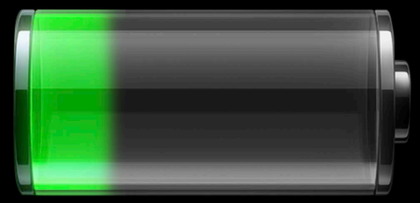
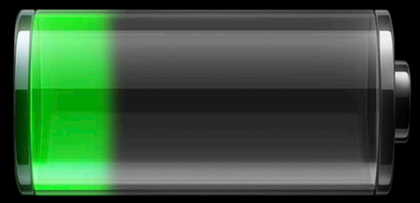
#WWDC19

Improving Battery Life and Performance

Phillip Azar, Software Engineer
Sastry Vadlamani, Software Engineer
Ashish Patro, Software Engineer
Anshul Dawra, Software Engineer





Tools overview

Metrics overview

Deep dives and Demos

Summary

Tools overview

Metrics overview

Deep dives and Demos

Summary

The Development Process, Abridged

Building your app comes in phases

The Development Process, Abridged

Building your app comes in phases

Development
and Testing

The Development Process, Abridged

Building your app comes in phases

Development
and Testing

Beta

The Development Process, Abridged

Building your app comes in phases

Development and Testing

Beta

Public Release

Every step is important

Development
and Testing

Beta

Public Release

Development
and Testing

Beta

Public Release

Development
and Testing

Beta

Public Release

Development
and Testing

Beta

Public Release

There are gaps we can fill

gaps

Metrics

New Tools for Gathering Metrics

NEW

New Tools for Gathering Metrics

NEW

XCTest Metrics

New Tools for Gathering Metrics

NEW

XCTest Metrics

- Performance of measure blocks

New Tools for Gathering Metrics

NEW

XCTest Metrics

- Performance of measure blocks

MetricKit

New Tools for Gathering Metrics

NEW

XCTest Metrics

- Performance of measure blocks

MetricKit

- Framework for battery and performance metrics collection

New Tools for Gathering Metrics

NEW

XCTest Metrics

- Performance of measure blocks

MetricKit

- Framework for battery and performance metrics collection

Xcode Metrics Organizer

New Tools for Gathering Metrics

NEW

XCTest Metrics

- Performance of measure blocks

MetricKit

- Framework for battery and performance metrics collection

Xcode Metrics Organizer

- Aggregated battery, performance, and I/O metrics in Xcode

Development and Testing

Beta

Public Release

Development
and Testing

Beta

Public Release

XCTest Metrics

Development
and Testing

Beta

Public Release

MetricKit

Development and Testing

Beta

Public Release

Xcode Metrics Organizer

Development and Testing

Beta

Public Release

XCTest Metrics

MetricKit

Xcode Metrics Organizer

More metrics at every stage of development

Tools overview

Metrics overview

Deep dives and Demos

Summary

Metrics Are the Key

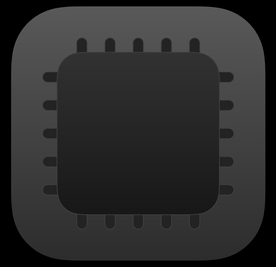
This year, two categories of metrics

Battery

Performance

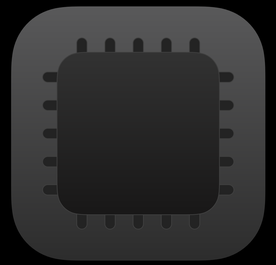
Battery Metrics

Battery Metrics



Processing

Battery Metrics



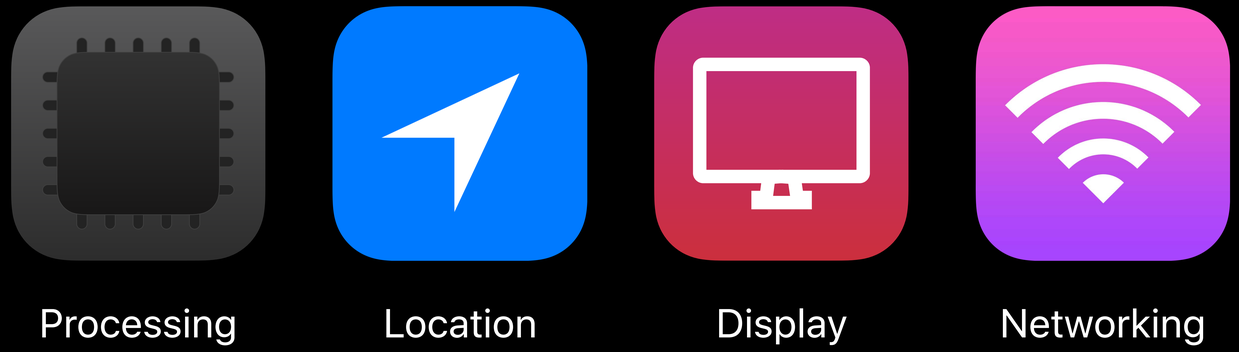
Processing

Location

Battery Metrics

Processing Location Display

Battery Metrics



Processing Location Display Networking

Battery Metrics

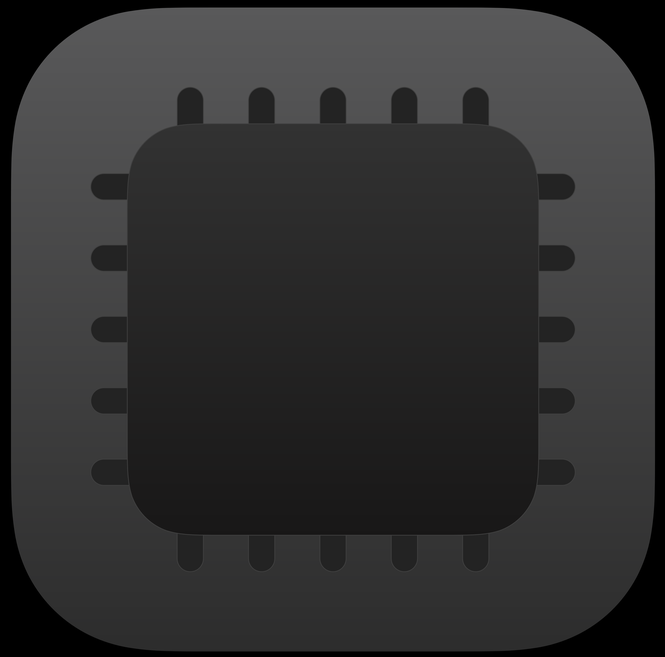
Processing Location Display Networking Accessories

Battery Metrics

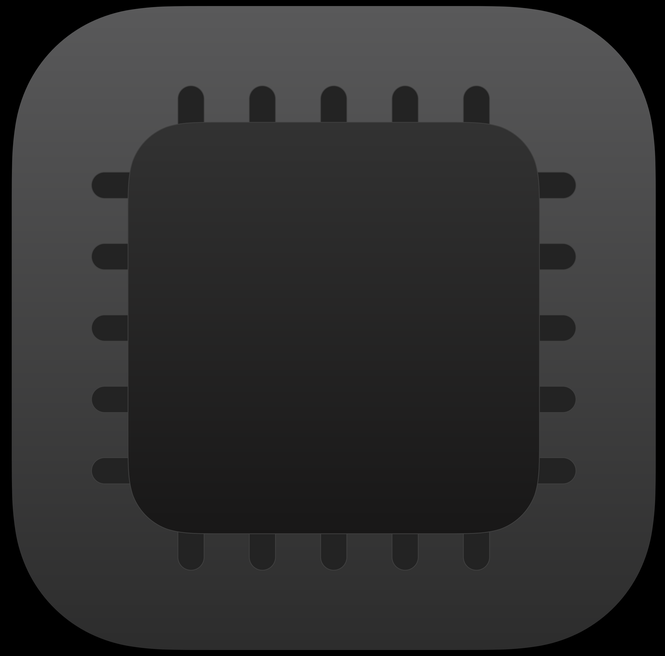
Processing Location Display Networking Accessories Multimedia

Battery Metrics

Battery Metrics

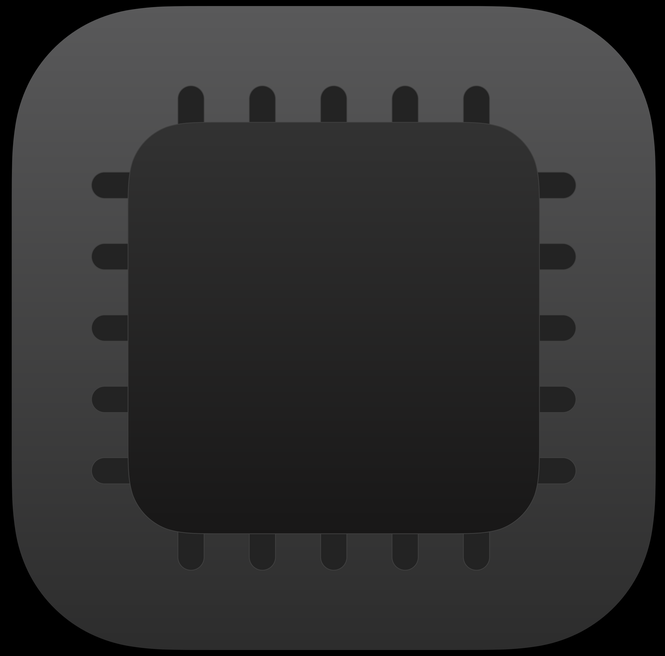


Processing Metrics



Processing Metrics

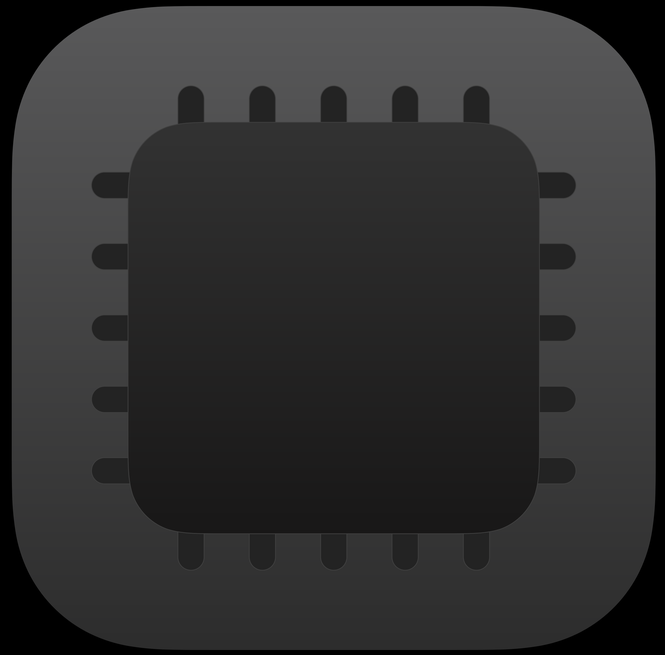
CPU time, GPU time, etc



Processing Metrics

CPU time, GPU time, etc

Use these metrics to understand workloads

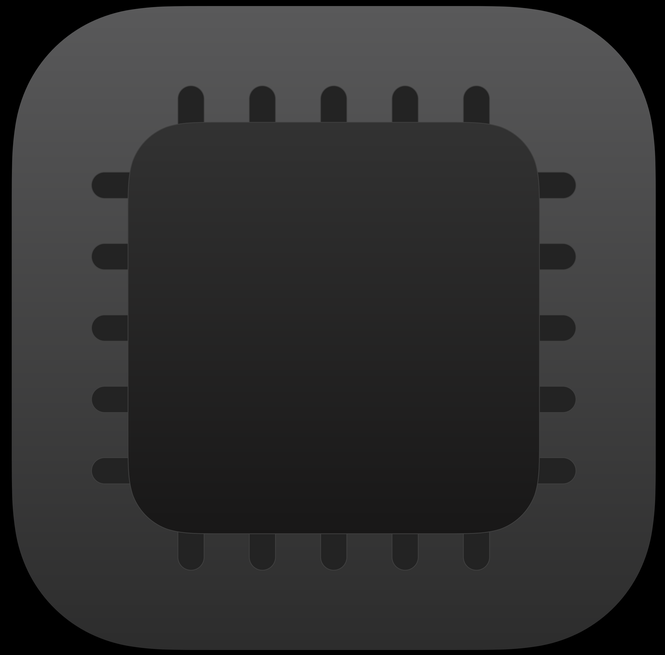


Processing Metrics

CPU time, GPU time, etc

Use these metrics to understand workloads

- CPU spinners

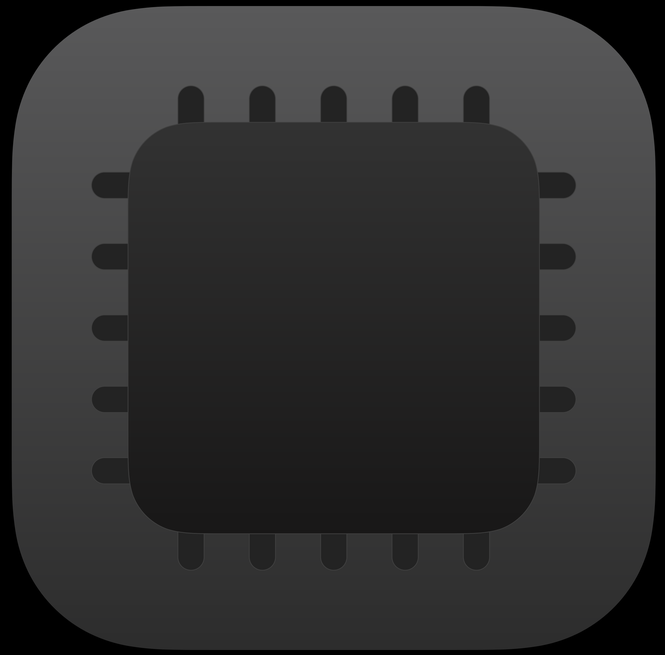


Processing Metrics

CPU time, GPU time, etc

Use these metrics to understand workloads

- CPU spinners
- Unexpected rendering



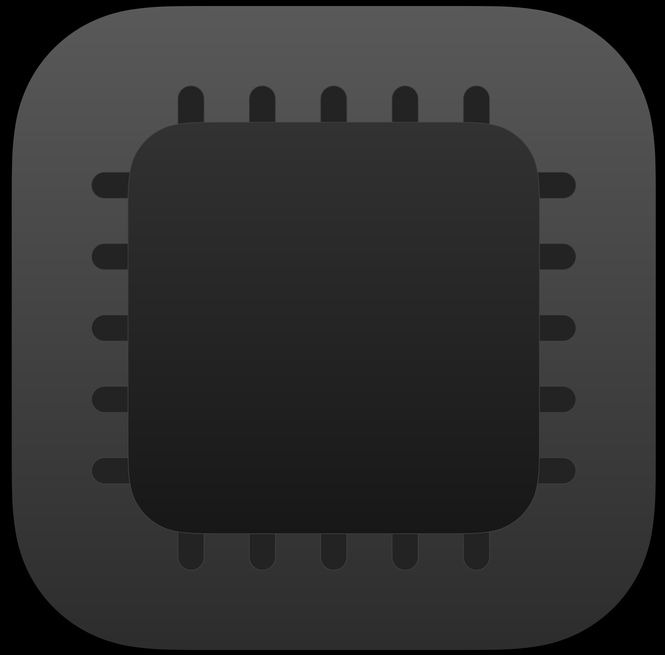
Processing Metrics

CPU time, GPU time, etc

Use these metrics to understand workloads

- CPU spinners
- Unexpected rendering

Compare algorithmic efficiency of features



Location Metrics

Location Metrics

Cumulative usage time, background time, etc.

Location Metrics

Cumulative usage time, background time, etc.

Use these to understand location usage

Location Metrics

Cumulative usage time, background time, etc.

Use these to understand location usage

- Identify cases where location is left on

Location Metrics

Cumulative usage time, background time, etc.

Use these to understand location usage

- Identify cases where location is left on
- Validate location accuracy usage

Display Metrics

Display Metrics

Average Pixel Luminance

Display Metrics

Average Pixel Luminance

Color of UI on OLED displays impacts energy

Display Metrics

Average Pixel Luminance

Color of UI on OLED displays impacts energy

- Lighter colors = more energy (high APL)

Display Metrics

Average Pixel Luminance

Color of UI on OLED displays impacts energy

- Lighter colors = more energy (high APL)
- Darker colors = less energy (low APL)

Networking Metrics

Networking Metrics

Upload and download bytes, connectivity, etc.

Networking Metrics

Upload and download bytes, connectivity, etc.

Optimize networking usage whenever possible

Networking Metrics

Upload and download bytes, connectivity, etc.

Optimize networking usage whenever possible

- Validate expected upload/download

Networking Metrics

Upload and download bytes, connectivity, etc.

Optimize networking usage whenever possible

- Validate expected upload/download
- Understand impact of poor connectivity

Battery Metrics

Performance Metrics

Performance Metrics

Hangs

Performance Metrics

Hangs

Disk

Performance Metrics

Hangs

Disk

Application Launch

Performance Metrics

Hangs

Disk

Application Launch

Memory

Performance Metrics

Hangs

Disk

Application Launch

Memory

Custom Intervals

Performance Metrics

Hangs

Disk

Application Launch

Memory

Custom Intervals

Hang Metrics

Hang Metrics

Histogram of application hang time

Hang Metrics

Histogram of application hang time

Huge user impact

Hang Metrics

Histogram of application hang time

Huge user impact

- Move work off the main thread if possible

Hang Metrics

Histogram of application hang time

Huge user impact

- Move work off the main thread if possible
- Utilize dispatches and queues for async tasks

Disk Metrics

Disk Metrics

Disk Logical Writes

Disk Metrics

Disk Logical Writes

Quantify disk usage with these metrics

Disk Metrics

Disk Logical Writes

Quantify disk usage with these metrics

- Identify instances of unexpected disk writes

Disk Metrics

Disk Logical Writes

Quantify disk usage with these metrics

- Identify instances of unexpected disk writes
- Validate coalescing strategies

Application Launch Metrics

Application Launch Metrics

Launch time histogram, resume time histogram

Application Launch Metrics

Launch time histogram, resume time histogram

Quantify launch and resume with these metrics

Application Launch Metrics

Launch time histogram, resume time histogram

Quantify launch and resume with these metrics

- Understand impact of launch activities

Application Launch Metrics

Launch time histogram, resume time histogram

Quantify launch and resume with these metrics

- Understand impact of launch activities
- See differences between launch and resume

Application Launch Metrics

Launch time histogram, resume time histogram

Quantify launch and resume with these metrics

- Understand impact of launch activities
- See differences between launch and resume

Optimizing App Launch

Friday, 4:20

Memory Metrics

Memory Metrics

Average Suspended Memory, Peak Memory

Memory Metrics

Average Suspended Memory, Peak Memory

Memory management can impact launch times

Memory Metrics

Average Suspended Memory, Peak Memory

Memory management can impact launch times

Use these metrics to understand memory usage

Memory Metrics

Average Suspended Memory, Peak Memory

Memory management can impact launch times

Use these metrics to understand memory usage

- Identify hard to reproduce memory leaks

Memory Metrics

Average Suspended Memory, Peak Memory

Memory management can impact launch times

Use these metrics to understand memory usage

- Identify hard to reproduce memory leaks
- Reduce average memory on suspend

Performance Metrics

Hangs

Disk

Application Launch

Memory

Custom Intervals

Development and Testing

Beta

Public Release

XCTest Metrics

MetricKit

Xcode Metrics Organizer

Development and Testing

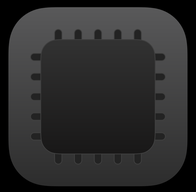
Beta

Public Release

XCTest Metrics

MetricKit

Xcode Metrics Organizer



NEW

Tools overview

Metrics overview

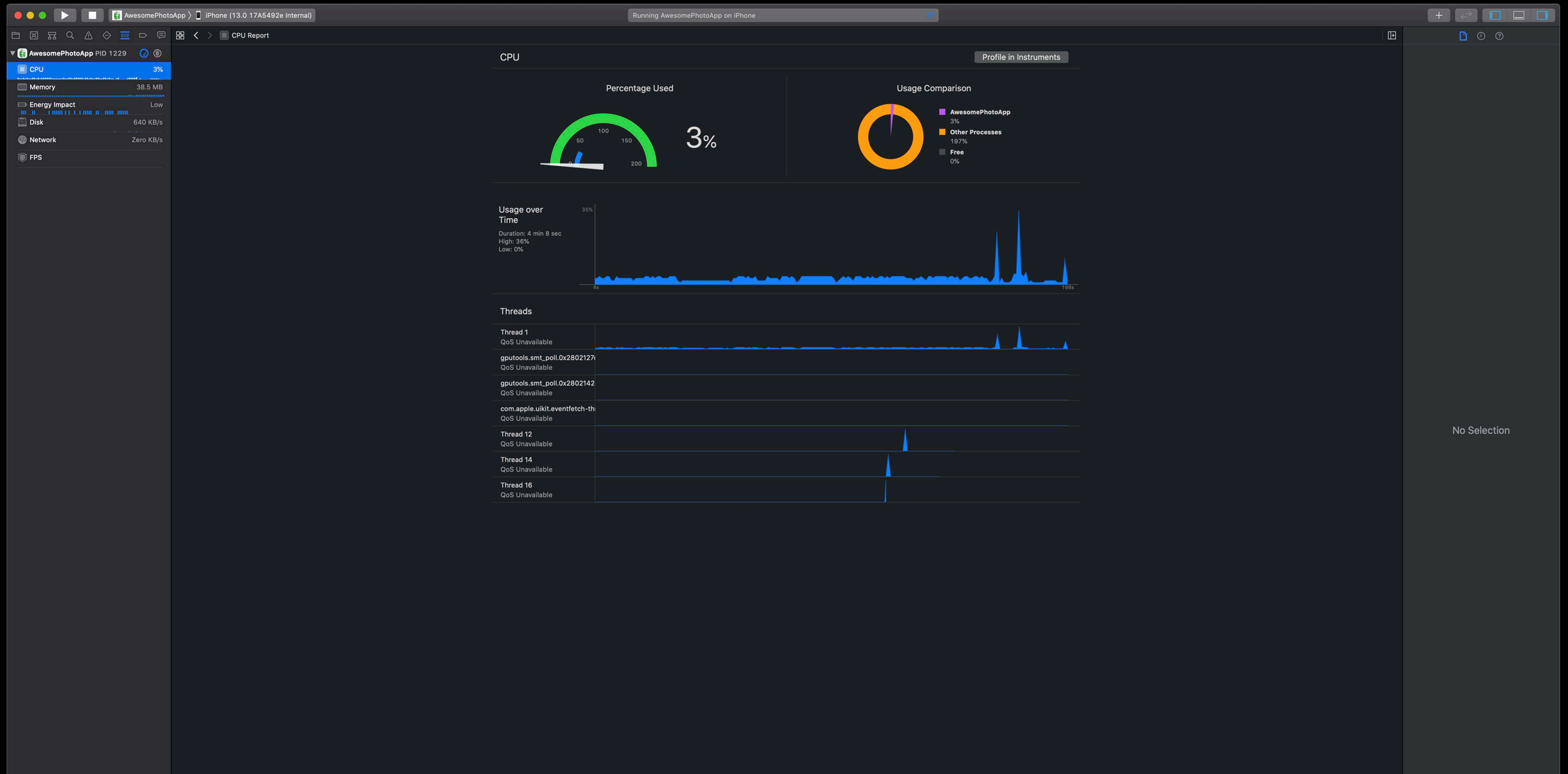
Deep dives and Demos

Summary

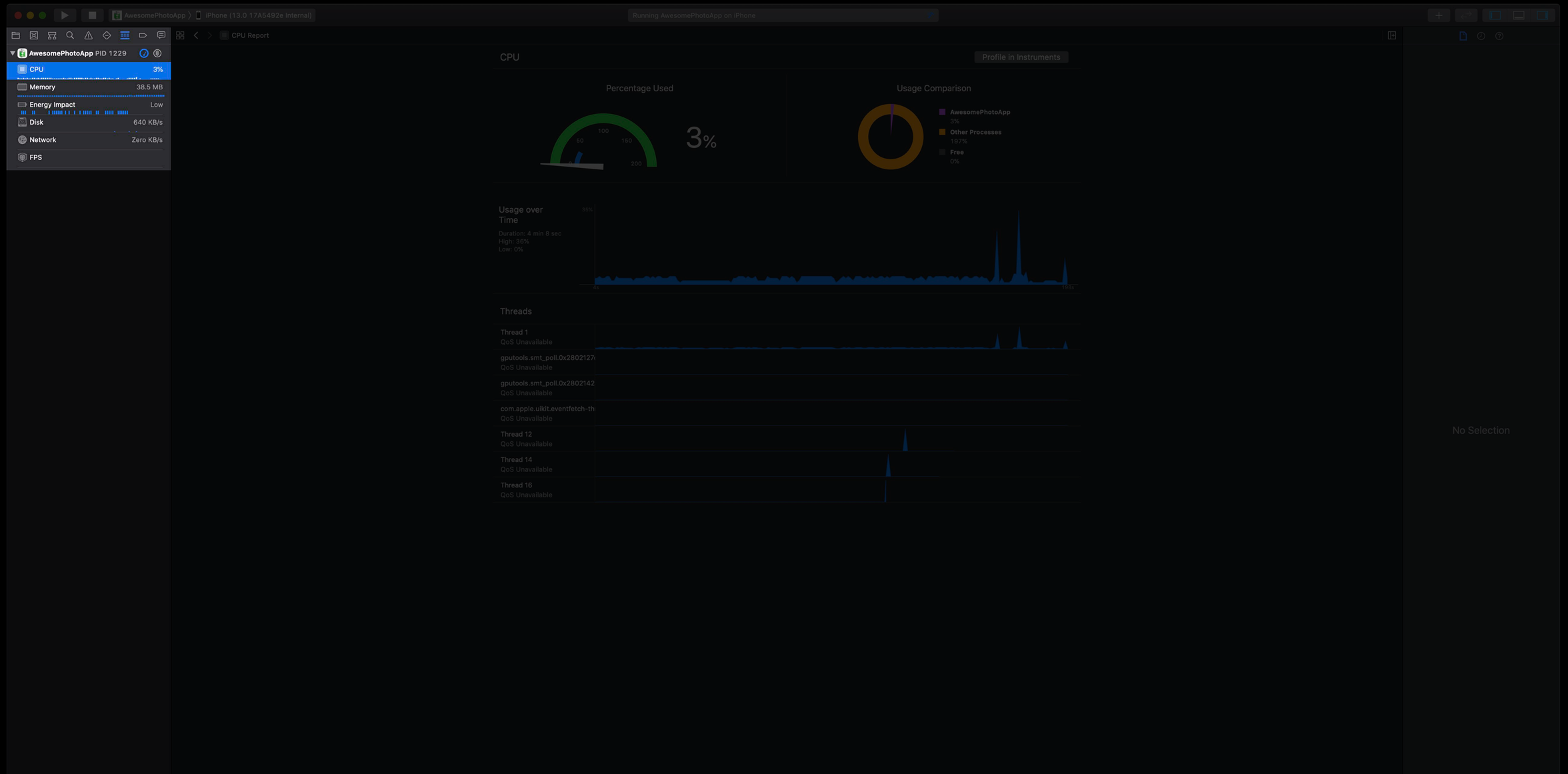
Measuring App Impact during Development and Testing

Sastry Vadlamani, Software Engineer

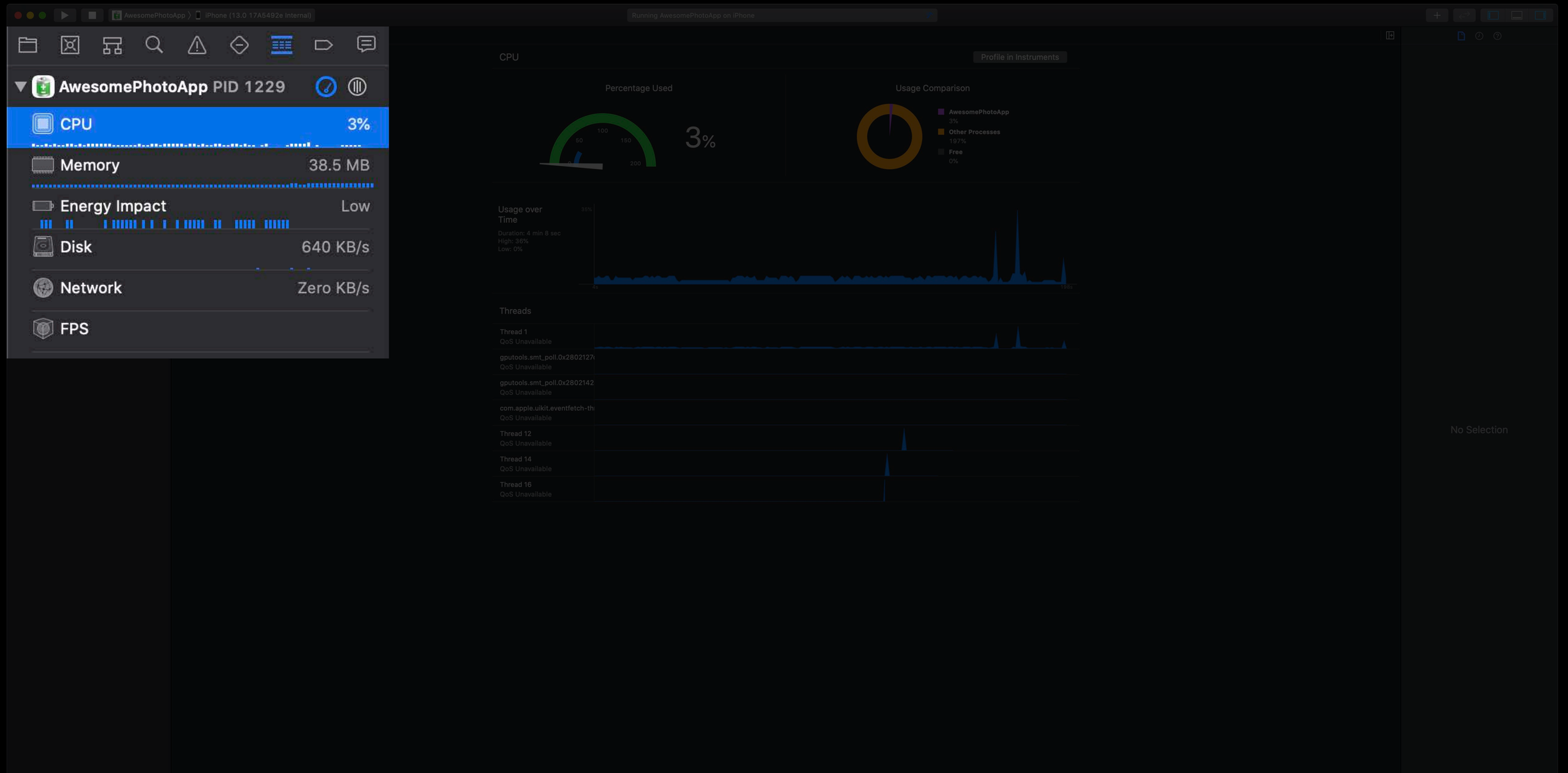
Xcode Debug Navigator



Xcode Debug Navigator



Xcode Debug Navigator



Templates in Instruments

Templates in Instruments

Allocations

Templates in Instruments

Allocations

Time Profiler

Templates in Instruments

Allocations

Time Profiler

System Usage

Templates in Instruments

Allocations Time Profiler System Usage Energy Log

Collecting Metrics Using XCTest

Collecting Metrics Using XCTest

Catch regressions with baselines

Collecting Metrics Using XCTest

NEW

Catch regressions with baselines

New performance metrics

```
// This test measures time, memory, and CPU impact

func testPhotoUploadPerformance() {
    let app = XCUIApplication()
    measure() {


                        app.buttons["Apply Effect"].tap()
                        app.dialogs["alert"].buttons["OK"].tap()
    }
}
```

```swift
// This test measures time, memory, and CPU impact

func testPhotoUploadPerformance() {
    let app = XCUIApplication()
    measure(metrics: [XCTClockMetric(),
                      XCTMemoryMetric(application: app),
                      XCTCPUMetric(application: app)]) {
                        app.buttons["Apply Effect"].tap()
                        app.dialogs["alert"].buttons["OK"].tap()
    }
}
```

```swift
// This test measures time, memory, and CPU impact

func testPhotoUploadPerformance() {
    let app = XCUIApplication()
    measure(metrics: [XCTClockMetric(),
                      XCTMemoryMetric(application: app),
                      XCTCPUMetric(application: app)]) {
                        app.buttons["Apply Effect"].tap()
                        app.dialogs["alert"].buttons["OK"].tap()
    }
}
```

```
// This test measures your Application's Launch Time

func testApplicationLaunchTime() {
    measure(metrics: [XCTOSSignpostMetric.applicationLaunch]) {
        XCUIApplication().launch()
    }
}
```

Demo

XCTest Metrics

XCTest Metrics Demo Takeaway

XCTest Metrics Demo Takeaway

CPU, memory, storage, clock and OSSignpost

XCTest Metrics Demo Takeaway

CPU, memory, storage, clock and OSSignpost

Custom metrics

XCTest Metrics Demo Takeaway

CPU, memory, storage, clock and OSSignpost

Custom metrics

A/B testing

XCTest Metrics Demo Takeaway

CPU, memory, storage, clock and OSSignpost

Custom metrics

A/B testing

Daily dev and continuous integration

Measuring App Impact in the Field

Ashish Patro, Software Engineer

Benefits of Field Metrics

Benefits of Field Metrics

Leverage beta and public population

Benefits of Field Metrics

Leverage beta and public population

Uncover issues missed during on-desk testing

Benefits of Field Metrics

Leverage beta and public population

Uncover issues missed during on-desk testing

Compare metrics with previous app versions

Benefits of Field Metrics

Leverage beta and public population

Uncover issues missed during on-desk testing

Compare metrics with previous app versions

Impact of new features and A/B testing

Introducing MetricKit

NEW

Introducing MetricKit

NEW

Framework for collecting battery + performance metrics

Introducing MetricKit

NEW

Framework for collecting battery + performance metrics

Ability to collect metrics around your critical code sections

Introducing MetricKit

NEW

Framework for collecting battery + performance metrics

Ability to collect metrics around your critical code sections

Data aggregation designed to protect user privacy

Introducing MetricKit

NEW

Framework for collecting battery + performance metrics

Ability to collect metrics around your critical code sections

Data aggregation designed to protect user privacy

Easy to adopt

```swift
// Adopting MetricKit to receive metrics
import MetricKit

// 1. Conform to MXMetricManagerSubscriber protocol
class MySubscriber: NSObject, MXMetricManagerSubscriber {

    var metricManager: MXMetricManager?
    override init() {
        super.init()
        // 2. Initialize MetricManager
        metricManager = MXMetricManager.shared

        // 3. Subscribe for metrics
        metricManager?.add(self)
    }

    deinit {
        metricManager?.remove(self)
    }
```

```swift
// Adopting MetricKit to receive metrics
import MetricKit

// 1. Conform to MXMetricManagerSubscriber protocol
class MySubscriber: NSObject, MXMetricManagerSubscriber {

    var metricManager: MXMetricManager?
    override init() {
        super.init()
        // 2. Initialize MetricManager
        metricManager = MXMetricManager.shared

        // 3. Subscribe for metrics
        metricManager?.add(self)
    }

    deinit {
        metricManager?.remove(self)
    }
```

```swift
// Adopting MetricKit to receive metrics
import MetricKit

// 1. Conform to MXMetricManagerSubscriber protocol
class MySubscriber: NSObject, MXMetricManagerSubscriber {

    var metricManager: MXMetricManager?
    override init() {
        super.init()
        // 2. Initialize MetricManager
        metricManager = MXMetricManager.shared

        // 3. Subscribe for metrics
        metricManager?.add(self)
    }

    deinit {
        metricManager?.remove(self)
    }
```

```
// Adopting MetricKit to receive metrics

// 4. Implement delegate method
    func didReceive(_ payload: [MXMetricPayload]) {
        for metricPayload in payload {
            // 5. Consume metric payloads
            ……
        }
    }
```

```
// Adopting MetricKit to receive metrics

// 4. Implement delegate method
    func didReceive(_ payload: [MXMetricPayload]) {
        for metricPayload in payload {
            // 5. Consume metric payloads
            ……
        }
    }
```

Receiving Aggregate Metrics

Receiving Aggregate Metrics

Receiving Aggregate Metrics

Receiving Aggregate Metrics

Receiving Aggregate Metrics

Daily aggregated metrics
(24 hr. intervals)

Measuring critical code sections

Metrics for Critical Code Sections

Metrics for Critical Code Sections

Metrics for Critical Code Sections

Take Photo

Apply Effects

Metrics for Critical Code Sections

Take Photo

Apply Effects

Save Photo

Metrics for Critical Code Sections

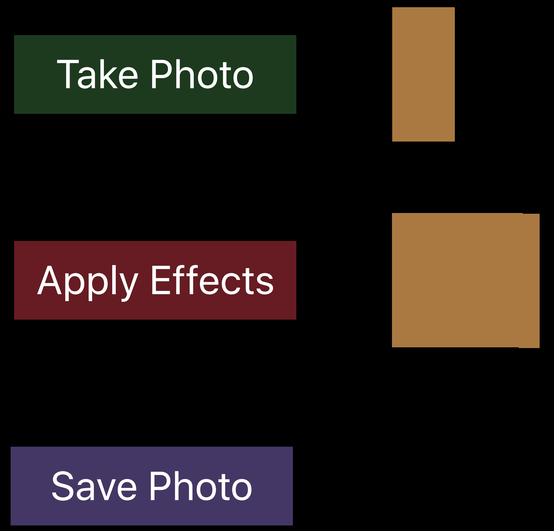
Take Photo

Apply Effects

Save Photo

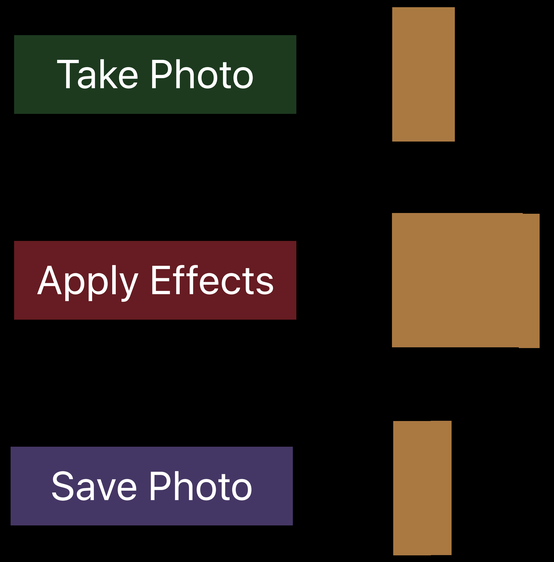
Daily metrics per critical section

Metrics for Critical Code Sections



Daily metrics per critical section

Metrics for Critical Code Sections



Daily metrics per critical section

Introducing MetricKit's mxSignposts

NEW

Introducing MetricKit's mxSignposts

NEW

Signpost-style API available directly in MetricKit

Introducing MetricKit's mxSignposts

NEW

Signpost-style API available directly in MetricKit

Collect metrics around your critical code sections

```
// Example: Collect metrics for critical code sections using mxSignposts
// Each mxSignpost (build over os_signposts) snapshots CPU time, memory and logical Writes
// Currently, custom metadata isn't supported with mxSignposts

// 1. Create log handle using MetricKit's makeLogHandle method
let photosLogHandle : OSLog = MXMetricManager.makeLogHandle(category: "Photos")

// 2. Drop mxSignpost around critical code sections
mxSignpost(.begin, log: photosLogHandle,  name: "SavePhoto")

SavePhoto() // Application code

mxSignpost(.end, log: photosLogHandle,  name: "SavePhoto")
```

```
// Example: Collect metrics for critical code sections using mxSignposts
// Each mxSignpost (build over os_signposts) snapshots CPU time, memory and logical Writes
// Currently, custom metadata isn't supported with mxSignposts

// 1. Create log handle using MetricKit's makeLogHandle method
let photosLogHandle : OSLog = MXMetricManager.makeLogHandle(category: "Photos")

// 2. Drop mxSignpost around critical code sections
mxSignpost(.begin, log: photosLogHandle,  name: "SavePhoto")

SavePhoto() // Application code

mxSignpost(.end, log: photosLogHandle,  name: "SavePhoto")
```

```
// Example: Collect metrics for critical code sections using mxSignposts
// Each mxSignpost (build over os_signposts) snapshots CPU time, memory and logical Writes
// Currently, custom metadata isn't supported with mxSignposts

// 1. Create log handle using MetricKit's makeLogHandle method
let photosLogHandle : OSLog = MXMetricManager.makeLogHandle(category: "Photos")

// 2. Drop mxSignpost around critical code sections
mxSignpost(.begin, log: photosLogHandle,  name: "SavePhoto")

SavePhoto() // Application code

mxSignpost(.end, log: photosLogHandle,  name: "SavePhoto")
```

```
// Example: Collect metrics for critical code sections using mxSignposts
// Each mxSignpost (build over os_signposts) snapshots CPU time, memory and logical Writes
// Currently, custom metadata isn't supported with mxSignposts

// 1. Create log handle using MetricKit's makeLogHandle method
let photosLogHandle : OSLog = MXMetricManager.makeLogHandle(category: "Photos")

// 2. Drop mxSignpost around critical code sections
mxSignpost(.begin, log: photosLogHandle,  name: "SavePhoto")

SavePhoto() // Application code

mxSignpost(.end, log: photosLogHandle,  name: "SavePhoto")
```

Demo

Adopting MetricKit

Example Scenario — Beta Testing Across Few Users

Taking a road trip with our Awesome Photo App!

Receiving Data from Field!

MetricKit summary from one beta user

Receiving Data from Field!

MetricKit summary from one beta user

Received metrics after 24 hrs.

Payload uploaded to server by application

Receiving Data from Field!

MetricKit summary from one beta user

Received metrics after 24 hrs.

Payload uploaded to server by application

****** 24 Hour Metric Summary ******

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2
11 ms to 20 ms: 1

Launch Time (Time To First Draw) Histogram
151 ms to 160 ms: 1
171 ms to 180 ms: 1
191 ms to 200 ms: 1
201 ms to 210 ms: 2
241 ms to 250 ms: 1
261 ms to 270 ms: 1
381 ms to 390 ms: 1
2611 ms to 2620 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Receiving Data from Field!

MetricKit summary from one beta user

Let’s identify some hotspots now!

**** 24 Hour Metric Summary ****

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2
11 ms to 20 ms: 1

Launch Time (Time To First Draw) Histogram
151 ms to 160 ms: 1
171 ms to 180 ms: 1
191 ms to 200 ms: 1
201 ms to 210 ms: 2
241 ms to 250 ms: 1
261 ms to 270 ms: 1
381 ms to 390 ms: 1
2611 ms to 2620 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Receiving Data from Field!

MetricKit summary from one beta user

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall Location Usage (Ordered by battery impact)
- Best Accuracy For Navigation Time: 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Receiving Data from Field!

MetricKit summary from one beta user

Left location on!

High accuracy location
drains more battery

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall Location Usage (Ordered by battery impact)
- Best Accuracy For Navigation Time: 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Receiving Data from Field!

MetricKit summary from one beta user

**** 24 Hour Metric Summary ****

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2
171 ms to 180 ms: 1
201 ms to 210 ms: 2
381 ms to 390 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Receiving Data from Field!

MetricKit summary from one beta user

> 5 second hang durations!

Degrades user experience

**** 24 Hour Metric Summary ****

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Receiving Data from Field!

MetricKit summary from one beta user

Let’s dig deeper into critical code sections

Metrics summarized from mxSignposts

(CPU time, Memory, Logical writes, Durations)

**** 24 Hour Metric Summary ****

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2
11 ms to 20 ms: 1

Launch Time (Time To First Draw) Histogram
151 ms to 160 ms: 1
171 ms to 180 ms: 1
191 ms to 200 ms: 1
201 ms to 210 ms: 2
241 ms to 250 ms: 1
261 ms to 270 ms: 1
381 ms to 390 ms: 1
2611 ms to 2620 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Receiving Data from Field!

MetricKit summary from one beta user

Let’s dig deeper into critical code sections

Metrics summarized from mxSignposts

(CPU time, Memory, Logical writes, Durations)

****** Summary for mxSignpost intervals ******

Category	Name	Count
ImageProcessing	LoadPhoto	30
ImageProcessing	ApplyEffect	52
ImageProcessing	TakePhoto	20
ImageProcessing	SavePhoto	25
NetworkActivity	UploadPhoto	6

****** 24 Hour Metric Summary ******

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2
11 ms to 20 ms: 1

Launch Time (Time To First Draw) Histogram
151 ms to 160 ms: 1
171 ms to 180 ms: 1
191 ms to 200 ms: 1
201 ms to 210 ms: 2
241 ms to 250 ms: 1
261 ms to 270 ms: 1
381 ms to 390 ms: 1
2611 ms to 2620 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
681 ms to 690 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time : 739 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Receiving Data from Field!

MetricKit summary from one beta user

Let’s dig deeper into critical code sections

Metrics summarized from mxSignposts

(CPU time, Memory, Logical writes, Durations)

****** Summary for mxSignpost intervals ******

Category	Name	Count	CummulativeCPUTime
ImageProcessing	LoadPhoto	30	1 s
ImageProcessing	ApplyEffect	52	42 s
ImageProcessing	TakePhoto	20	1 s
ImageProcessing	SavePhoto	25	3 s
NetworkActivity	UploadPhoto	6	10 s

****** 24 Hour Metric Summary ******

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2
11 ms to 20 ms: 1

Launch Time (Time To First Draw) Histogram
151 ms to 160 ms: 1
171 ms to 180 ms: 1
191 ms to 200 ms: 1
201 ms to 210 ms: 2
241 ms to 250 ms: 1
261 ms to 270 ms: 1
381 ms to 390 ms: 1
2611 ms to 2620 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
581 ms to 600 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time : 726 s
Background Time : 23 s
- Background Audio Time : 0 s
- Background Location Time : 0 s

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Overall CPU Time: 78 s

Overall GPU Time: 0 s

Receiving Data from Field!

MetricKit summary from one beta user

> 50% CPU used by “ApplyEffect” feature

Optimizing feature can reduce battery usage

****** Summary for mxSignpost intervals ******

Category	Name	Count	CummulativeCPUTime
ImageProcessing	LoadPhoto	30	1 s
ImageProcessing	ApplyEffect	52	42 s
ImageProcessing	TakePhoto	20	1 s
ImageProcessing	SavePhoto	25	3 s
NetworkActivity	UploadPhoto	6	10 s

****** 24 Hour Metric Summary ******

Application Build Version: 8
Device Type: iPhone11,2
Os Version: iPhone OS 13.0 (17A492)
Region Format: US

Application Resume Time Histogram
0 ms to 10 ms: 2
11 ms to 20 ms: 1

Launch Time (Time To First Draw) Histogram
151 ms to 160 ms: 1
171 ms to 180 ms: 1
191 ms to 200 ms: 1
201 ms to 210 ms: 2
241 ms to 250 ms: 1
261 ms to 270 ms: 1
381 ms to 390 ms: 1
2611 ms to 2620 ms: 1

Application Hang Histogram
331 ms to 340 ms: 1
581 ms to 600 ms: 10
691 ms to 700 ms: 13
5111 ms to 5120 ms: 4
5131 ms to 5140 ms: 1

Foreground Time [illegible]
Background Time [illegible]
- Background Audio Time : 0 s
- Background Location [illegible]

Overall CPU Time: 78 s
Overall GPU Time: 0 s

Overall Location Usage (Ordered by power impact)
- Best Accuracy For Navigation Time : 0 s
- Best Accuracy Time : 714 s
- Nearest Ten Meters Accuracy Time : 0 s
- Hundred Meters Accuracy Time : 0 s
- Kilometers Accuracy Time : 0 s
- Three Kilometer Accuracy Time : 0 s

Overall CPU Time: 78 s

Overall GPU Time: 0 s

MetricKit Demo Takeaway

MetricKit Demo Takeaway

Ability to collect field battery + performance metrics yourself

MetricKit Demo Takeaway

Ability to collect field battery + performance metrics yourself

Use MetricKit to identify hotspots early from field users

MetricKit Demo Takeaway

Ability to collect field battery + performance metrics yourself

Use MetricKit to identify hotspots early from field users

Aggregate MetricKit data from multiple users

NEW

Introducing Xcode Metrics Organizer

Anshul Dawra, Software Engineer

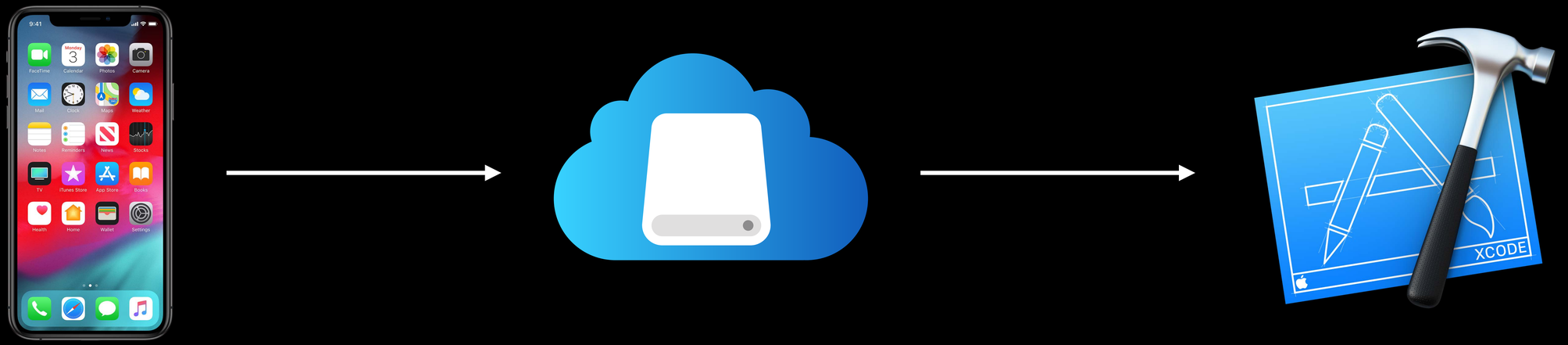
Xcode Metrics Organizer

Out-of-box battery and performance app analytics

No changes required to app

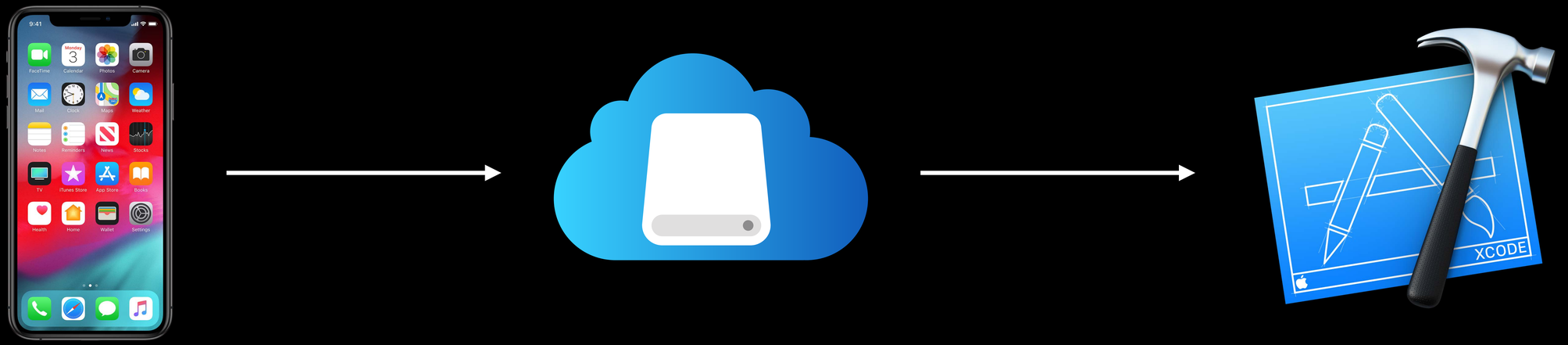
Data aggregation designed to protect user privacy

How It Works


9:41
FaceTime
Calendar
Photos
Camera
Mail
Clock
Maps
Weather
Notes
Reminders
News
Stocks
TV
iTunes Store
App Store
Books
Health
Home
Wallet
Settings
XCODE

How It Works

No app changes required to gather metrics



Demo

Xcode Metrics Organizer

Xcode Metrics Organizer Demo Takeaway

Out-of-box tool to view battery and performance analytics

Detect regressions across app versions

Available in Xcode 11

Tools overview

Metrics overview

Deep dives and Demos

Summary

Development
and Testing

Beta

Public Release

Development and Testing

Beta

Public Release

XCTest Metrics

MetricKit

Xcode Metrics Organizer

Development and Testing

Beta

Public Release

XCTest Metrics

MetricKit

Xcode Metrics Organizer

Summary

Summary

Three new tools this year

- XCTest Metrics
- MetricKit
- Xcode Metrics Organizer

Summary

Three new tools this year

- XCTest Metrics
- MetricKit
- Xcode Metrics Organizer

Tools can quantify the battery and performance impact of your app

Summary

Three new tools this year

- XCTest Metrics
- MetricKit
- Xcode Metrics Organizer

Tools can quantify the battery and performance impact of your app

Metrics can help you make better decisions about your app

More Information

developer.apple.com/wwdc19/417

Optimizing App Launch	Friday, 4:20
Performance, Power, Crashes, and Debugging Lab	Friday, 3:00


WWDC19