

Optimizing iOS Applications

Marc-Antoine Scheurer
marco@sente.ch

Optimizing is...

- Risky: “If it ain’t broke, don’t fix it”
- A hardware problem: “Buy a faster computer”
- Time consuming: “I don’t have time for performance work”
- Complicated: “I don’t know where to start or what to do”

Why?

- For the user: better experience
- For you: beat the competition
- For the OS: watch out the watchdog

A Hardware Problem: Limited Resources

	3G	3GS	4	iPad
CPU clock L1 L2	ARM 11 412 MHz 32 KB -	ARM Cortex A8 600 MHz 64 KB 256 KB	A4 (Cortex A8) 1 GHz 64 KB 640 KB	A4 (Cortex A8) 1 GHz 64 KB 640 KB
GPU triangles / s	PowerVR MBX Lite 0.6 M	PowerVR SGX 5.2 M	A4 (PowerVR SGX) 28 M ??	A4 (PowerVR SGX) 28 M ??
RAM available	128 MB 40 MB	256 MB 150 MB	512 MB 400 MB ?	256 MB 150 MB
Flash available	8 - 32 GB 2 GB	16 - 32 GB 2 GB	16 - 32 GB 2 GB	16 - 32 - 64 GB 2 GB
Screen	480x320	480x320	960x640	1024x728
More	GPS	Compass	Gyroscope	
Battery	6h on WiFi	9h on WiFi	10h on WiFi	10-12h video

Where to start?

Measuring Performance

- Guessing often does not work
- Measure - Change - Measure
- Use measuring tools

Measuring Tools

- NSLog
- Instruments
- Simulator

NSLog

```
NSDate *startTime = [NSDate date];  
...  
NSLog(@"Elapsed time: %.3f", -[startTime timeIntervalSinceNow]);
```

```
double start = CFAbsoluteTimeGetCurrent();  
...  
double elapsedTime = CFAbsoluteTimeGetCurrent() - start;  
NSLog(@"Elapsed time: %.3f", elapsedTime);
```



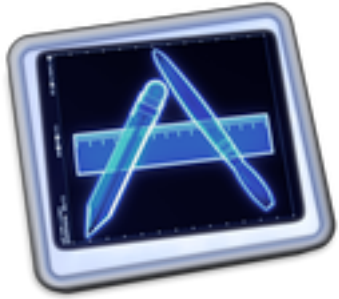
Instruments

Tool	Introduced
Objective C	1988
Interface Builder	1988*
Xcode	1988†
Instruments	2007‡

* SOS Interface, 1984-1986 (LISP, Mac)

† Project Builder, renamed Xcode in 2003

‡ Shikari, Shark in 2002



Instruments

- Profiling tool
 - CPU
 - Memory
 - CoreAnimation
 - CoreData
 - OpenGL
 - Power
 - etc.





Simulator

- Software simulator \neq hardware emulator

Simulator	3G
i386	ARM
2.5 GHz	400 MHz
Unlimited memory	40 MB

- Measure and test on oldest supported device (3G)
- Simulator OK to check memory usage

Better user experience

1. Responsive application
2. No crash (or kill by the watchdog)
3. Good battery life

I. Responsive applications





Time

- Faster code
 - makes applications more responsive
 - uses less power



Time

- Animation
- Objective C
- C and ARM code generation
- Frameworks
- File system



Animation

- Demo



Animation

- Scrolling at 60 fps
 - Reuse identifier
 - Reuse formatters
 - Opaque views
 - UINib (4.x)



Objective C

```
[array indexesOfObjectsWithOptions:NSMakeRangeerationConcurrent  
    passingTest:^(id obj, NSUInteger idx, BOOL *stop) {  
        return [obj test:value];  
    }  
]
```

- Blocks
- IMP caching



Objective C

- Demo



C and ARM Code Generation

- Demo



C and ARM Code Generation

- ARM v6 (2G, 3G)
- ARM v7 (3GS, 4)
- -Os vs -O3
- Floating points: remove Thumb on v6
- Contiguous memory, shadow variables, etc.



Frameworks

- Foundation
- Accelerate
- Grand Central Dispatch



Foundation

- NSMutableArray, NSMutableString
 - insert $O(N)$, $O(1)$ at begin and end
- NSMutableDictionary, NSMutableSet
 - lookup $O(1)$ with good hash function
- NSMutableIndexSet
 - Store integers without NSNumber



Accelerate

- ARM v6
 - Integer unit with 16 registers
 - Hardware floating-point (VFP)
 - 32 single precision registers
 - 16 double precision registers
- ARM v7
 - Integer unit with 16 registers
 - Legacy VFP support
 - NEON: 16 128-bit vector registers
 - Single precision floating-point uses NEON
 - NEON can decode MP3 on 10MHz CPU



Accelerate

- Makes using VFP and NEON easy
- Demo



Accelerate

- 2000 APIs for hardware accelerated math
- vDSP (Digital Signal Processing)
 - Fourier transforms
- BLAS (Basic Linear Algebra Subprograms)
 - Matrix product
- LAPACK (Linear algebra)
 - Solving system of linear equations
- Later? vImage, vForce, ...



Grand Central Dispatch

- Concurrent programming
- Multicore or single core
- Easier and lighter than threads
- Use to move long processes off the main thread
- `dispatch_async()`



Grand Central Dispatch

- Demo



File system

- Access to Flash memory is relatively slow
- Speed vary a lot from device to device
- For large files use memory mapped files
`+ [NSData dataWithContentsOfMappedFile:]`
- Long I/O off the main thread



File System

- I/O of small amount of data:
 - UserDefaults
 - Property Lists
 - NSArchiver
- I/O of large amount of data:
 - Incremental formats
 - Databases (CoreData)



Property Lists

- Binary Format 2-3x faster than XML
- Use NSPropertyListSerialization
- Slow:

- [NSArray writeToFile:atomically:]
 - [NSDictionary writeToFile:atomically:]
 - [NSString writeToFile:atomically:encoding:error:]

- Fast:

```
NSData *data = [NSPropertyListSerialization dataWithPropertyList:dictionary
                                                    format:NSPropertyListBinaryFormat_v1_0
                                                    options:0
                                                    error:NULL];
[data writeToFile:path atomically:YES];
```



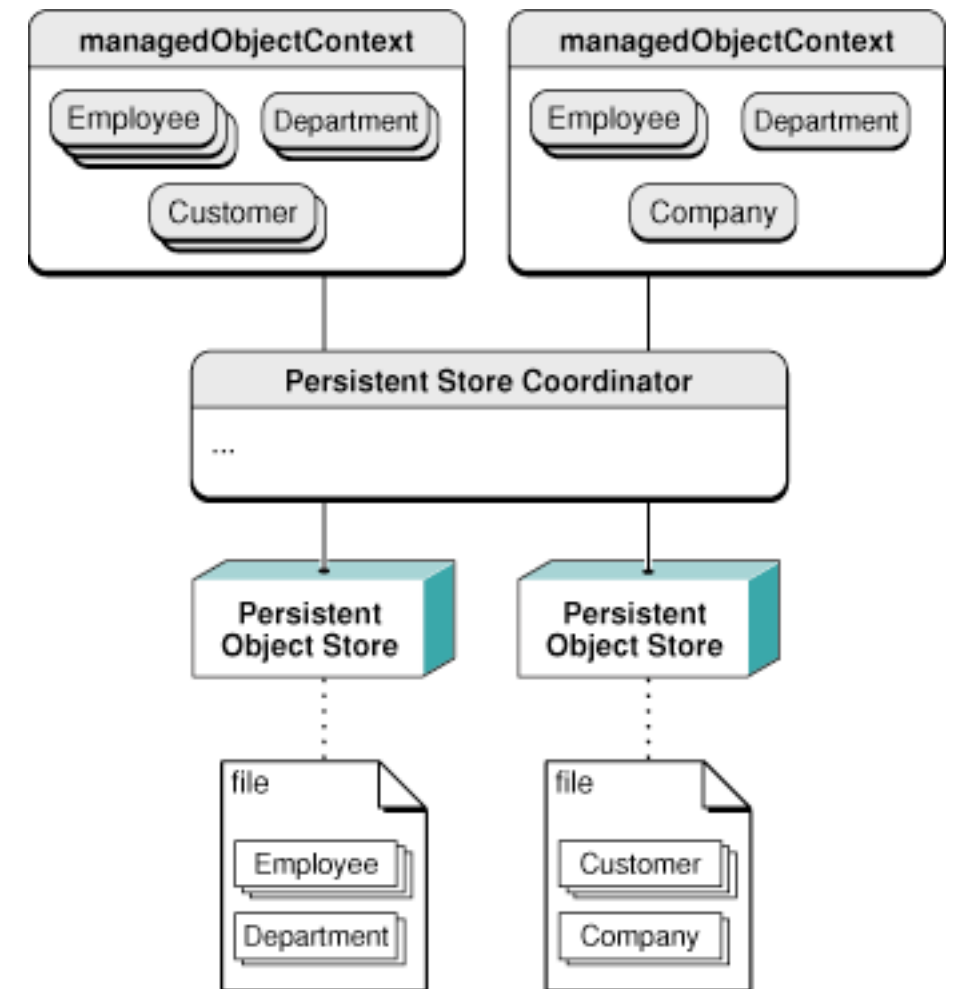
NSArchiver

- Not an incremental file format



Database

- Use CoreData
- Do not import large quantity of data...
- Add object stores instead





Database Search

- Unicode string comparisons are expensive
- Use normalized derived attribute (without accents)
- Use `<=` and `<` instead of `BEGINSWITH`
- Prefer prefix searching



Database Search

- Point de Vue app: 25'000 records

3G

	Before	After
	<pre>[[each name] rangeOfString:searchString options:(NSAnchored NSCaseInsensitive NSDiacriticInsensitiveSearch)].location != NSNotFound</pre>	<pre>[[each asciiName] hasPrefix:searchString]</pre>
“l”	0.74	0.45
“la”	0.03	0.02
“lau”	0.01	0.01
“laus”	0.00	0.00
“lau”	0.28	0.15

2. No Crash





Activity Watchdog

Maximum Allowed Time	
Launch	20 s
Resume	10 s
Suspend	10 s
Quit	6 s
Operation	10 min



Memory Watchdog “Jetsam”

- Watches memory pressure
- Issues low memory warning
- Instant termination of application



Memory

- No swap
 - (but yes: virtual memory)
- Memory usage also impacts time
 - allocation time,
 - unloading of executable code, ...



Memory Usage

- Avoid
 - spikes
 - leaks
 - abandoned memory
 - zombies



Memory Spikes



- Be smart with autorelease
- Process large quantities of data in batches
- Nested autorelease pools



Memory Spikes

- Demo



Memory Leaks



- Allocated memory that is inaccessible
- Unbalanced retain/release
- Forget to release property's original value
- Leaks Instrument examines heap for leaked memory, identify allocation
- Xcode: Build and Analyze



Abandoned Memory



- Wasted memory
- Accessible but never used
- Memory should not grow without bounds
- Detect with Allocation Instrument: “heap shots”



Zombies



- Messages send to deallocated object
- Detect with Zombies template
 - or Allocations “Enable zombie detection”
- Simulator only



Zombies



- Demo

3. Battery Life





Energy (iPhone 3G)

- 3G, WiFi, Bluetooth, GPS: 2 W
- CPU + GPU: 800 mW
- Screen: 200 mW



Radios and sensors

- Network
- CoreLocation
- CoreMotion



Network

- Bandwidth usage impacts time and power
- Use compact protocols
- (CSV vs JSON vs XML)



3G

- 3G networks require phones stay in high-power state for a few seconds after last packet is sent or received
- Limit number of connections
- Do not poll
- Use compact data formats

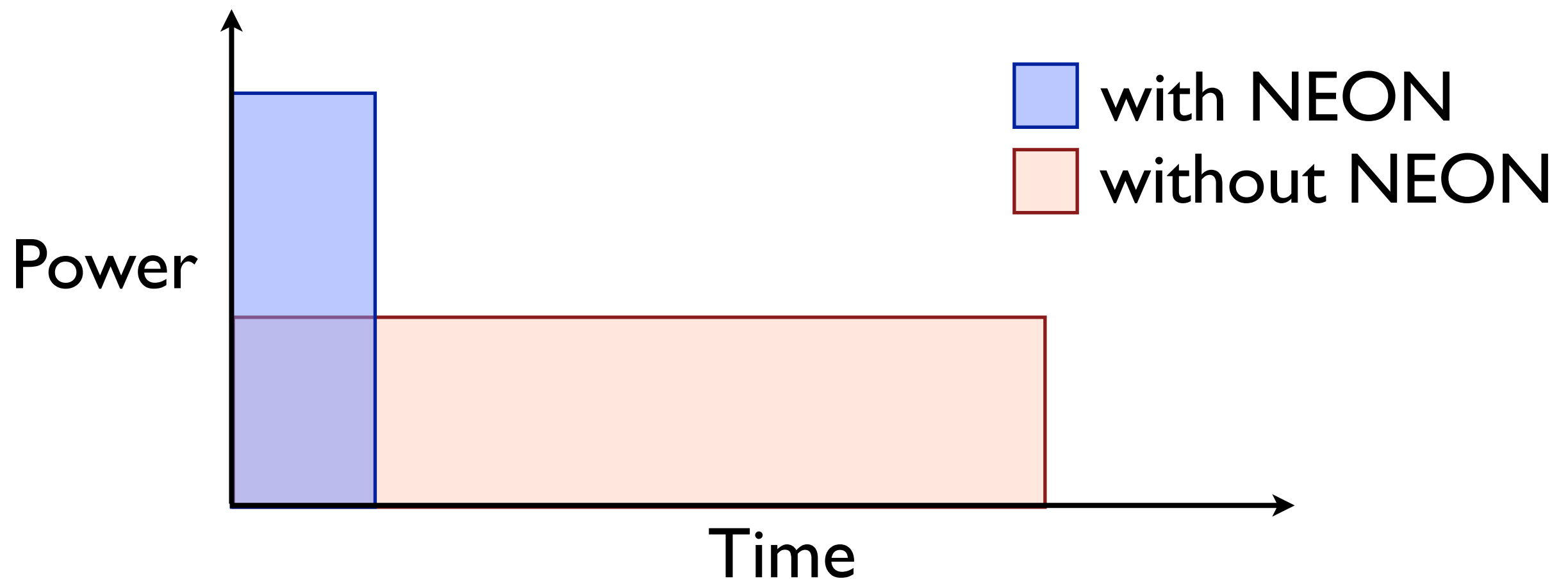


WiFi & 2G

- Less power than 3G
 - $3G > 2G > \text{WiFi}$
- Allowed to idle immediately
- 2G much slower

CPU

- Do not poll, use events
- Accelerate: more power, less energy





CoreLocation

- Use minimum required accuracy

GPS	kCLLocationAccuracyBest
GPS	kCLLocationAccuracyNearestTenMeters
WiFi	kCLLocationAccuracyHundredMeters
Cell / WiFi	kCLLocationAccuracyKilometer



CoreLocation

- `distanceFilter`
 - how often you receive location changed notifications
 - default: all changes, many events, high CPU usage
- `stopUpdatingLocation`
 - when good enough accuracy, switch GPS off.

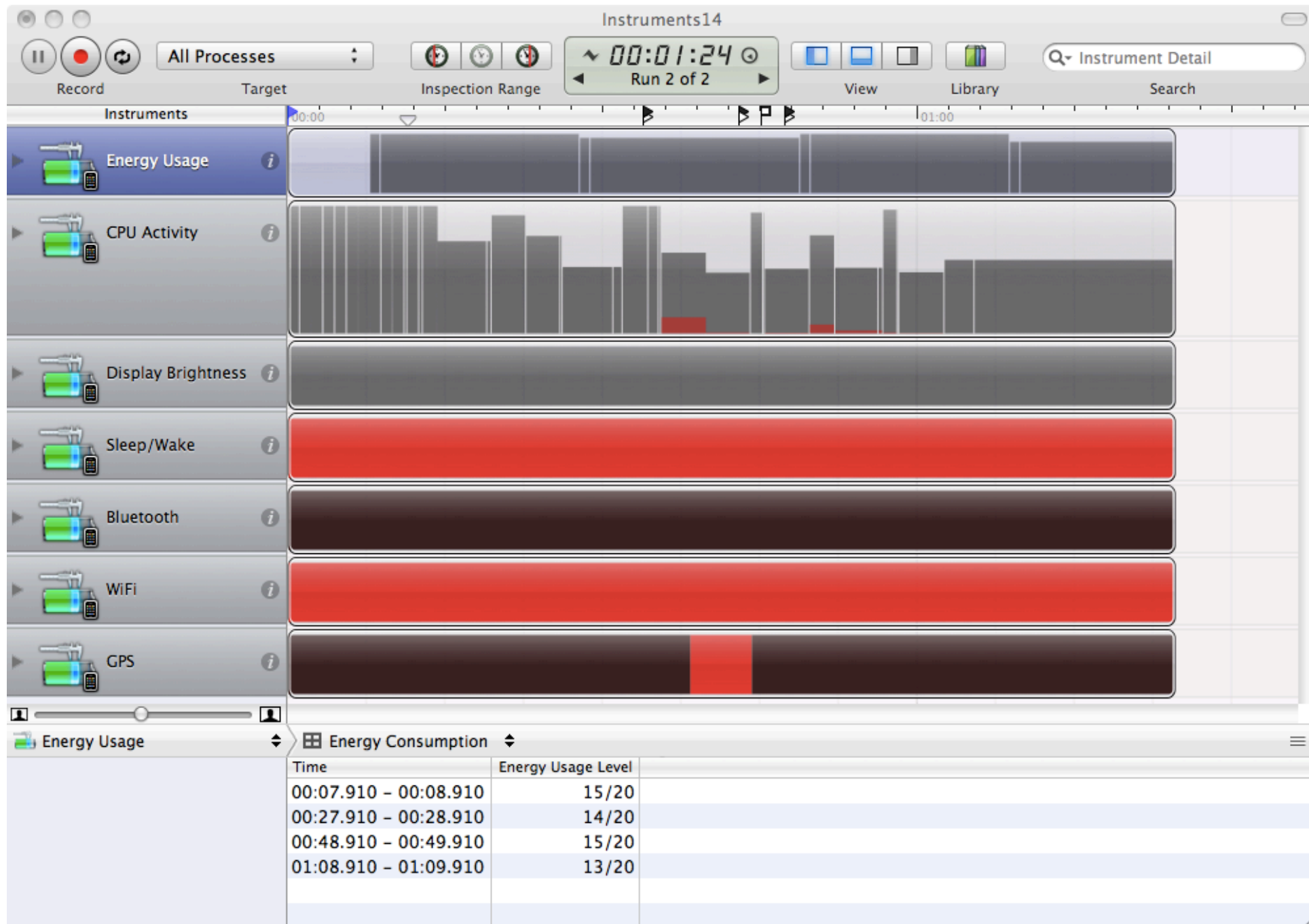


CoreMotion

- Same thing
- Turn sensors off when in background (4.x)



Energy diagnostics





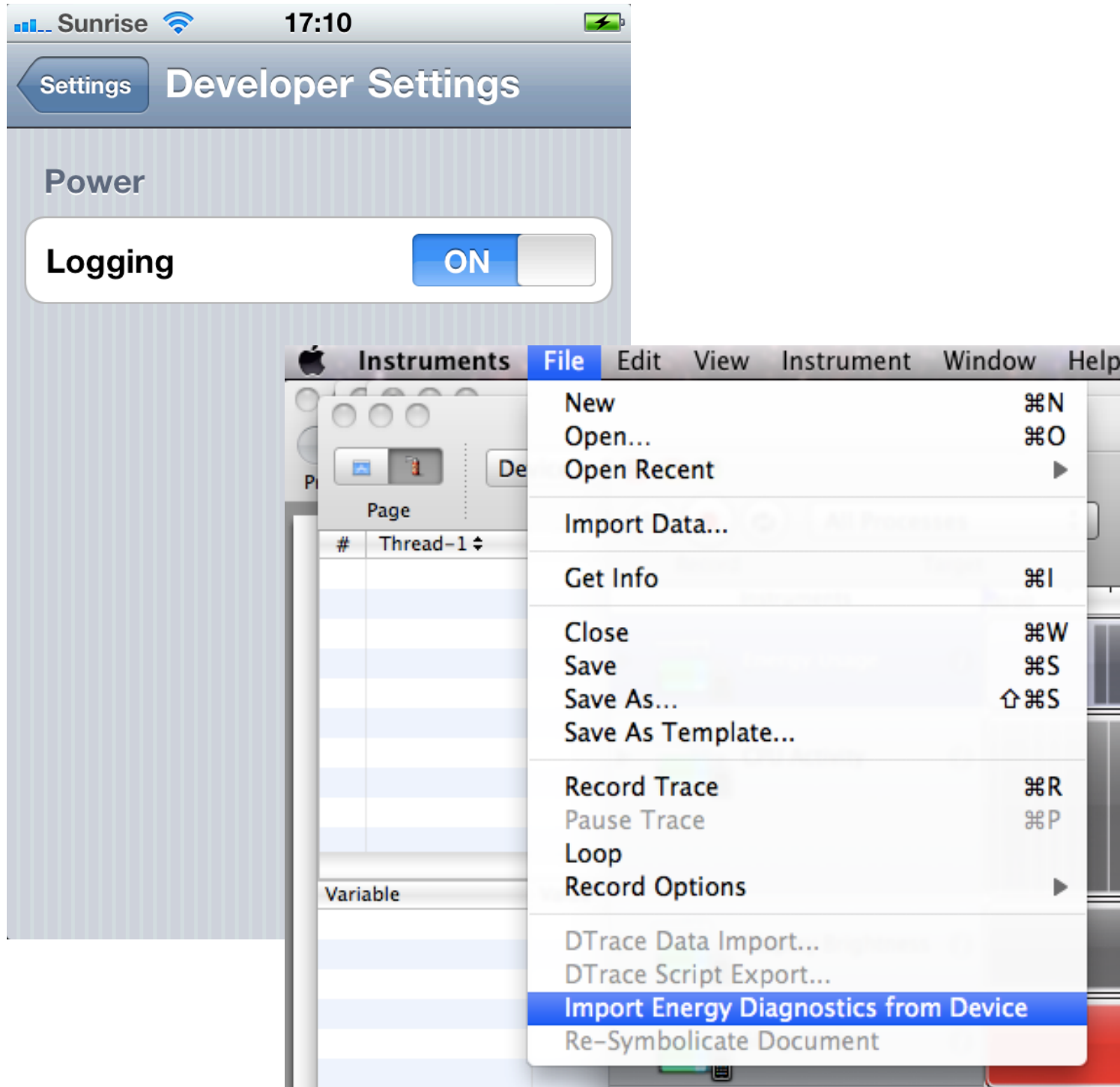
Energy usage

- Works on
 - 3GS
 - 4
 - iPod 2 & 3

Level	Battery life
20	Less than 1 hour
10	About 10 hours
1	More than 20 hours



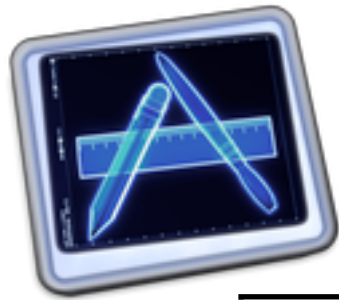
Measure on the go




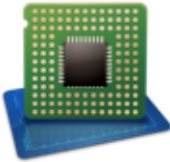









- Enable device to collect data
- Import in Instruments

Summary





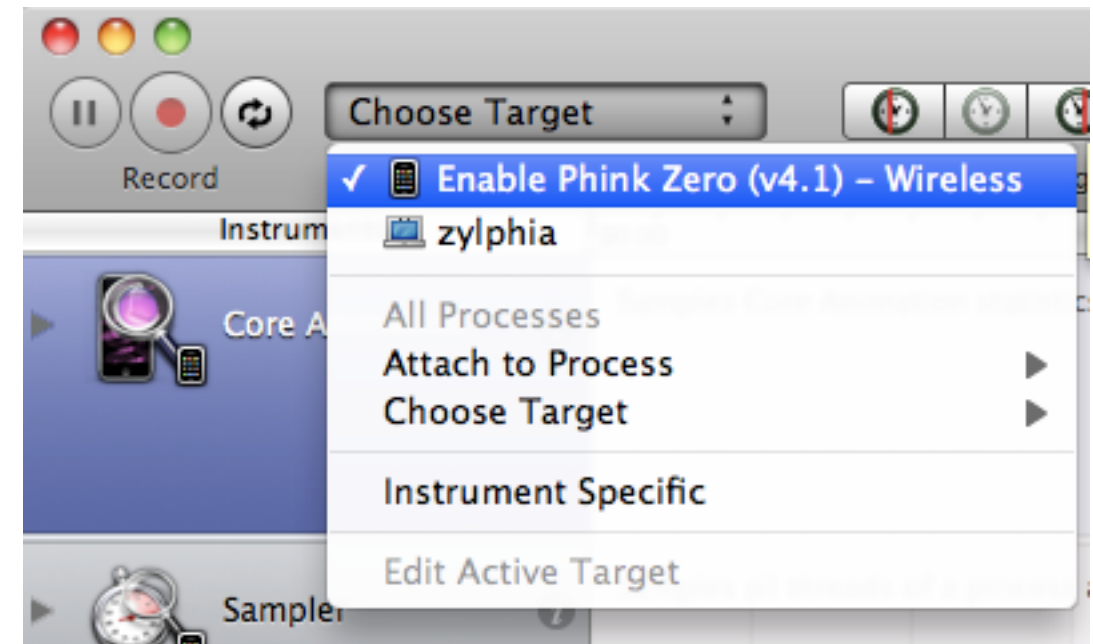
Instruments' instruments

CPU	  Time ProfilerCPU Sampler
Memory	   AllocationsLeaksActivity Monitor
File System	  Activity MonitorSystem Usage
Core Data	 CoreData
Graphics	  Core AnimationOpenGL ES
Energy	 Energy Diagnostics



Instruments Wireless

- Connect device with USB
- Hold alt while choosing target
- Choose wireless device
- Disconnect from USB



When?

- All the time

Useful links

- Xcode Developer documentation
- Apple

<http://developer.apple.com/iphone>

<http://developer.apple.com/videos/wwdc/2010/>

- Developer forums

<http://www.iphonedevsdk.com/>

<http://stackoverflow.com>

- Developer classes

Stanford iPhone Application Development (english, free) sur **iTunesU**

<http://deimos3.apple.com/WebObjects/Core.woa/Browse/itunes.stanford.edu.3124430053.03124430055>

Sen:te (french, paying)

<http://www.iphone-class.com>