Writing Energy Efficient Apps

Session 238

Daniel Schucker, Software Power Engineer
Prajakta Karandikar, Software Power Engineer
Settings

Battery Usage

Last 24 Hours

App Usage

- EnergyBuddy: 28%
- Podcasts: 27%
- Photos: 17%
- Home & Lock Screen: 15%
- Settings: 5%
- Calculator: 2%
- Phone: 2%
- Tips: 1%
- Videos: 1%
- News: 1%
- Messages: 1%

Shows proportion of battery used by each app
Battery Life Concepts
Energy Efficient Coding
Energy Debugging Tools and Demo
Final Thoughts
General Battery Life Concepts
What Is Energy?
What Is Energy?

- Idle
- Active
- Overhead

Time

Power
What Is Energy?

Power vs. Time

- Idle
- Active
- Overhead

Energy
What Is Energy?

- **Power**
  - Idle
  - Active
  - Overhead

- **Time**
  - Dynamic cost
  - Fixed cost

- **Your work**
  - Overhead
Balancing Power and Battery Life
Balancing Power and Battery Life
What Consumes Energy?
What Consumes Energy?

- Processing
- Networking
- Location
- Graphics
How to Reduce Energy Consumption

Identify

Optimize

Coalesce

Reduce
Energy Efficient Coding
Social Networking App

Main feed
Post a photo
Analytics
Social Networking App
Main Feed

Current implementation
• Reloads on a timer
Social Networking App

Energy Impact

Feed Refresh

Power

Time
Social Networking App
Energy Impact

Feed Refresh

Power

Time
Social Networking App

Energy Impact

Power

Time

Feed Refresh
Radio cost

Dynamic cost
Overhead cost
Social Networking App
Main Feed

Reload only when needed
• User interaction
• Notification

Use NSURLSession Default Session
• New: WaitsForConnectivity
• Cache
Social Networking App
Energy Impact

Time

Power

Feed Refresh
Radio cost

User Interaction

Notification

Dynamic cost
Overhead cost
// Setup URLSession Default Session
let config = URLSessionConfiguration.default()

// Use WaitsForConnectivity
config.waitsForConnectivity = true

// URLSession Cache
let cachesDirectoryURL = FileManager.default().urlsForDirectory(.cachesDirectory, inDomains: .userDomainMask).first!
let cacheURL = try! cachesDirectoryURL.appendingPathComponent("MyCache")
var diskPath = cacheURL.path

let cache = URLCache(memoryCapacity: 16384, diskCapacity: 268435456, diskPath: diskPath)
config.urlCache = cache
config.requestCachePolicy = .useProtocolCachePolicy
// Setup NSURLSession Default Session
let config = URLSessionConfiguration.default()

// Use WaitsForConnectivity
cfg.waitsForConnectivity = true

// NSURLSession Cache
let cachesDirectoryURL = FileManager.default().urlsForDirectory(.cachesDirectory, inDomains: .userDomainMask).first!
let cacheURL = try! cachesDirectoryURL.appendingPathComponent("MyCache")
var diskPath = cacheURL.path

let cache = URLCache(memoryCapacity: 16384, diskCapacity: 268435456, diskPath: diskPath)
config.urlCache = cache
config.requestCachePolicy = .useProtocolCachePolicy
// Setup NSURLSession Default Session
let config = URLSessionConfiguration.default()

// Use WaitsForConnectivity
config.waitsForConnectivity = true

// NSURLSession Cache
let cachesDirectoryURL = FileManager.default().urlsForDirectory(.cachesDirectory, inDomains: .userDomainMask).first!
let cacheURL = try! cachesDirectoryURL.appendingPathComponent("MyCache")
var diskPath = cacheURL.path

let cache = URLCache(memoryCapacity: 16384, diskCapacity: 268435456, diskPath: diskPath)
config.urlCache = cache
config.requestCachePolicy = .useProtocolCachePolicy
// Setup NSURLSession Default Session
let config = URLSessionConfiguration.default()

// Use WaitsForConnectivity
config.waitsForConnectivity = true

// NSURLSession Cache
let cachesDirectoryURL = FileManager.default().urlsForDirectory(.cachesDirectory, inDomains: .userDomainMask).first!
let cacheURL = try! cachesDirectoryURL.appendingPathComponent("MyCache")
var diskPath = cacheURL.path

let cache = URLCache(memoryCapacity:16384, diskCapacity: 268435456, diskPath: diskPath)
config.urlCache = cache
config.requestCachePolicy = .useProtocolCachePolicy
Social Networking App
Posting a Photo

Current implementation
• Send immediately
• Retry on failure
Social Networking App

Energy Impact

- Send photo cost
- Radio cost

Time

Power

- Send Photo
- Timeout
- Retry
- Timeout

Dynamic cost

Overhead cost
Social Networking App
Posting a Photo

Use NSURLSession Default Session
• Minimize retries
• Set timeouts
• Batch Transactions

When retry limit hit
• Use Background Session
Social Networking App
Energy Impact

Time

Power

Send photo cost
Radio cost

Send Photo
Timeout

Retry
Timeout
Create Background
Session

Dynamic cost
Overhead cost
Social Networking App
Sending analytics data

Use NSURLSession Background Session
• Automatic retries
• Throughput monitoring

Properties
• New
  - Start time
  - Workload size
• Discretionary

Analytics
Social Networking App
Sending analytics data

Use NSURLSession Background Session
• Automatic retries
• Throughput monitoring

Properties
• New
  - Start time
  - Workload size
• Discretionary
Social Networking Application

Energy Impact

- Send analytics cost
- Other networking cost
- Radio cost

Time

Power

Dynamic cost

Overhead cost
// Setup NSURLSession Background Session
let config = URLSessionConfiguration.background(withIdentifier: "com.socialapp.background")
let session = URLSession(configuration: config, delegate: …, delegateQueue: …)

// Set discretionary property
config.discretionary = true

// Create Request and Task
var request = URLRequest(url: URL(string: "https://www.example.com/"))!
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 80
task.countOfBytesClientExpectsToReceive = 2048

task.resume()
let config = URLSessionConfiguration.background(withIdentifier: "com.socialapp.background")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)  

config.discretionary = true

var request = URLRequest(url: URL(string: "https://www.example.com/"))!
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

task.countOfBytesClientExpectsToSend = 80
task.countOfBytesClientExpectsToReceive = 2048

task.resume()
// Setup NSURLSession Background Session
let config = URLSessionConfiguration.background(withIdentifier: "com.socialapp.background")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

// Set discretionary property
config.discretionary = true

// Create Request and Task
var request = URLRequest(url: URL(string: "https://www.example.com/")!
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 80
task.countOfBytesClientExpectsToReceive = 2048

task.resume()
Social Networking Application
WatchOS

Use background session for screen off work
• Complication updates
• Background app refresh
• Runtime when task completes
Networking Best Practices

Identify
• Ensure transactions not repeated

Optimize
• Use background session

Coalesce
• Batch transactions

Reduce
• Minimize retries
Location Best Practices

Location APIs

Continuous location

Quick location update

Region monitoring

Visit monitoring

Significant location change
Location Best Practices

Continuous Location

Navigate to a destination
- Continuous location updates
- Prevents device sleep

Stop location updates
- Allows device to sleep
// Create location manager
locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()

// Set desired accuracy, auto-pause, and activity type appropriately
locationManager.desiredAccuracy = kCLLocationAccuracyThreeKilometers
locationManager.pausesLocationUpdatesAutomatically = true
locationManager.activityType = CLActivityTypeNavigation

// Set allows background if its needed
locationManager.allowsBackgroundLocationUpdates = true

// Start location updates
locationManager.startUpdatingLocation()
// Create location manager

```swift
locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()
```

// Set desired accuracy, auto-pause, and activity type appropriately
```swift
locationManager.desiredAccuracy = kCLLocationAccuracyThreeKilometers
locationManager.pausesLocationUpdatesAutomatically = true
locationManager.activityType = CLActivityTypeNavigation
```

// Set allows background if its needed
```swift
locationManager.allowsBackgroundLocationUpdates = true
```

// Start location updates
```swift
locationManager.startUpdatingLocation()
```
// Create location manager
locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()

// Set desired accuracy, auto-pause, and activity type appropriately
locationManager.desiredAccuracy = kCLLocationAccuracyThreeKilometers
locationManager.pausesLocationUpdatesAutomatically = true
locationManager.activityType = CLActivityTypeNavigation

// Set allows background if its needed
locationManager.allowsBackgroundLocationUpdates = true

// Start location updates
locationManager.startUpdatingLocation()
// Create location manager
locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()

// Set desired accuracy, auto-pause, and activity type appropriately
locationManager.desiredAccuracy = kCLLocationAccuracyThreeKilometers
locationManager.pausesLocationUpdatesAutomatically = true
locationManager.activityType = CLActivityTypeNavigation

// Set allows background if its needed
locationManager.allowsBackgroundLocationUpdates = true

// Start location updates
locationManager.startUpdatingLocation()
// Create location manager
locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()

// Set desired accuracy, auto-pause, and activity type appropriately
locationManager.desiredAccuracy = kCLLocationAccuracyThreeKilometers
locationManager.pausesLocationUpdatesAutomatically = true
locationManager.activityType = CLActivityTypeNavigation

// Set allows background if its needed
locationManager.allowsBackgroundLocationUpdates = true

// Start location updates
locationManager.startUpdatingLocation()
// Start location updates
locationManager.startUpdatingLocation()

// Get location updates
...

// Disable background updates when no longer needed
locationManager.allowsBackgroundLocationUpdates = false

// Stop location when no longer needed
locationManager.stopUpdatingLocation()
// Start location updates
locationManager.startUpdatingLocation()

// Get location updates
...

// Disable background updates when no longer needed
locationManager.allowsBackgroundLocationUpdates = false

// Stop location when no longer needed
locationManager.stopUpdatingLocation()
// Start location updates
locationManager.startUpdatingLocation()

// Get location updates
...

// Disable background updates when no longer needed
locationManager.allowsBackgroundLocationUpdates = false

// Stop location when no longer needed
locationManager.stopUpdatingLocation()
Location Best Practices

Request location

Get news based on current location

• Use quick location update

locationManager.requestLocation()
Location Best Practices

Request location

Get news based on current location

• Use quick location update

`locationManager.requestLocation()`
Location Best Practices

Region monitoring

Updating content when arriving at home

• Use region monitoring

// Create the geographic region to be monitored.
let geoRegion = CLCircularRegion(center: overlay.coordinate, radius: radius, identifier: identifier)

locationManager.startMonitoring(for: geoRegion)
Location Best Practices
Region monitoring

Updating content when arriving at home
• Use region monitoring

// Create the geographic region to be monitored.
let geoRegion = CLCircularRegion(center: overlay.coordinate, radius: radius, identifier: identifier)

locationManager.startMonitoring(for: geoRegion)
Location Best Practices

Visit monitoring

Updating content when arriving at frequently visited locations

• Use visit monitoring

// Start monitoring
locationManager.startMonitoringVisits();

// Stop monitoring when no longer needed
locationManager.stopMonitoringVisits();
Location Best Practices
Visit monitoring

Updating content when arriving at frequently visited locations
• Use visit monitoring

// Start monitoring
locationManager.startMonitoringVisits()

// Stop monitoring when no longer needed
locationManager.stopMonitoringVisits()
Location Best Practices
Visit monitoring

Updating content when arriving at frequently visited locations

• Use visit monitoring

// Start monitoring
locationManager.startMonitoringVisits()

// Stop monitoring when no longer needed
locationManager.stopMonitoringVisits()
Location Best Practices

Significant location change

Updating content based user location

• Use significant location change

```java
// Start monitoring
locationManager.startMonitoringSignificantLocationChanges()

// Stop monitoring when no longer needed
locationManager.stopMonitoringSignificantLocationChanges()
```
Location Best Practices
Significant location change

Updating content based user location

• Use significant location change

```java
// Start monitoring
locationManager.startMonitoringSignificantLocationChanges();

// Stop monitoring when no longer needed
locationManager.stopMonitoringSignificantLocationChanges();
```
Location Best Practices

Significant location change

Updating content based user location

- Use significant location change

```java
// Start monitoring
locationManager.startMonitoringSignificantLocationChanges()

// Stop monitoring when no longer needed
locationManager.stopMonitoringSignificantLocationChanges()
```
Location Best Practices

Identify:
• Accuracy level needed

Optimize:
• Use alternatives to continuous location

Reduce:
• Stop location when not used

Coalesce:
• Defer location updates
Graphics

Minimize screen updates
• Ensure screen updates provide needed changes

Review blur usage
• Avoid placing blur over updating elements
Minimize use of Discrete GPU

Use Discrete GPU only when:

• Animation performance suffers
• Functionality isn't supported
MTLCreateSystemDefaultDevice()

- Always uses Discrete GPU

Use Integrated GPU when possible

- MTLCopyAllDevices
  - Select device with isLowPower attribute set
Graphics
MacOS—OpenGL

Make your app mux-aware by either:

• Adding NSSupportsAutomaticGraphicsSwitching to your info.plist

![NSSupportsAutomaticGraphicsSwitching](image)

• Creating an OpenGL context with the automatic graphics switching attribute

```swift
let attributes3 : [CGLPixelFormatAttribute] = [
    kCGLPFASupportsAutomaticGraphicsSwitching,
    kCGLPFAAllowOfflineRenderers,
    CGLPixelFormatAttribute(0)
]

CGLChoosePixelFormat(attributes3, &pix, &npix)

CGLCreateContext(pPixelFmt, nil, &pContext)
```
Graphics Best Practices

Identify:
• Blur usage

Optimize:
• Only use discrete GPU when needed (macOS)

Reduce:
• Minimize screen updates
Processing Best Practices

- Identify tasks
- Do work quickly and efficiently
- Avoid timers
- Set leeway
Background Processing

Finish work quickly

Use background app refresh

Call completion handler
Background Processing
iOS

PushKit API

• Now has completion handler
• Call after handling push

```swift
func pushRegistry(_ registry: PKPushRegistry, didReceiveIncomingPushWith payload: PKPushPayload, forType type: PKPushType, withCompletionHandler completion: @escaping () -> Void) {
    // Process the incoming push payload here...

    // Then signal that processing has completed
    completion()
}
```
Background Processing

iOS

PushKit API

• Now has completion handler
• Call after handling push

```swift
func pushRegistry(_ registry: PKPushRegistry, didReceiveIncomingPushWith payload: PKPushPayload, forType type: PKPushType, withCompletionHandler completion: @escaping () -> Void) {
    // Process the incoming push payload here...

    // Then signal that processing has completed
    completion()
}
```
Background Processing
WatchOS

New: Navigation background mode
  • CPU limits like Workout
  • Minimize networking
  • Ensure work is relevant to the user

Use background app refresh and complication updates to refresh data
Background Processing

Identify:
• Work done in the background

Optimize:
• Use Background App Refresh

Reduce:
• Limit transactions

Coalesce:
• Use NSURLSession background session
Battery Life Concepts
Energy Efficient Coding
Energy Debugging Tools and Demo
Final Thoughts
Overview

Energy debugging tools

Measuring energy impact of your apps

Demo
Settings: Battery
Settings: Battery

Can be deleted by users
Energy Debugging Workflow

Energy Gauges
Energy

Average Energy Impact

High Energy Impact

Average Component Utilization

- Overhead 54.4%
- CPU 4%
- Network 4%
- Location 15.8%
- GPU 0%

Energy Impact

00:00:000 | 00:00:059 | 00:00:10 | 00:00:15 | 00:00:20 | 00:00:25 | 00:00:30 | 00:00:35 | 00:00:40

Foreground | Background | Background | Background

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent leveling uses more energy. Request location and increase precision only when truly necessary.

Energy Report
Running EnergyBuddy on iPhone
Energy

Average Energy Impact

High
Energy Impact

Average Component Utilization

- Overhead: 54.4%
- CPU: 4%
- Network: 6%
- Location: 15.8%
- GPU: 0%

Energy Impact

00:00:000 | 00:05:000 | 00:10:000 | 00:15:000 | 00:20:000 | 00:25:000 | 00:30:000 | 00:35:000

Foreground
Background
Foreground

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locationing uses more energy. Request location and increase precision only when truly
Energy

Average Energy Impact

High
Energy Impact

Average Component Utilization

- Overhead 54.4%
- CPU 4%
- Network 4%
- Location 16.8%
- GPU 0%

Energy Impact

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.
Energy Impact

Average Energy Impact

High Energy Impact

Average Component Utilization

Overhead: 54.4%
CPU: 4%
Network: 4%
Location: 5.5%
GPU: 0%

Energy Impact

00:00:000 00:00:005 00:00:010 00:00:015 00:00:020 00:00:025 00:00:030 00:00:035 00:00:040

Overhead
CPU
Network
Location
GPU
Background
Foreground
Suspended

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 20%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly needed.
Energy

Average Energy Impact

High
Energy Impact

Average Component Utilization

Overhead 54.4%
CPU 4%
Location 10.5%
Network 0%
GPU 0%

Energy Impact

00:00:000 00:00:009 00:00:100 00:00:110 00:00:120 00:00:130 00:00:140 00:00:150 00:00:160 00:00:170 00:00:180 00:00:190 00:00:200 00:00:210 00:00:220 00:00:230 00:00:240 00:00:250 00:00:260 00:00:270 00:00:280 00:00:290 00:00:300 00:00:310 00:00:320 00:00:330 00:00:340

Foreground
Background Foreground

Overhead
CPU Location GPU Background Foreground Suspended

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locationing uses more energy. Request location and increase precision only when truly
Energy

Average Energy Impact

High Energy Impact

Average Component Utilization

Overhead 54.4%
CPU 4%
Network 6%
Location 10.8%
GPU 0%

Energy Impact

00:00:000 00:00:005 00:00:010 00:00:015 00:00:020 00:00:025 00:00:030 00:00:035

Foreground
Background

Overhead represents energy usage as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent location uses more energy. Request location and increase precision only when truly needed.
Average Energy Impact

High Energy Impact

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 25%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.
Average Energy Impact

High Energy Impact

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 70%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.
Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.
Average Energy Impact

High Energy Impact

Overhead
Overhead represents energy use as a result of bringing-up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly needed.
Average Energy Impact

High Energy Impact

Overhead
Overhead represents energy use as a result of bringing-up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device’s battery. Always use the CPU efficiently and return to idles as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly needed.
Average Energy Impact

High Energy Impact

Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.
Energy Report

Average Energy Impact

High Energy Impact

Average Component Utilization

- Overhead: 54.4%
- CPU: 4%
- Network: 4%
- GPU: 0%

Energy Impact

00:00:000 | 00:00:599 | 00:01:000 | 00:01:599 | 00:02:000 | 00:02:599 | 00:03:000 | 00:03:599 | 00:04:000

Foreground

Background

Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization

CPU usage in excess of 10% is cause for concern. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity

Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity

Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.
Energy

Average Energy Impact

High
Energy Impact

Average Component Utilization

Overhead 54.4%
CPU 4%
Network 4%

Energy Impact

00:10:000 00:15:000 00:20:000 00:25:000 00:30:000 00:35:000

Foreground
Background

Overhead
CPU
Network
Location
GPU

Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization

CPU usage of greater than 10%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity

Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity

Location activity performed by your app. More precise and frequent location uses more energy. Request location and increase precision only when truly necessary.
Overhead

Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization

CPU usage of greater than 20%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity

Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity

Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.

High GPU Utilization

Graphics activity requested by your app. Extraneous graphics and animations reduce responsiveness and pull system resources out of low-power states or prevent them from powering down all together, resulting in significant energy use. Make updates to visible content only, reduce the use of opacity, and prefer lower, consistent frame rates when performing animations.

Background State

Your app is in a background state, keeping the system awake. Even an idle background app uses energy. If your app requires background operations, use deferral APIs that let the system schedule the work efficiently and wake to run your app only when necessary. Otherwise, reduce activity immediately when placed in the background and notify the system once the activity is complete.

Foreground State

Your app is in the foreground. Use recommended APIs, batch and reduce network operations, and avoid unnecessary updates to the user interface. Strive to make your app absolutely idle when it's not responding to user input.

Suspended State

Your app was suspended by the system. Overhead produced by your app and out-of-process activities your app initiated, like location updates, may still consume energy.
Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 20%. High CPU utilization rapidly drains a device’s battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.

High GPU Utilization
Graphics activity requested by your app. Extraneous graphics and animations reduce responsiveness and pull system resources out of low-power states or prevent them from powering down all together, resulting in significant energy use. Make updates to visible content only, reduce the use of opacity, and prefer lower, consistent frame rates when performing animations.

Background State
Your app is in a background state, keeping the system awake. Even an idle background app uses energy. If your app requires background operations, use deferral APIs that let the system schedule the work efficiently and wake to run your app only when necessary. Otherwise, reduce activity immediately when placed in the background and notify the system once the activity is complete.

Foreground State
Your app is in the foreground. Use recommended APIs, batch and reduce network operations, and avoid unnecessary updates to the user interface. Strive to make your app absolutely idle when it’s not responding to user input.

Suspended State
Your app was suspended by the system. Overhead produced by your app and out-of-process activities your app initiated, like location updates, may still consume energy.
Overhead

Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization

CPU usage of greater than 20%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity

Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity

Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.

High GPU Utilization

Graphics activity requested by your app. Extraneous graphics and animations reduce responsiveness and pull system resources out of low-power states or prevent them from powering down altogether, resulting in significant energy use. Make updates to visible content only, reduce the use of opacity, and prefer lower, consistent frame rates when performing animations.

Background State

Your app is in a background state, keeping the system awake. Even an idle background app uses energy. If your app requires background operations, use deferral APIs that let the system schedule the work efficiently and wake to run your app only when necessary. Otherwise, reduce activity immediately when placed in the background and notify the system once the activity is complete.

Foreground State

Your app is in the foreground. Use recommended APIs, batch and reduce network operations, and avoid unnecessary updates to the user interface. Strive to make your app absolutely idle when it’s not responding to user input.

Suspended State

Your app was suspended by the system. Overhead produced by your app and out-of-process activities your app initiated, like location updates, may still consume energy.
Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 20%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.

High GPU Utilization
Graphics activity requested by your app. Extraneous graphics and animations reduce responsiveness and pull system resources out of low-power states or prevent them from powering down altogether, resulting in significant energy use. Make updates to visible content only, reduce the use of opacity, and prefer lower, consistent frame rates when performing animations.

Background State
Your app is in a background state, keeping the system awake. Even an idle background app uses energy. If your app requires background operations, use deferral APIs that let the system schedule the work efficiently and wake to run your app only when necessary. Otherwise, reduce activity immediately when placed in the background and notify the system once the activity is complete.

Foreground State
Your app is in the foreground. Use recommended APIs, batch and reduce network operations, and avoid unnecessary updates to the user interface. Strive to make your app absolutely idle when it's not responding to user input.

Suspended State
Your app was suspended by the system. Overhead produced by your app and out-of-process activities your app initiated, like location updates, may still consume energy.
Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 20%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.

High GPU Utilization
Graphics activity requested by your app. Extraneous graphics and animations reduce responsiveness and pull system resources out of low-power states or prevent them from powering down all together, resulting in significant energy use. Make updates to visible content only, reduce the use of opacity, and prefer lower, consistent frame rates when performing animations.

Background State
Your app is in a background state, keeping the system awake. Even an idle background app uses energy. If your app requires background operations, use deferral APIs that let the system schedule the work efficiently and wake to run your app only when necessary. Otherwise, reduce activity immediately when placed in the background and notify the system once the activity is complete.

Foreground State
Your app is in the foreground. Use recommended APIs, batch and reduce network operations, and avoid unnecessary updates to the user interface. Strive to make your app absolutely idle when it’s not responding to user input.

Suspended State
Your app was suspended by the system. Overhead produced by your app and out-of-process activities your app initiated, like location updates, may still consume energy.
Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 20%. High CPU utilization rapidly drains a device's battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.

High GPU Utilization
Graphics activity requested by your app. Extraneous graphics and animations reduce responsiveness and pull system resources out of low-power states or prevent them from powering down all together, resulting in significant energy use. Make updates to visible content only, reduce the use of opacity, and prefer lower, consistent frame rates when performing animations.

Background State
Your app is in a background state, keeping the system awake. Even an idle background app uses energy. If your app requires background operations, use deferral APIs that let the system schedule the work efficiently and wake to run your app only when necessary. Otherwise, reduce activity immediately when placed in the background and notify the system once the activity is complete.

Foreground State
Your app is in the foreground. Use recommended APIs, batch and reduce network operations, and avoid unnecessary updates to the user interface. Strive to make your app absolutely idle when it's not responding to user input.

Suspended State
Your app was suspended by the system. Overhead produced by your app and out-of-process activities your app initiated, like location updates, may still consume energy.
Overhead
Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

High CPU Utilization
CPU usage of greater than 20%. High CPU utilization rapidly drains a device’s battery. Always use the CPU efficiently and return to idle as quickly as possible when not directly responding to user input.

Network Activity
Network activity occurring in response to your app. Networking brings up radios, which require power for prolonged periods. Batch network activity whenever possible to reduce overhead.

Location Activity
Location activity performed by your app. More precise and frequent locating uses more energy. Request location and increase precision only when truly necessary.

High GPU Utilization
Graphics activity requested by your app. Extraneous graphics and animations reduce responsiveness and pull system resources out of low-power states or prevent them from powering down all together, resulting in significant energy use. Make updates to visible content only, reduce the use of opacity, and prefer lower, consistent frame rates when performing animations.

Background State
Your app is in a background state, keeping the system awake. Even an idle background app uses energy. If your app requires background operations, use deferral APIs that let the system schedule the work efficiently and wake to run your app only when necessary. Otherwise, reduce activity immediately when placed in the background and notify the system once the activity is complete.

Foreground State
Your app is in the foreground. Use recommended APIs, batch and reduce network operations, and avoid unnecessary updates to the user interface. Strive to make your app absolutely idle when it’s not responding to user input.

Suspended State
Your app was suspended by the system. Overhead produced by your app and out-of-process activities your app initiated, like location updates, may still consume energy.
Scenarios for Energy Debugging

General/common
• Launch and Idle
• Background

Application specific
• For example, Navigation App
  - Search for an address
  - Get directions
  - Navigate
Demo
Battery Life Concepts
Energy Efficient Coding
Energy Debugging Tools and Demo
Final Thoughts
Final Thoughts

Use NSURLSession Background Session

Minimize use of continuous location

Avoid timers

Coalesce work

Use energy gauges
<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advances in Networking, Part 1</td>
<td>Executive Ballroom</td>
<td>Wednesday 3:00PM</td>
</tr>
<tr>
<td>Advances in Networking, Part 2</td>
<td>Executive Ballroom</td>
<td>Wednesday 4:00PM</td>
</tr>
<tr>
<td>NSURLSession: New Features and Best Practices</td>
<td></td>
<td>WWDC 2016</td>
</tr>
<tr>
<td>Networking for the Modern Internet</td>
<td></td>
<td>WWDC 2016</td>
</tr>
<tr>
<td>Labs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Power and Performance Lab</strong></td>
<td><strong>Technology Lab I</strong></td>
<td>Fri 11:00AM–1:00PM</td>
</tr>
</tbody>
</table>