)testUseFileHandlePerformance
{
    NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
    XCTAssertNotNil(fileHandle);

    [self measureBlock:^{

        UseFileHandle(fileHandle);

        [fileHandle closeFile];
    }];
}
```

这些代码都重要吗?

Measuring Precisely

Example

```
- (void)testUseFileHandlePerformance
{
    NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
    XCTAssertNotNil(fileHandle);

    [self measureBlock:^{

        UseFileHandle(fileHandle);
    }];

   [fileHandle closeFile];
}
```

这才是我们想要的

Measuring Precisely

More XCTestCase APIs

```
- (void)measureMetrics:(NSArray *)metrics automaticallyStartMeasuring:
(BOOL)automaticallyStartMeasuring withBlock:(void (^)(void))block;
```

Use this to measure part of the block 使用这个来测试block内的一部分

Measures passed in metrics 测试通过的计量

Currently supports one metric: `XCTPerformanceMetric_WallClockTime`
当前支持一个计量

Measuring Precisely

More XCTestCase APIs

```
- (void)startMeasuring;
- (void)stopMeasuring;
```

Isolate part of the block to measure 隔离block的一部分来进行测量

May be called once per block invocation 每次调用block的时候调用

-`startMeasuring` requires `automaticallyStartMeasuring:NO`

-`stopMeasuring` called automatically after block

Measuring Precisely

Example

```
- (void)testUseFileHandlePerformance
{
    [self measureBlock:^{

        NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
        XCTAssertNotNil(fileHandle);

        UseFileHandle(fileHandle);

        [fileHandle closeFile];
    }];
}
```

原来的粗糙的性能测试写法!

Measuring Precisely

Example

注意参数

```
- (void)testUseFileHandlePerformance
{
    [self measureMetrics:@[XCTPerformanceMetric_WallClockTime]
          automaticallyStartMeasuring:NO forBlock:^{

        NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
        XCTAssertNotNil(fileHandle);

        UseFileHandle(fileHandle);

        [fileHandle closeFile];
    }];
}
```

使用新的计量API是这样的

Measuring Precisely

Example

```
- (void)testUseFileHandlePerformance
{
    [self measureMetrics:@[XCTPerformanceMetric_WallClockTime]
          automaticallyStartMeasuring:NO forBlock:^{

        NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
        XCTAssertNotNil(fileHandle);

        [self startMeasuring];
        UseFileHandle(fileHandle);
        [self stopMeasuring];

        [fileHandle closeFile];
    }];
}
```

Measuring Precisely

Example

```
- (void)testUseFileHandlePerformance
{
    [self measureMetrics:@[XCTPerformanceMetric_WallClockTime]
          automaticallyStartMeasuring:NO forBlock:^{

        NSFileHandle *fileHandle = [NSFileHandle fileHandleForReadingAtPath:PATH];
        XCTAssertNotNil(fileHandle);

        [self startMeasuring];
        UseFileHandle(fileHandle);
        [self stopMeasuring];

        [fileHandle closeFile];
    }];
}
```

只测试block中的这一部分

Performance Testing

Wrap-Up

性能测试总结!

Use new APIs to measure performance 新的性能测试的API

Set Baseline to detect regressions 设置基线回归

Use Standard Deviation to show spread of measurements 使用标准偏差

Use Instruments to profile tests 使用仪表盘测试视图

More Information

Dave DeLong
Developer Tools Evangelist
delong@apple.com

Related Sessions

● Continuous Integration with Xcode 6	Marina	Thursday 2:00PM

Labs

● Continuous Integration Lab	Tools Lab C	Thursday 2:00PM

WWDC14