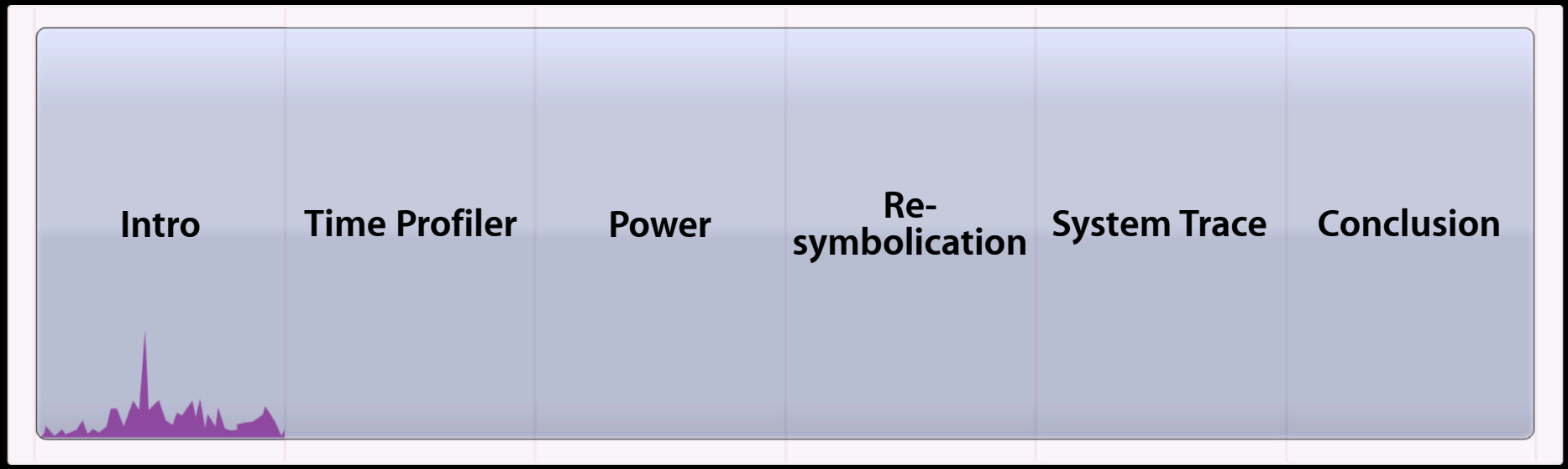
Advanced Performance Analysis with Instruments

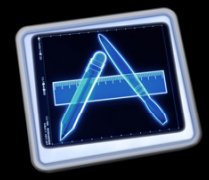
Lynne Salameh

Today's Agenda

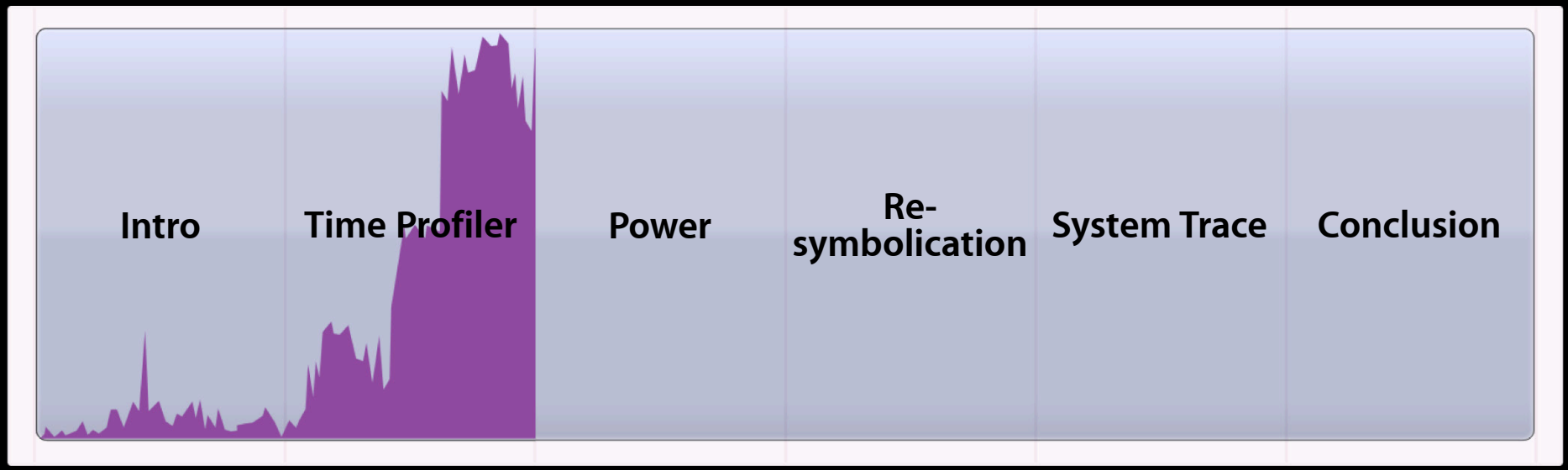


Introduction

Advanced performance analysis



- Leverage Time Profiling and Data Mining
- Understand power management for the iPhone
- Adopt a holistic view of the system


Intro
Time Profiler
Power
Re-symbolication
System Trace
Conclusion

Time Profiler

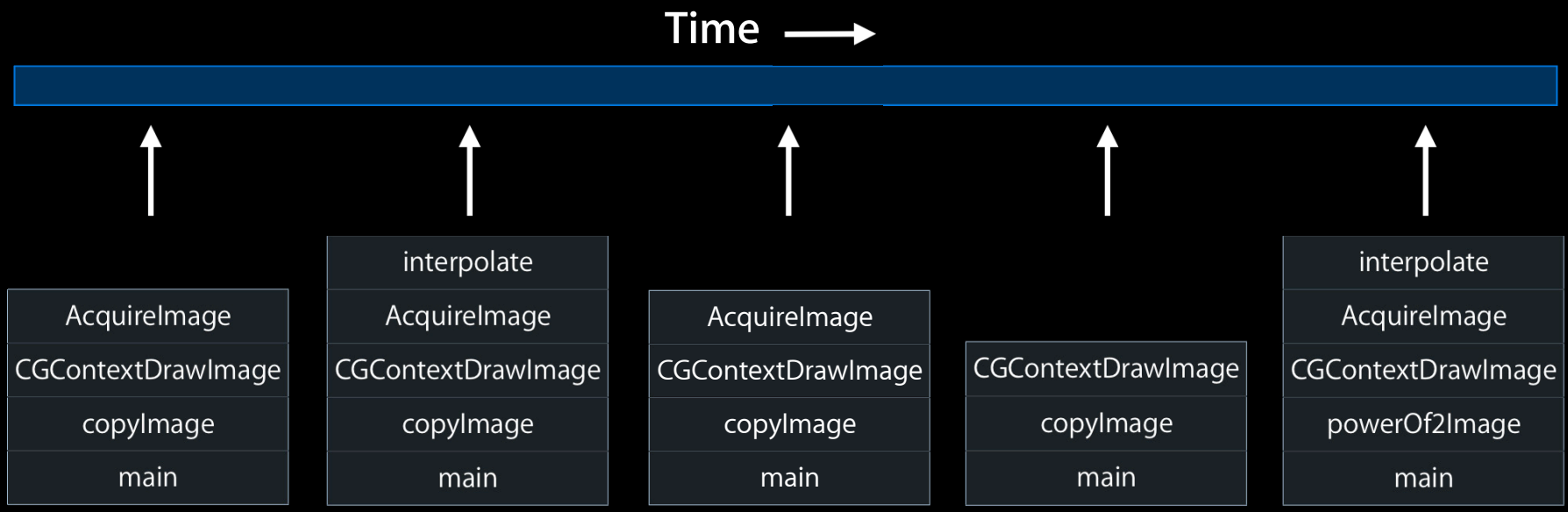
What is Time Profiling?

- Statistical sampling of target at timer interrupts
- Most efficient sampling mechanism
  - Pioneered by Shark
  - Mac and iPhone
- Where is your code spending its time?

Time Profiler

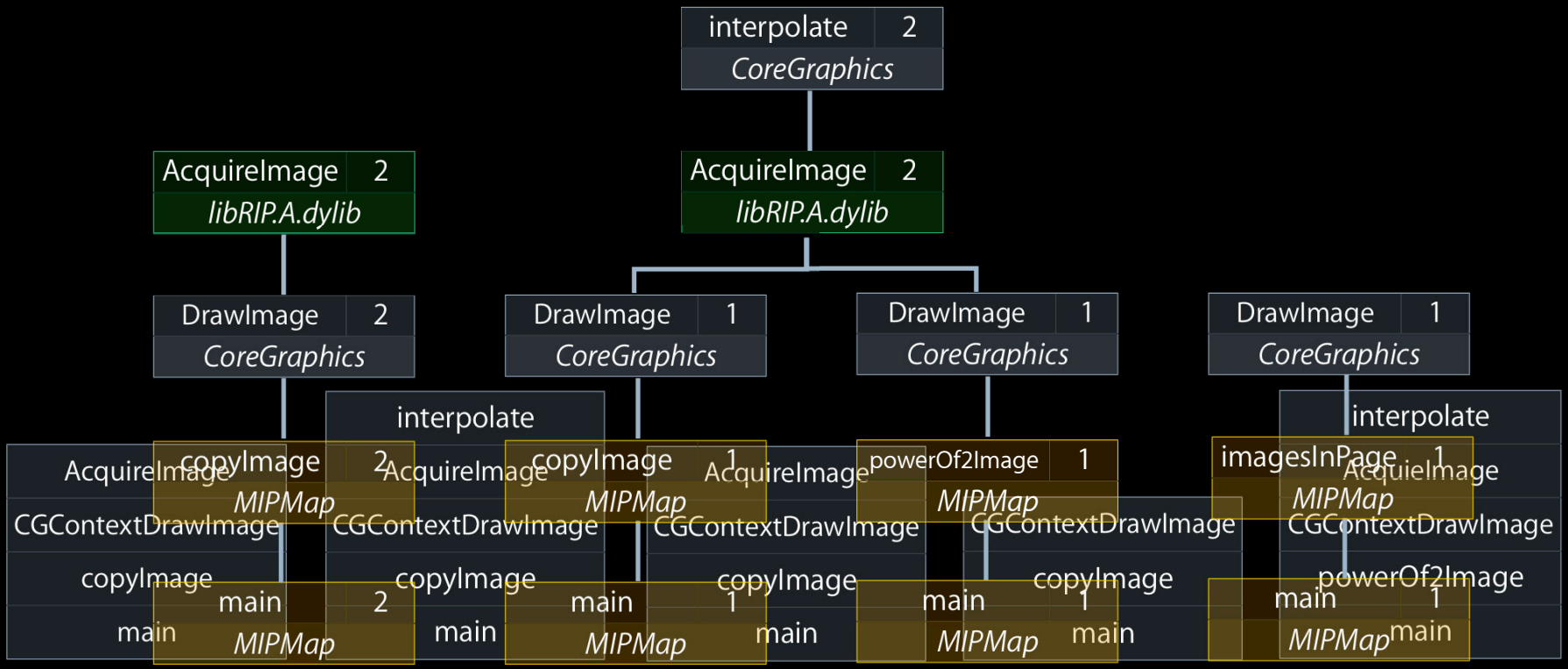
How does it work?

- Periodic samples of threads
- Gathers call stacks
  - Including kernel time and drivers



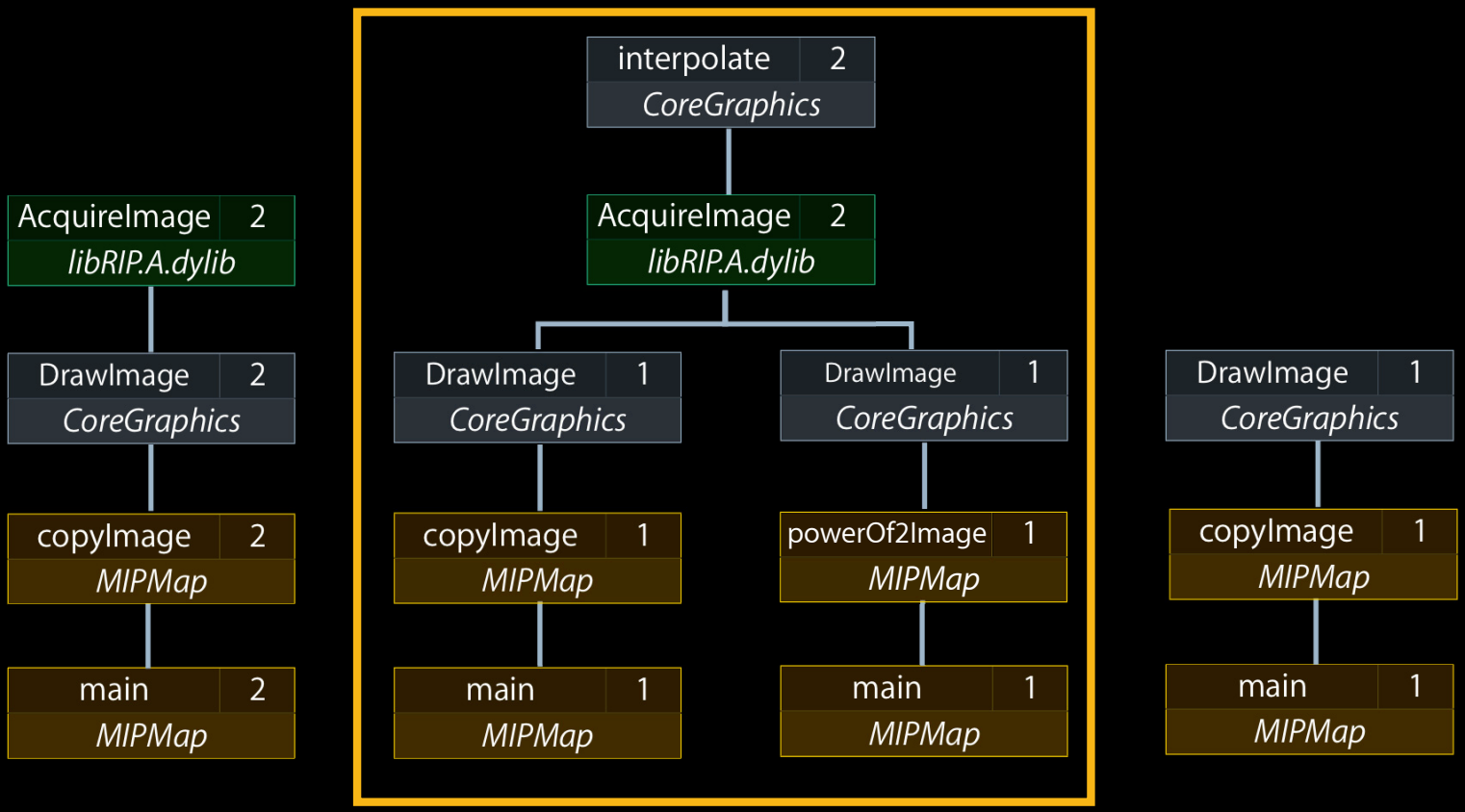
Time Profiler

Building call trees



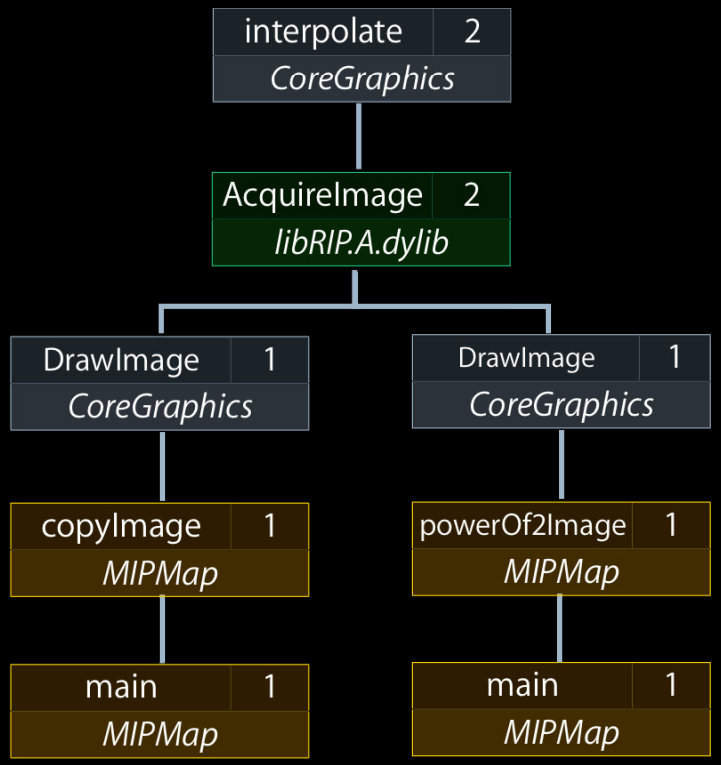
Time Profiler

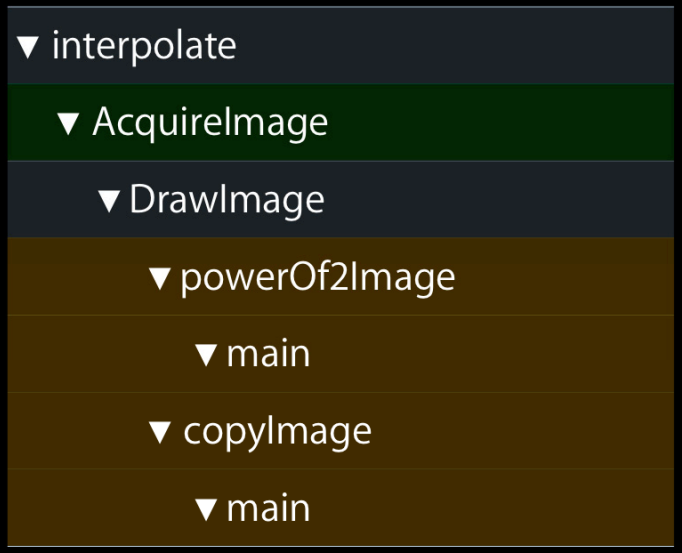
Building call trees



Time Profiler

Building call trees





Time Profiler

Properties

- Low impact, deferred mode
- Performed within kernel
- All vs. running thread states
  - System profiling
  - Kernel stacks

Time Profiler

Recording modes

- Immediate mode—original recording style
  - Allows you to correlate user events with profile
- Can effect your results

Time Profiler

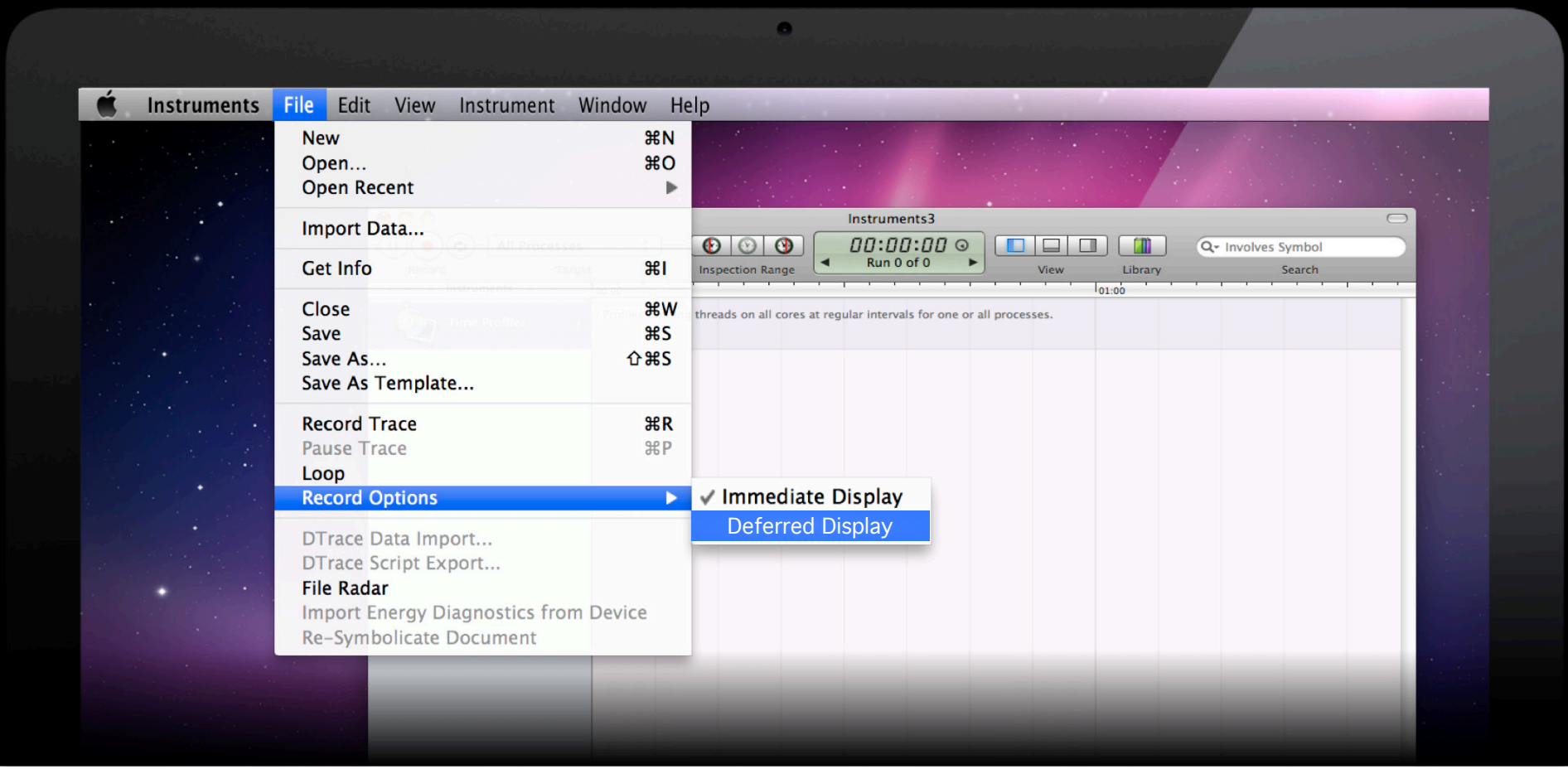
Recording modes

- Deferred mode—new recording style in Instruments
  - Disables track and detail views
  - Defers data processing until end

Minimize Observer Effect

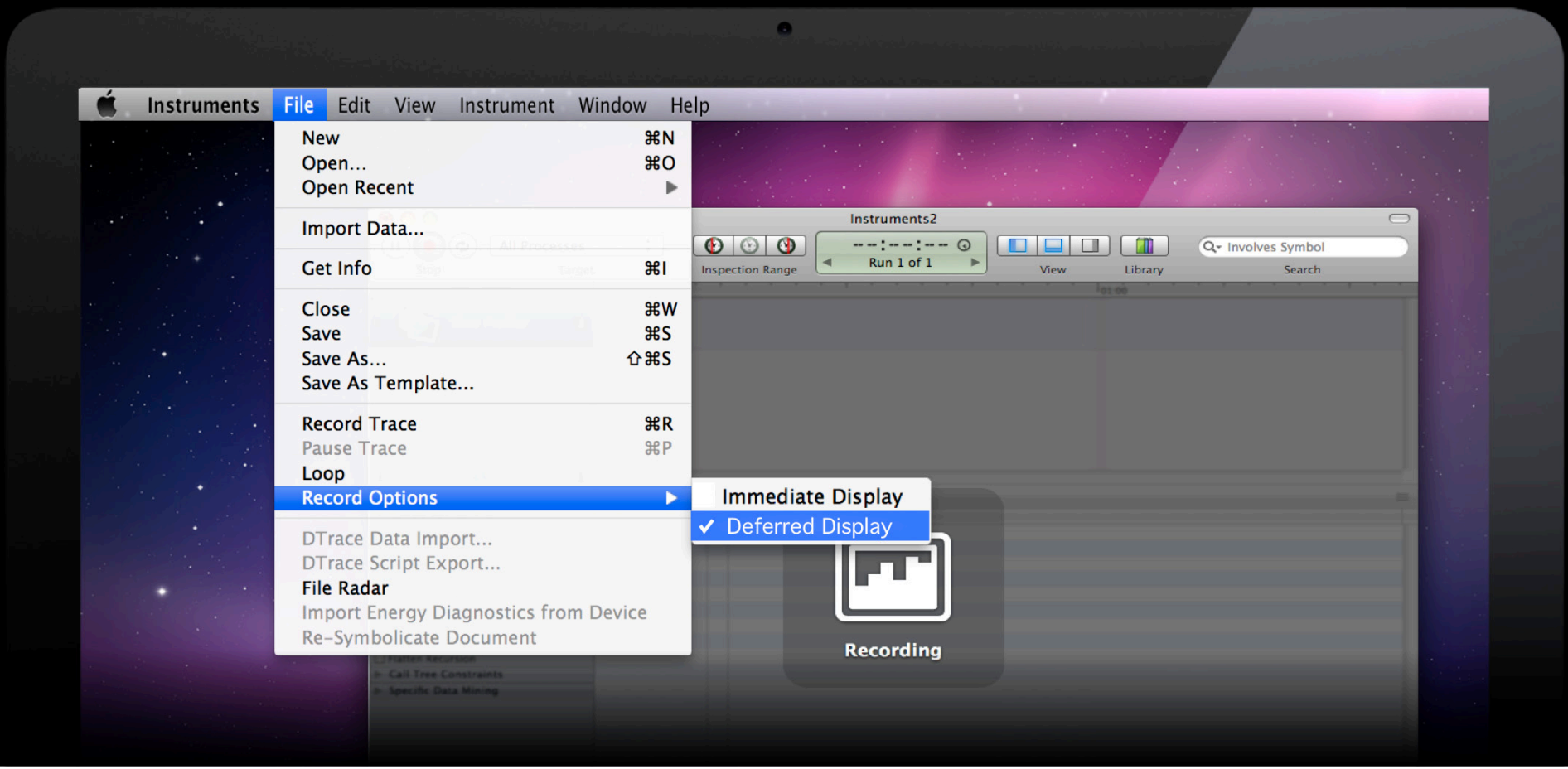
Time Profiler

Deferred mode



Time Profiler

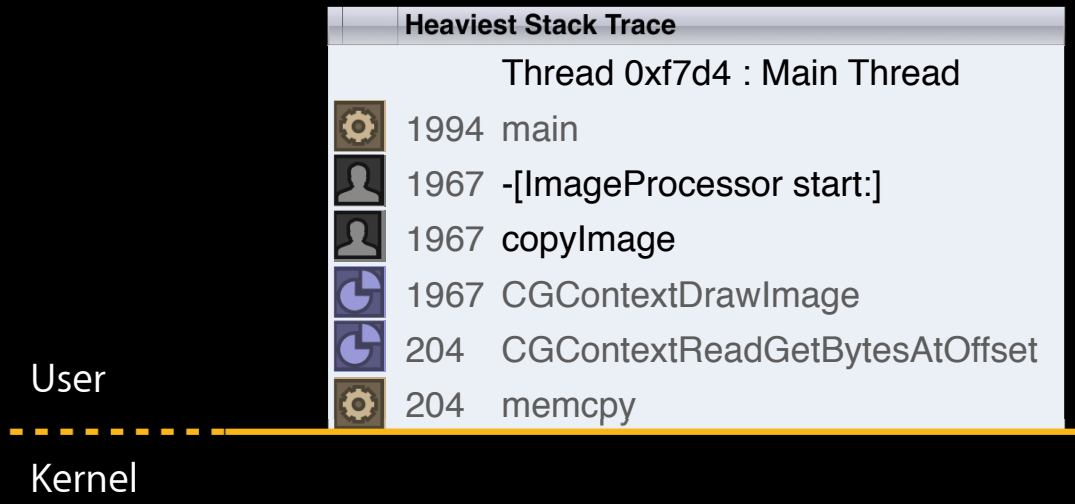
Deferred mode



Time Profiler

Kernel backtraces

- Allow you to "peek under the covers" for system calls
- Optimization of drivers and kernel extensions



Time Profiler

Kernel backtraces

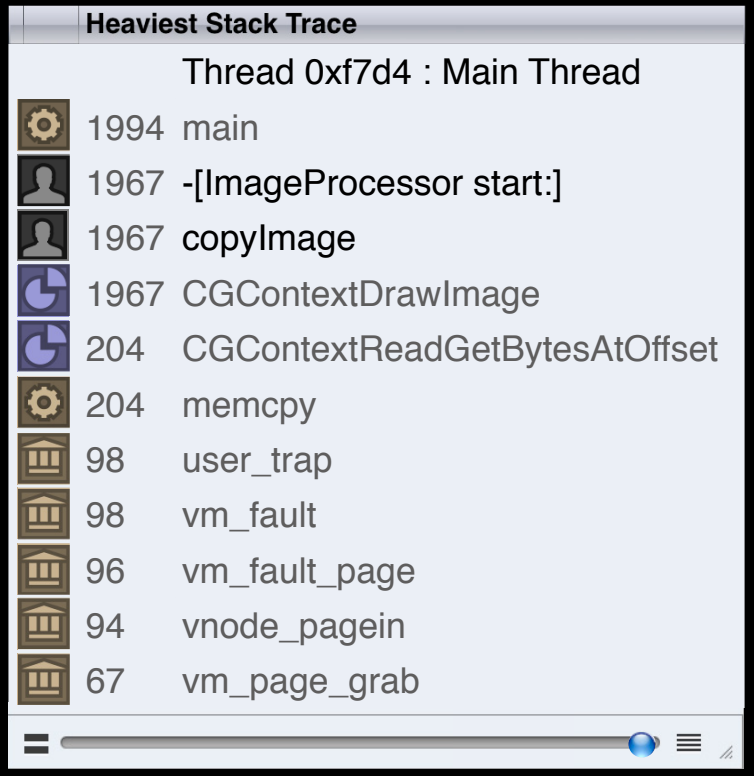
Running Time		Symbol Name
105.0ms	100.0%	▼blkclr mach_kernel
105.0ms	100.0%	▼pmap_zero_page mach_kernel
104.0ms	99.0%	▼vm_fault_copy_cleanup mach_kernel
104.0ms	99.0%	▼vm_fault mach_kernel
102.0ms	97.1%	▼user_trap mach_kernel
31.0ms	29.5%	▼vec_ycc_rgbx_convert libJPEG.dylib
31.0ms	29.5%	▼sep_upsample libJPEG.dylib
24.0ms	22.8%	▼process_data_context_main libJPEG.dylib
24.0ms	22.8%	▼_cg_jpeg_read_scanlines libJPEG.dylib
24.0ms	22.8%	▼copyImageBlockSetJPEG ImageIO
24.0ms	22.8%	▼ImageProviderCopyImageBlockSetCallback ImageIO
24.0ms	22.8%	▼img_blocks_create CoreGraphics
24.0ms	22.8%	▼img_data_lock CoreGraphics
24.0ms	22.8%	▼CGSImageDataLock CoreGraphics
24.0ms	22.8%	▼ripc_AcquireImage libRIP.A.dylib
24.0ms	22.8%	▼ripc_DrawImage libRIP.A.dylib
24.0ms	22.8%	▼CGContextDrawImage CoreGraphics
24.0ms	22.8%	▼scaleImage MIPMap
24.0ms	22.8%	▼__-[ImageProcessor start:]_block_invoke_3 MIPMap
24.0ms	22.8%	▼_dispatch_call_block_and_release libSystem.B.dylib
24.0ms	22.8%	▼_dispatch_worker_thread2 libSystem.B.dylib
24.0ms	22.8%	▼_pthread_wqthread libSystem.B.dylib
24.0ms	22.8%	start_wqthread libSystem.B.dylib

Kernel

User

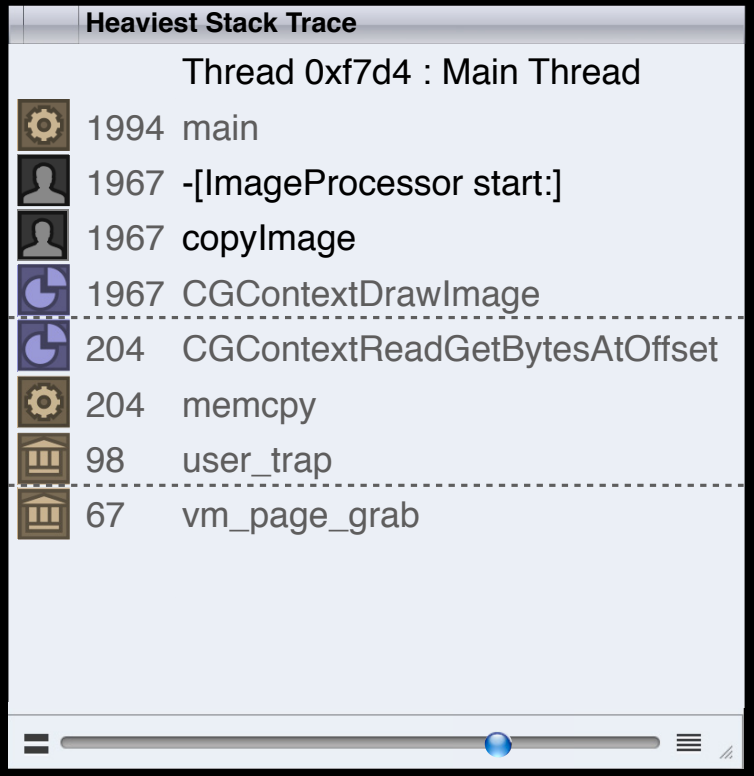
Time Profiler

Stack compression



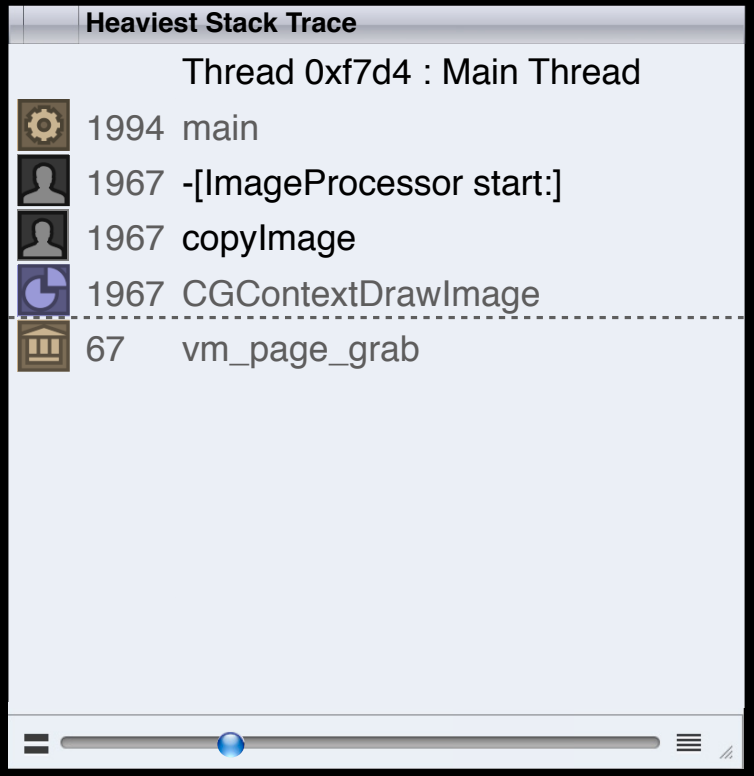
Time Profiler

Stack compression



Time Profiler

Stack compression



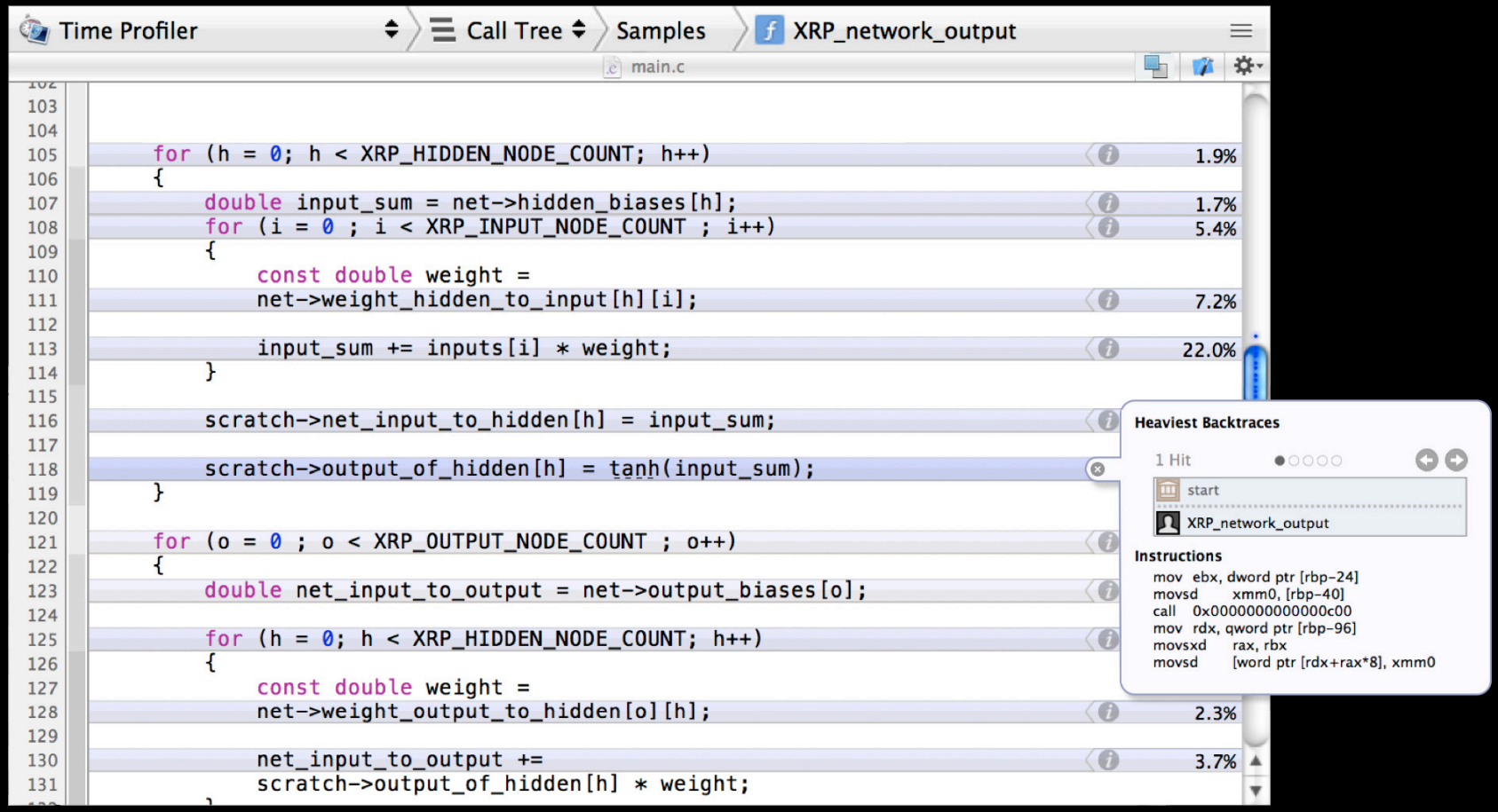
Time Profiler

Disassembly view

- Find precise function call in nested calls
- Effectiveness of template expansion and inlining
- Optimize tight loops
- Analyze code when source unavailable

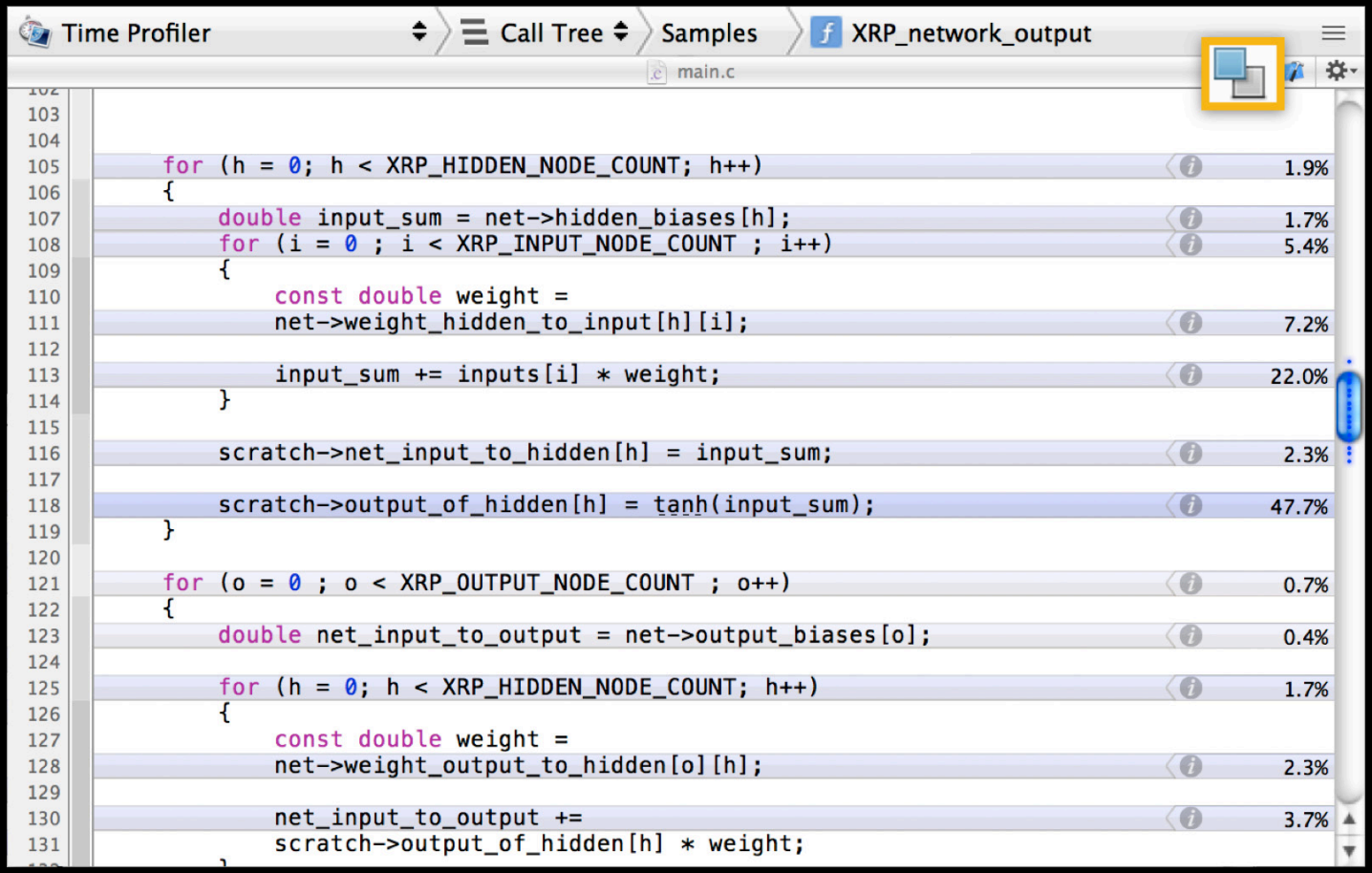
Time Profiler

Disassembly view



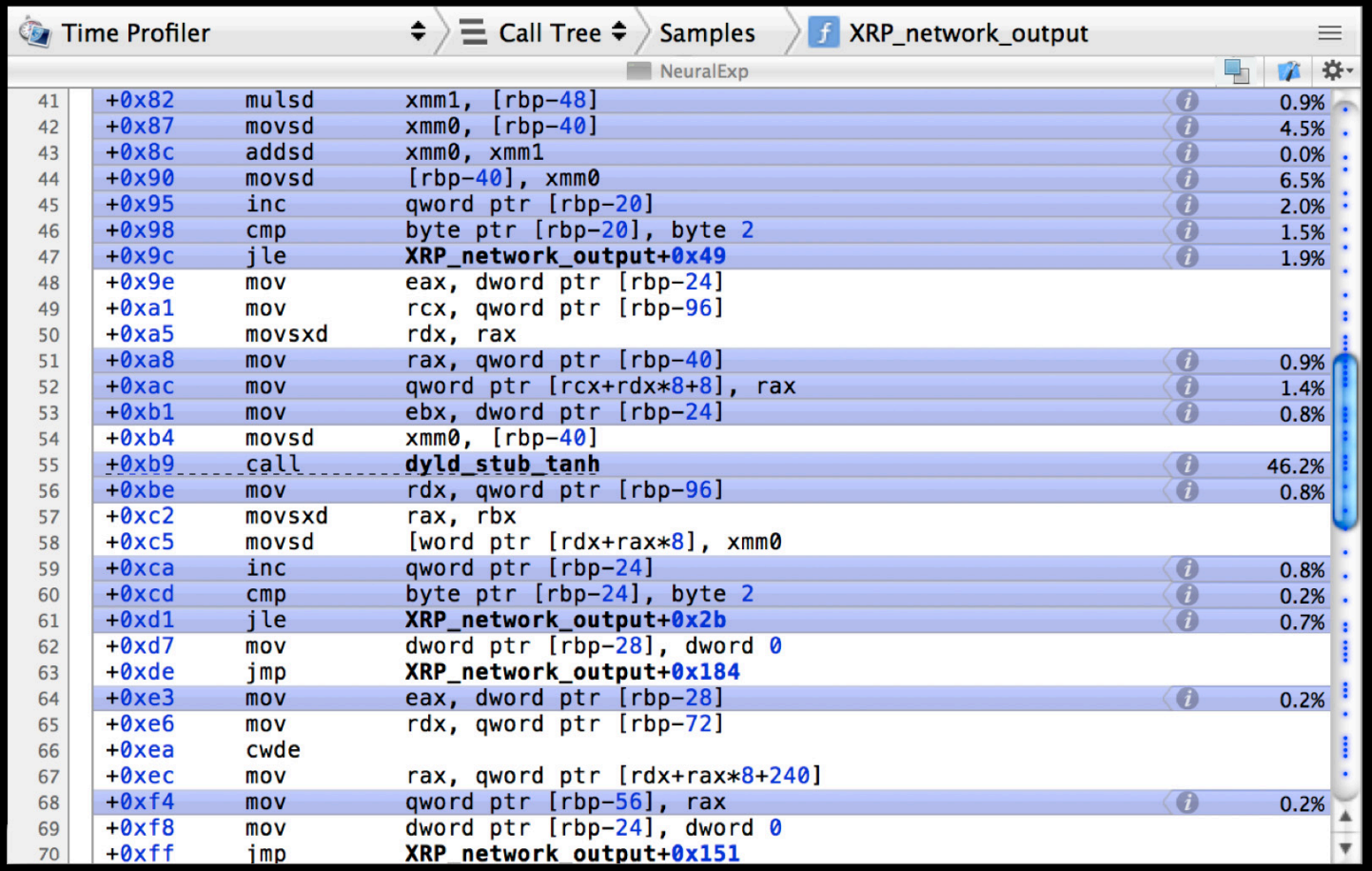
Time Profiler

Disassembly view



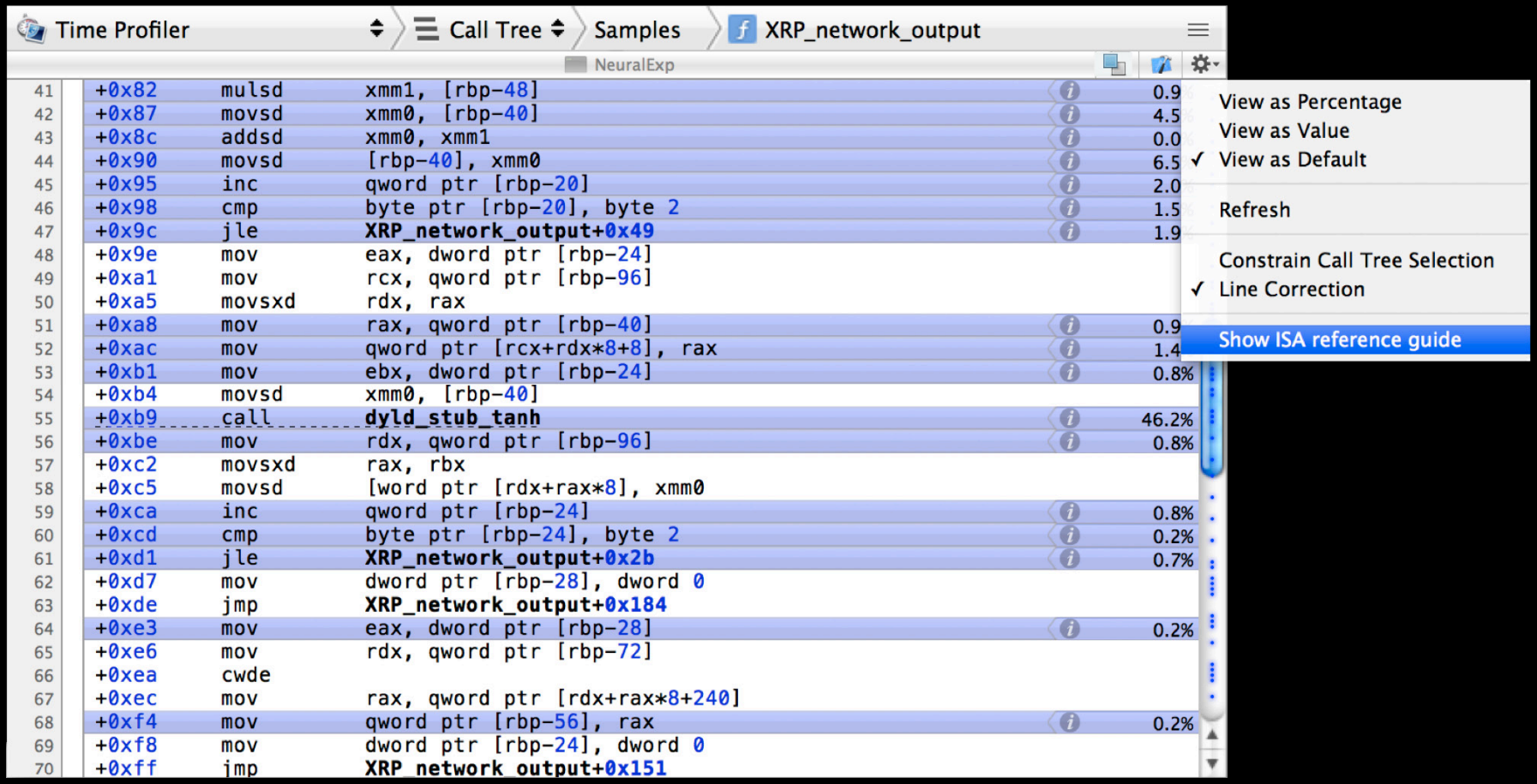
Time Profiler

Disassembly view



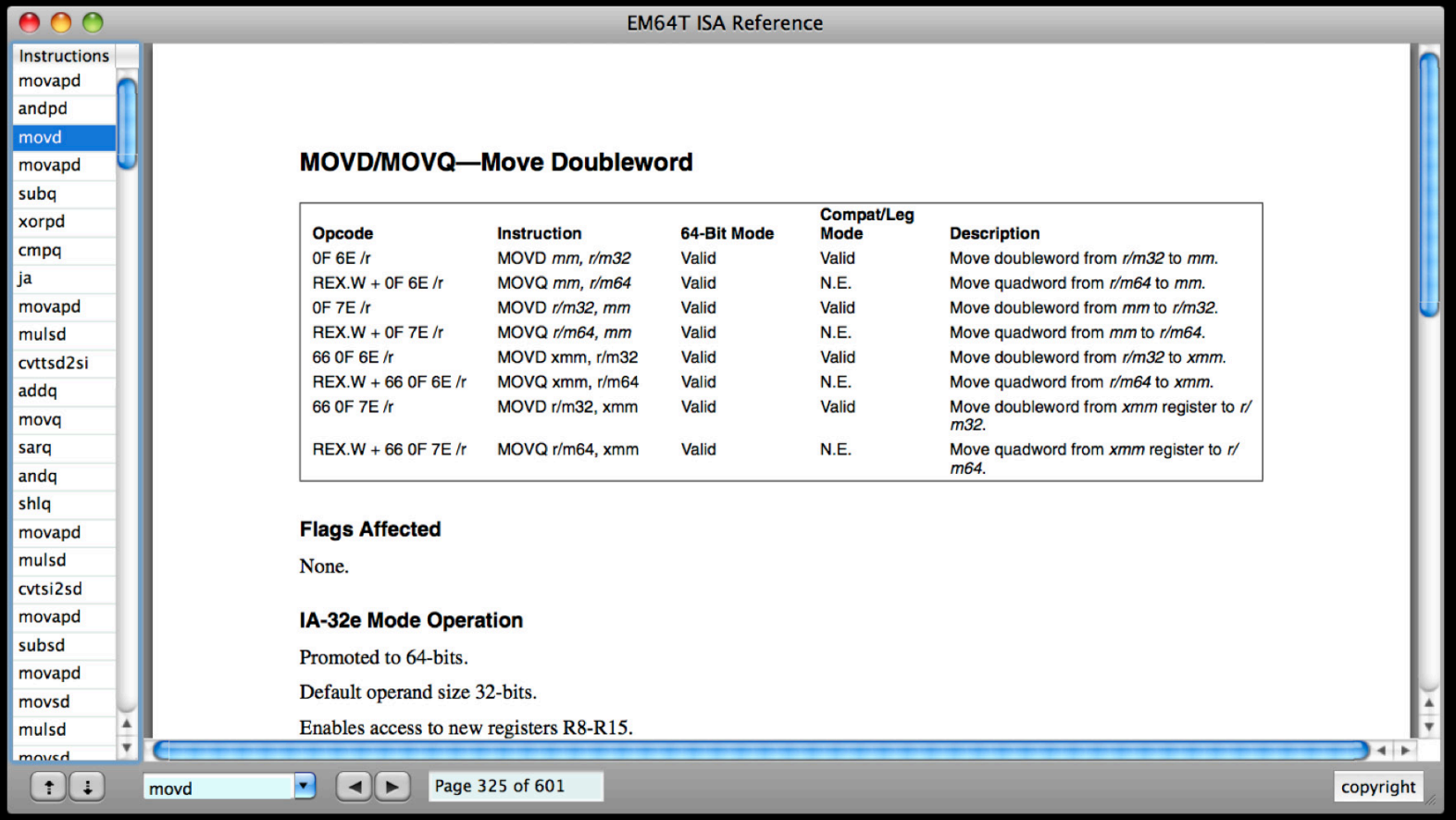
Time Profiler

Disassembly view



Time Profiler

Disassembly view



MOVD/MOVQ—Move Doubleword

Opcode	Instruction	64-Bit Mode	Compat/Leg Mode	Description
0F 6E /r	MOVD *mm, r/m32*	Valid	Valid	Move doubleword from *r/m32* to *mm*.
REX.W + 0F 6E /r	MOVQ *mm, r/m64*	Valid	N.E.	Move quadword from *r/m64* to *mm*.
0F 7E /r	MOVD *r/m32, mm*	Valid	Valid	Move doubleword from *mm* to *r/m32*.
REX.W + 0F 7E /r	MOVQ *r/m64, mm*	Valid	N.E.	Move quadword from *mm* to *r/m64*.
66 0F 6E /r	MOVD xmm, r/m32	Valid	Valid	Move doubleword from *r/m32* to *xmm*.
REX.W + 66 0F 6E /r	MOVQ xmm, r/m64	Valid	N.E.	Move quadword from *r/m64* to *xmm*.
66 0F 7E /r	MOVD r/m32, xmm	Valid	Valid	Move doubleword from *xmm* register to *r/m32*.
REX.W + 66 0F 7E /r	MOVQ r/m64, xmm	Valid	N.E.	Move quadword from *xmm* register to *r/m64*.

Flags Affected

None.

IA-32e Mode Operation

Promoted to 64-bits.

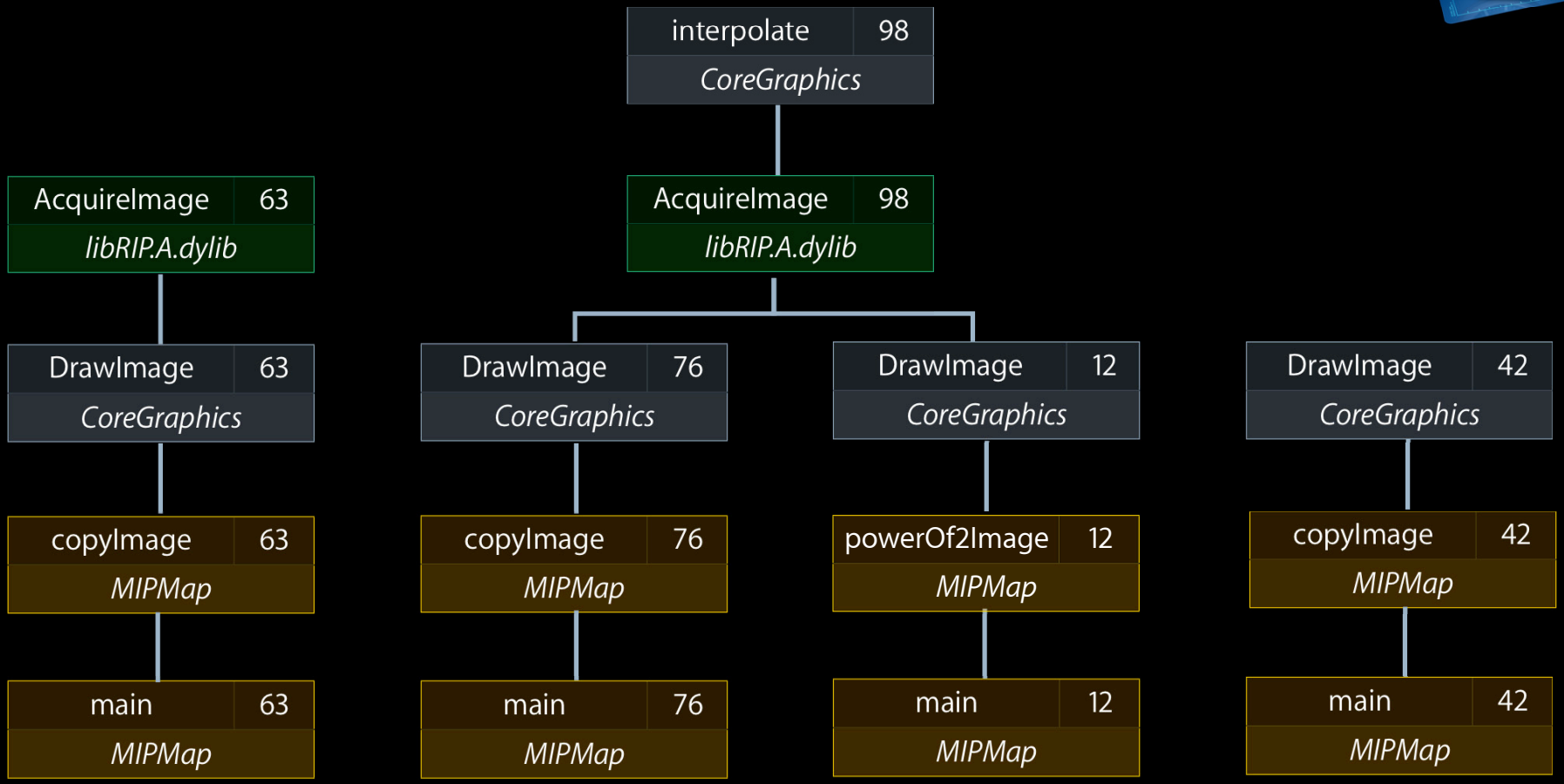
Default operand size 32-bits.

Enables access to new registers R8-R15.

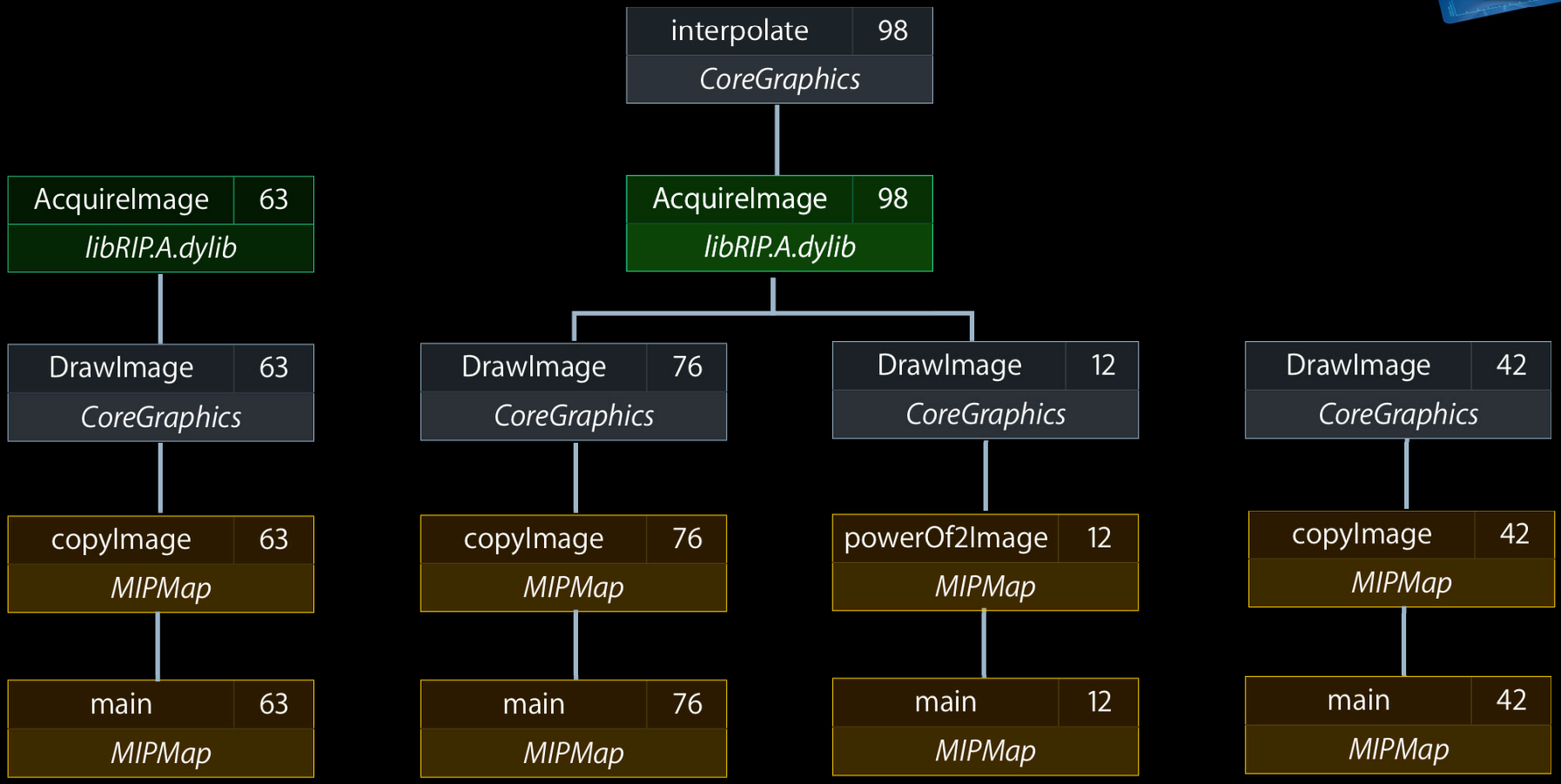
Demo

Time Profiler

Data Mining

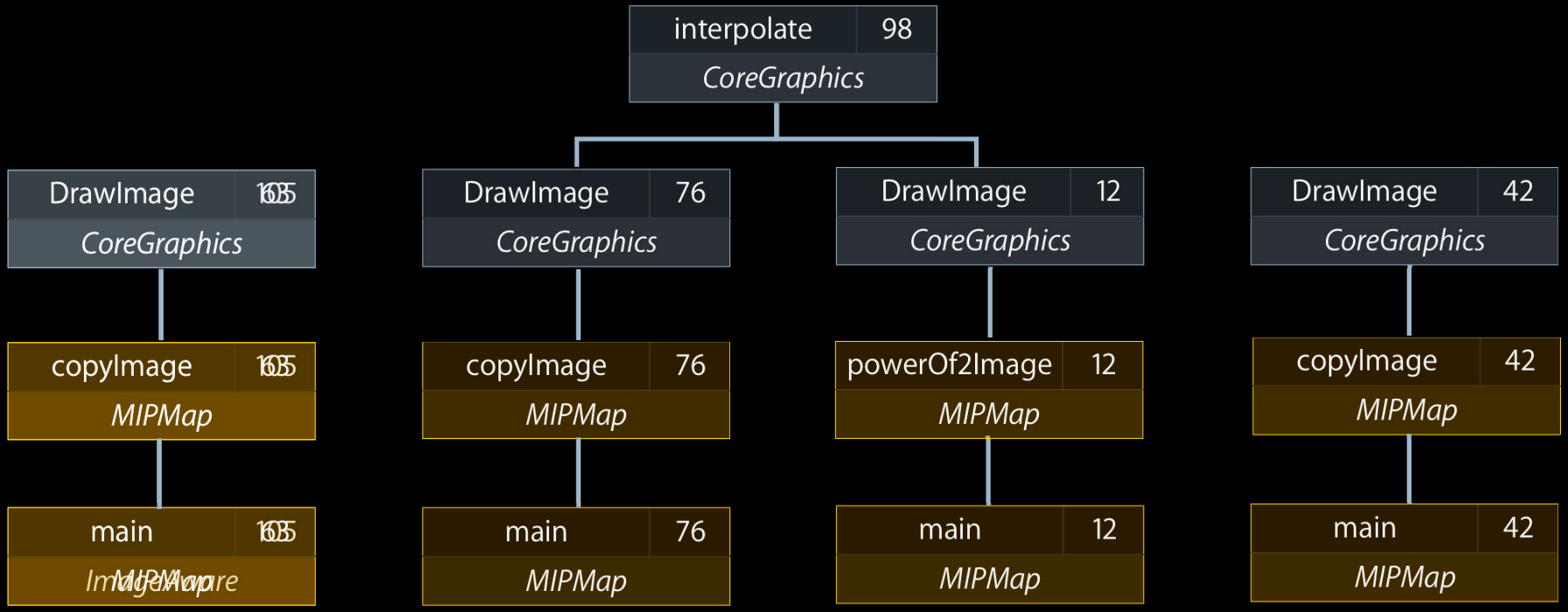

interpolate 98
CoreGraphics
AcquireImage 63
libRIP.A.dylib
AcquireImage 98
libRIP.A.dylib
DrawImage 63
CoreGraphics
DrawImage 76
CoreGraphics
DrawImage 12
CoreGraphics
DrawImage 42
CoreGraphics
copyImage 63
MIPMap
copyImage 76
MIPMap
powerOf2Image 12
MIPMap
copyImage 42
MIPMap
main 63
MIPMap
main 76
MIPMap
main 12
MIPMap
main 42
MIPMap

Data Mining



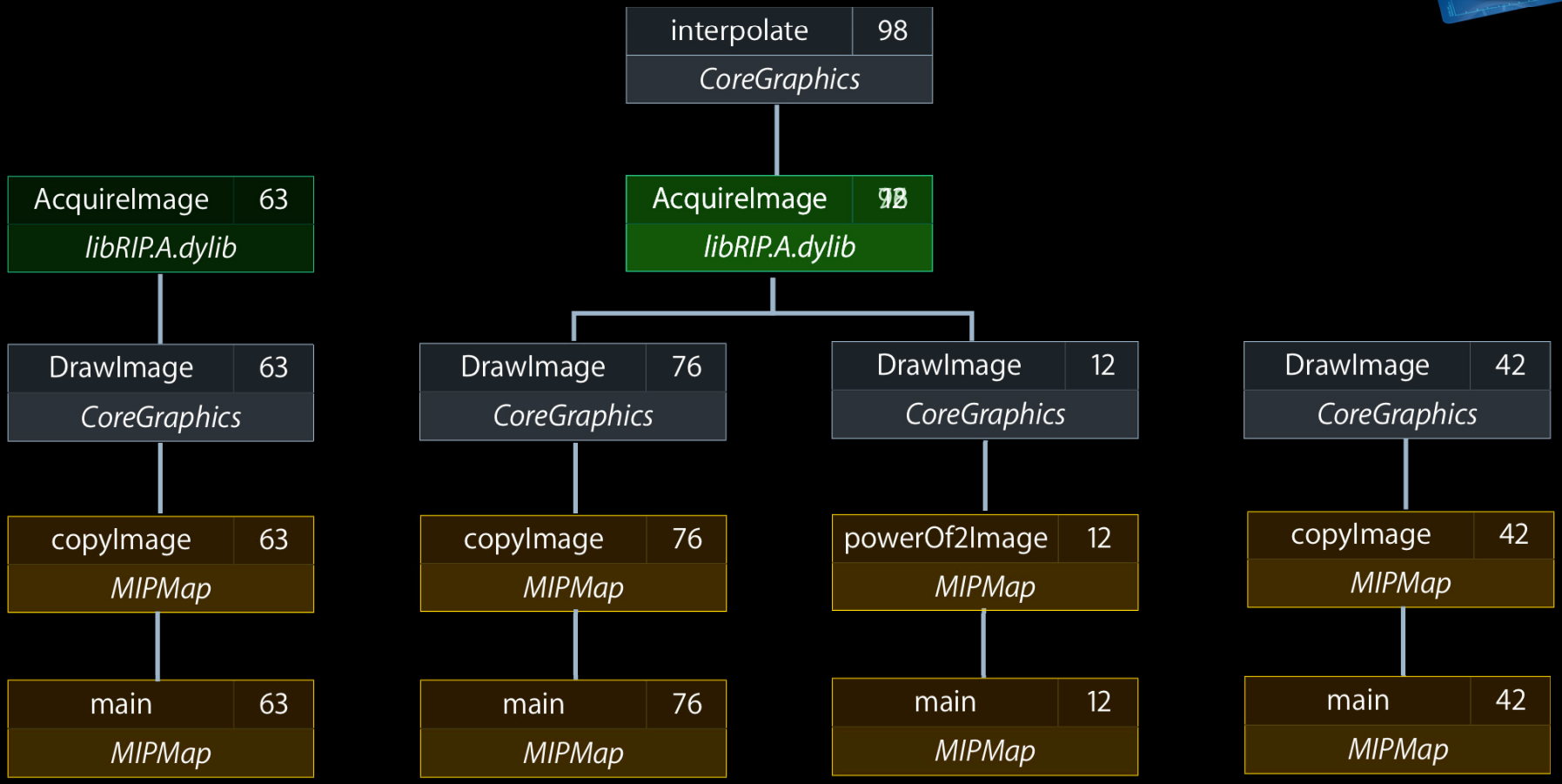
Charge symbol to caller

Data Mining



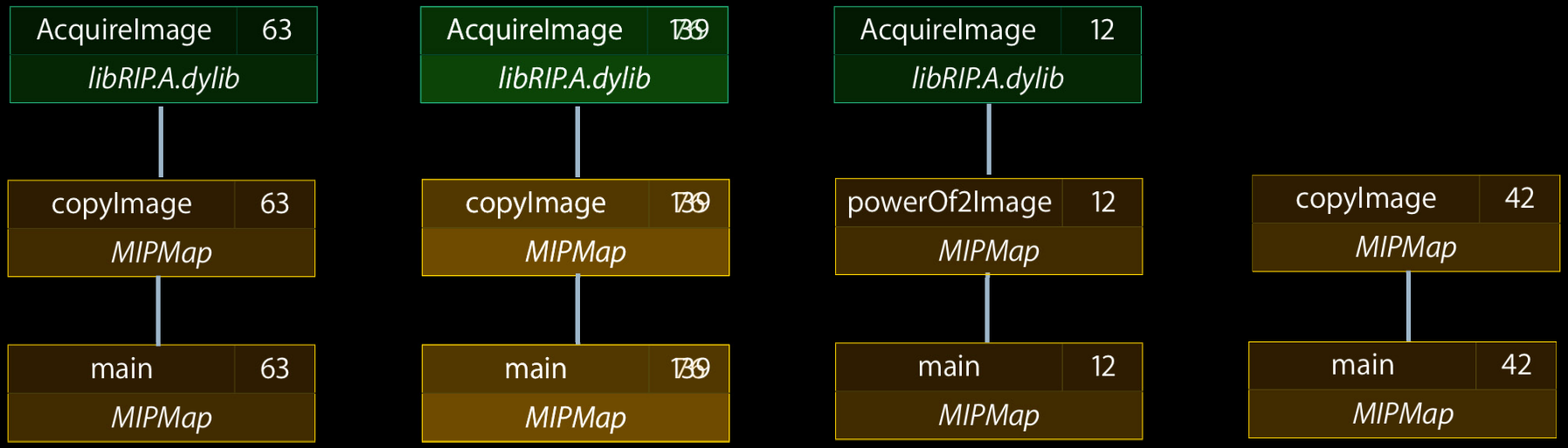
Charge symbol to caller

Data Mining



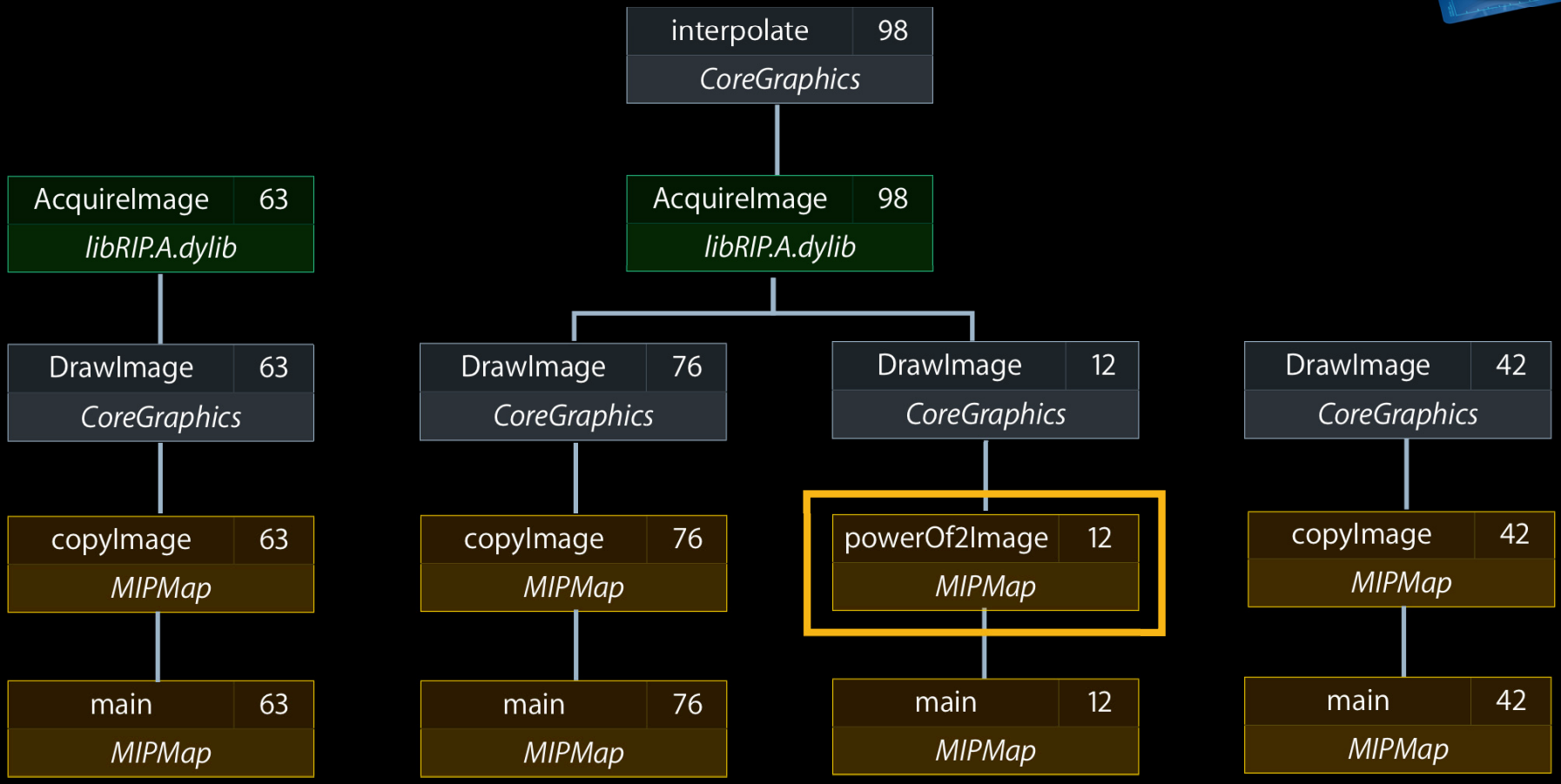
Charge library to caller

Data Mining



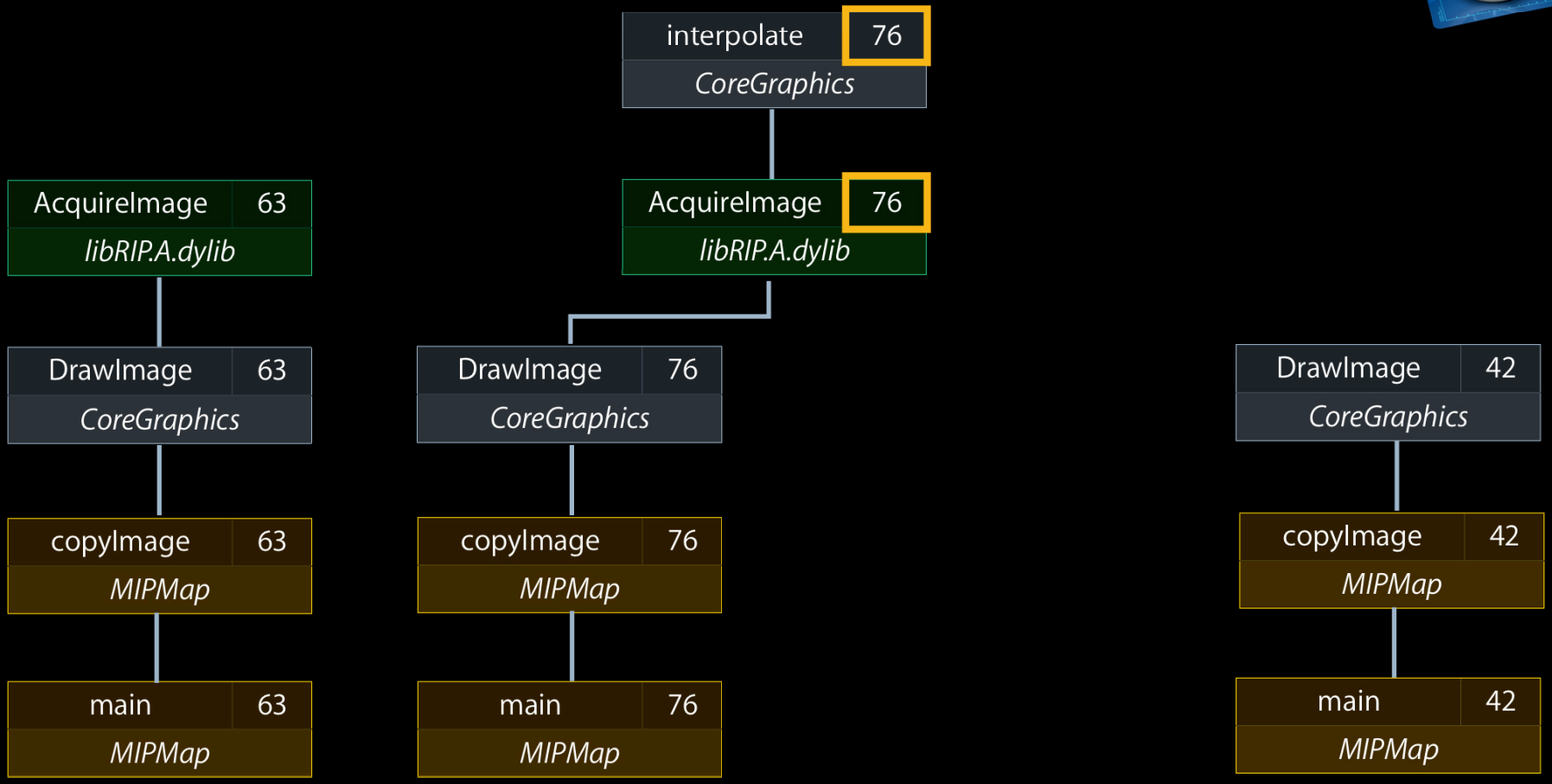
Charge library to caller

Data Mining



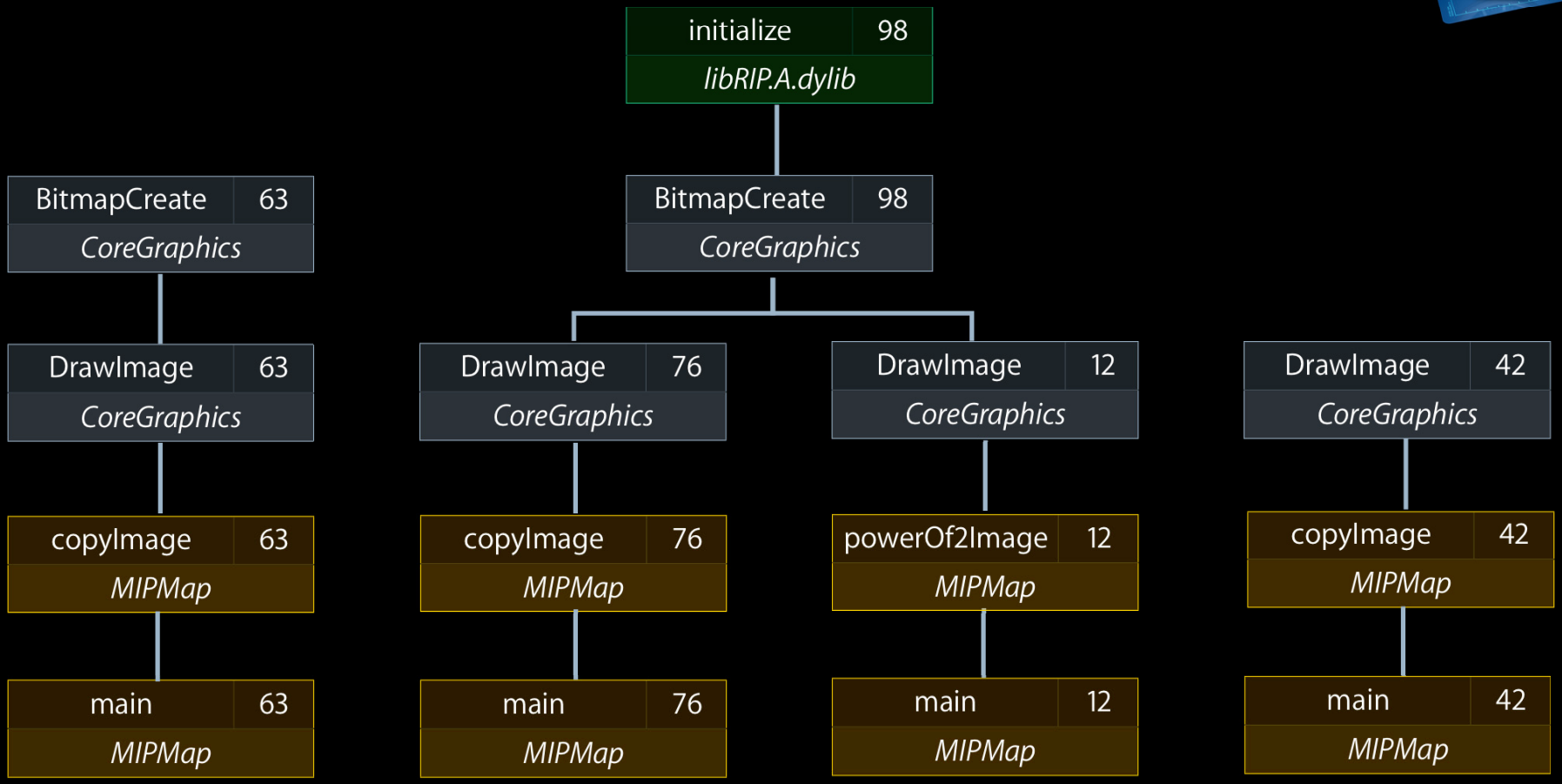
Prune symbol and subtree

Data Mining



Prune symbol and subtree

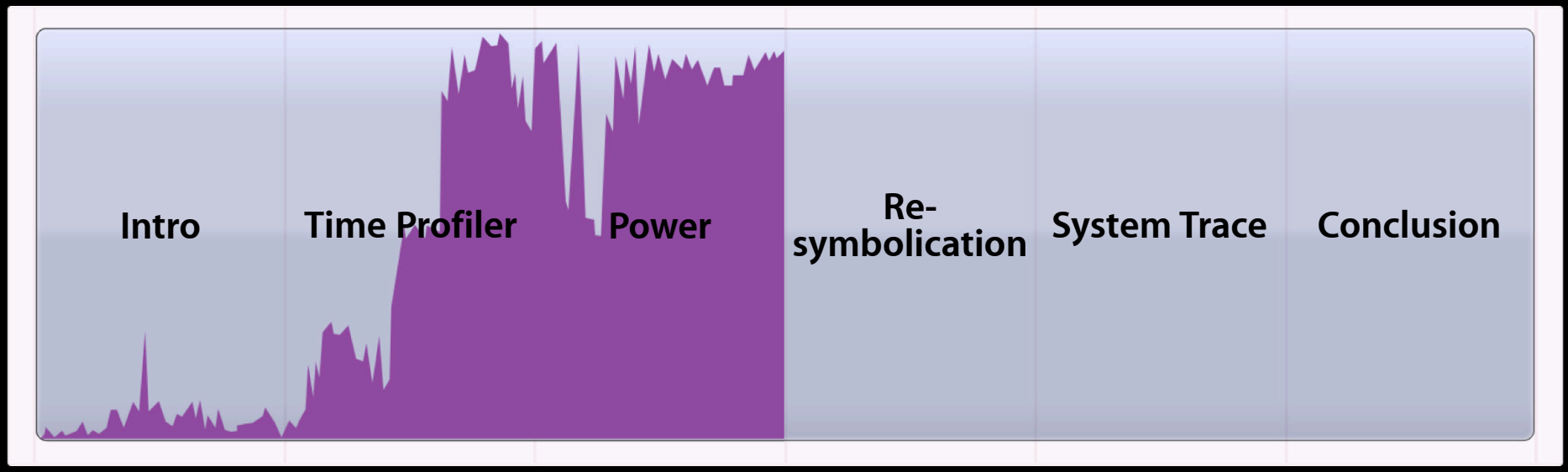
Data Mining



Flatten library to boundaries

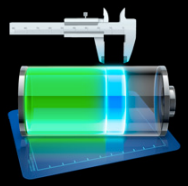
Advanced Performance Analysis with Instruments

Chad Woolf

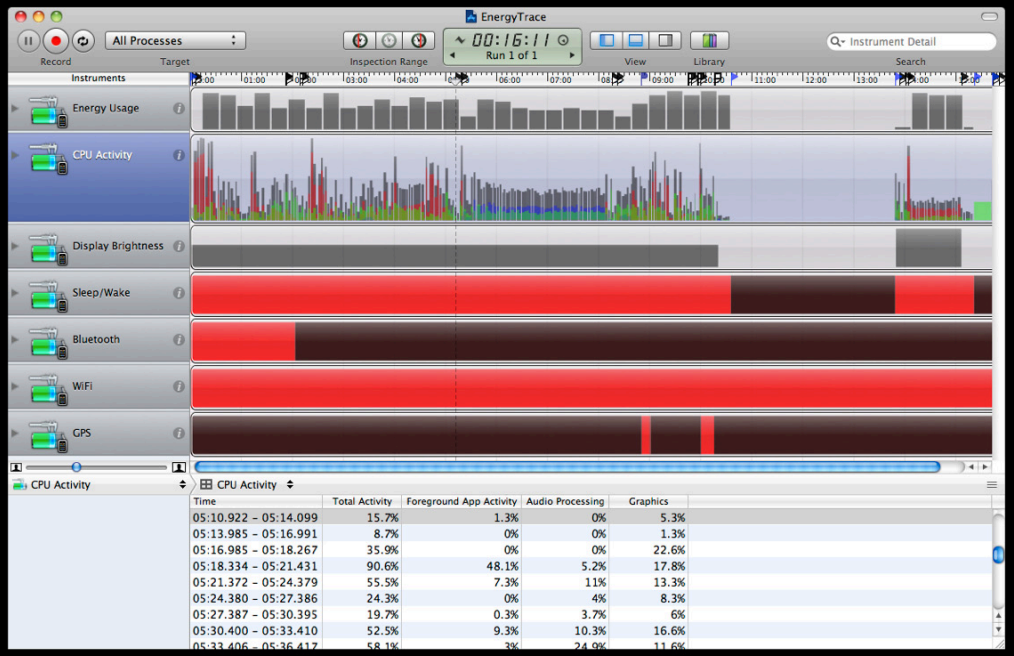

Intro
Time Profiler
Power
Re-
symbolication
System Trace
Conclusion

iPhone SDK 4

Power analysis tools

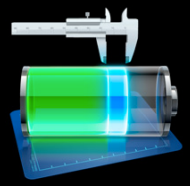


- Energy usage (power)
- CPU activity
- Sleep/Wake
- Display brightness
- Radios
- Energy diagnostics template



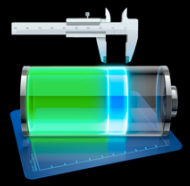
Tricks of the Trade

Mobile devices work differently at home

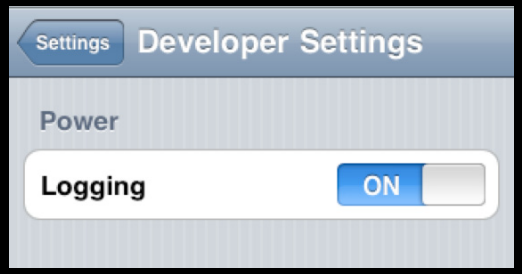


- When connected to USB
  - True Sleep/Wake is disabled
  - Energy is flowing into the battery
  - Data is flowing to the host
- When in the field
  - Diverse radio environment
  - Only way to test location updates

Tricks of the Trade

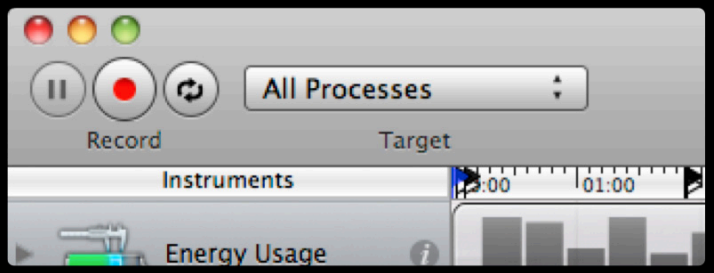


Record in the field



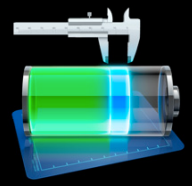
or

Record while connected



Energy Usage

True power measurement

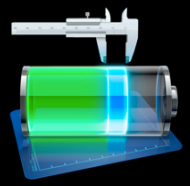


- Records energy flowing from the battery
- Energy usage factor from 1 to 20
  - low number ➔ long battery life
  - high number ➔ short battery life
- Works with
  - iPhone 3GS
  - iPhone 4
  - iPod Touch second and third generation

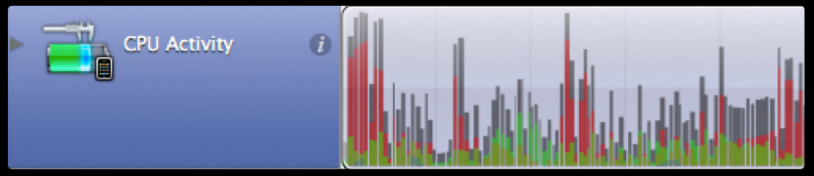
Level	Time to Discharge a Full Battery
20	Less than 1 hour
10	About 10 hours
1	20+hours

CPU Activity

Track important system activity



- CPU activity is a big part of your energy profile
- Power scales with CPU load
  - More load = more power
- Measurements
  - Total activity
  - Foreground app activity
  - Graphics/render (springboard)
  - Audio/video (mediaserverd)

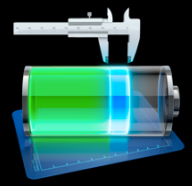


CPU Activity

Time	Total Activity	Foreground App Activity	Audio Processing	Graphics
05:10.922 – 05:14.099	15.7%	1.3%	0%	5.3%
05:13.985 – 05:16.991	8.7%	0%	0%	1.3%
05:16.985 – 05:18.267	35.9%	0%	0%	22.6%
05:18.334 – 05:21.431	90.6%	48.1%	5.2%	17.8%
05:21.372 – 05:24.379	55.5%	7.3%	11%	13.3%
05:24.380 – 05:27.386	24.3%	0%	4%	8.3%
05:27.387 – 05:30.395	19.7%	0.3%	3.7%	6%
05:30.400 – 05:33.410	52.5%	9.3%	10.3%	16.6%
05:33.406 – 05:36.417	58.1%	3%	24.9%	11.6%

CPU Activity

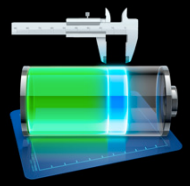
Work tends to move around



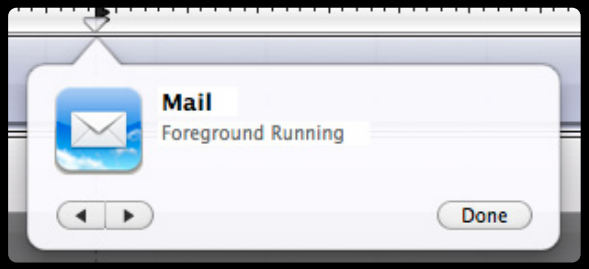
- Graphics/render service
  - CoreAnimation
- Audio/video service
  - AV Foundation
  - Media Player Framework
  - OpenAL

CPU Activity

Tracking important system activity

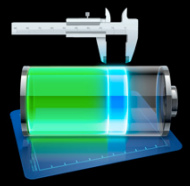


- Tracks app state changes
  - Background/foreground
  - Background suspended
  - Foreground obscured

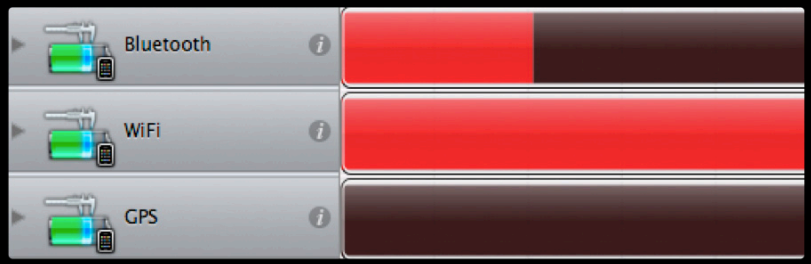


Radio Instrumentation

Track radio states

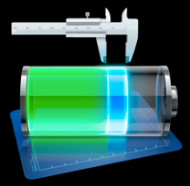


- Bluetooth
  - Enabled/disabled
- Wi-Fi
  - Enabled
    - More efficient location
    - More efficient data transfer
  - Disabled
    - Data transfers must use 3G or EDGE

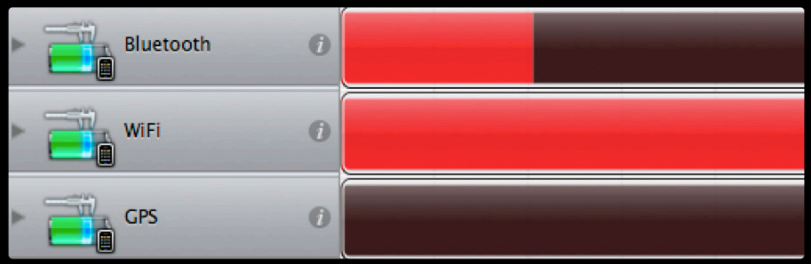


Radio Instrumentation

GPS state



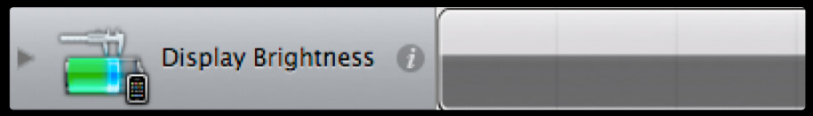
- Application driven via CoreLocation
- Enabled
  - Higher power
  - Needed for high accuracy location
- Should be disabled when not needed
  - Using CLLocationManager
  - Lower desiredAccuracy
  - Greater distanceFilter
  - Call stopUpdatingLocation



Display Brightness

Track display activity

- Backlight consumes energy
- User brightness setting recorded
- Becomes zero when off
  - Sleep/Wake button
  - Idle dim



Sleep/Wake

Extend your battery life with sleep

- Sleep should occur
  - ~15 seconds after screen is off
  - Background applications may extend that
- Wakes will occur
  - Local notifications
  - Push notifications
  - Location updates

Demo Scenario

What we'll see

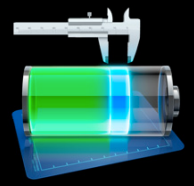
- Turning on logging
- Importing a trace
- Finding interesting events like
  - A wake caused by a push notification
  - Watching a video on YouTube
  - GPS activity

Demo

Power Instrumentation

Tricks of the Trade

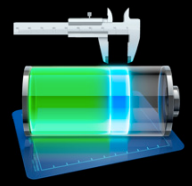
Controlling your environment



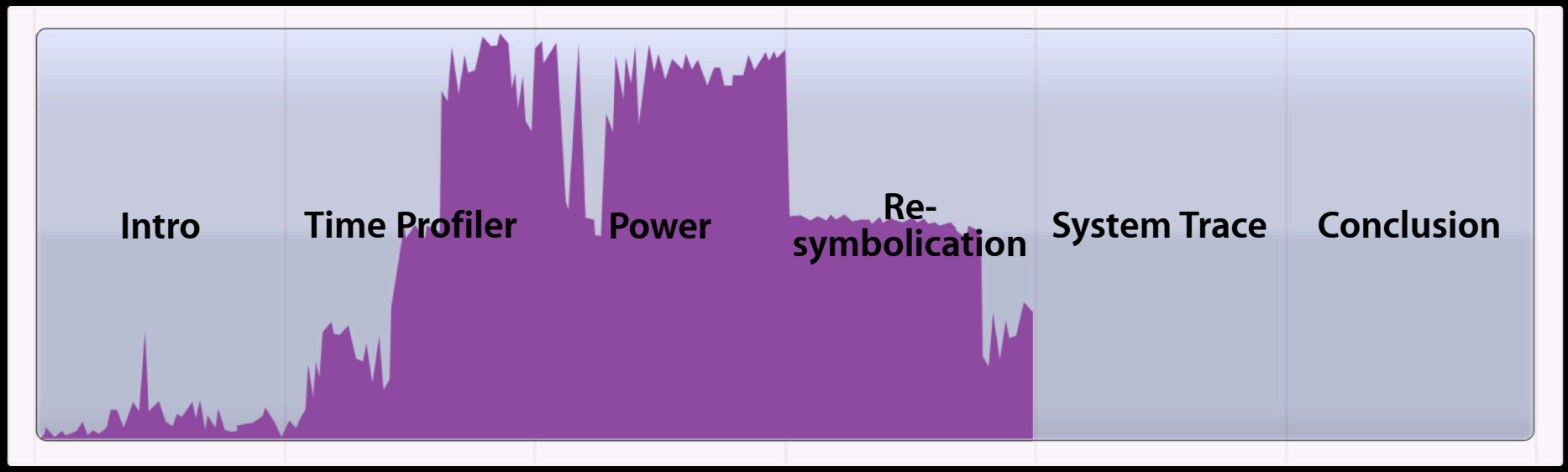
- Controlling iOS 4 features
  - Mail fetching (disable)
  - Push notifications (disable)
  - Auto-dimming (disable)
- Radios
  - Disable Wi-Fi to test 3G or EDGE transfers
  - Disable Bluetooth
- Temperature affects energy usage

Power Analysis Tools

Summary



- New template for understanding energy usage
- Extend battery life
- Record data while in field
- New in iPhone SDK 4


Intro
Time Profiler
Power
Re-
symbolication
System Trace
Conclusion

Missing Symbols

Running Time ▼		Symbol Name	
33.0ms	0.5%	0x394b	PictureViewer
31.0ms	0.5%	0x3b9c	PictureViewer
28.0ms	0.4%	0x3926	PictureViewer
24.0ms	0.4%	0x4826	PictureViewer
22.0ms	0.3%	0x3870	PictureViewer
19.0ms	0.3%	0x39ca	PictureViewer
19.0ms	0.3%	0x3fca	PictureViewer
15.0ms	0.2%	0x3886	PictureViewer
13.0ms	0.2%	0x3bde	PictureViewer
12.0ms	0.2%	0x3bd0	PictureViewer

Missing Symbols

Before iPhone SDK 4

- Solution
  - Rebuild your application
  - Install
  - Re-run your test
- Reason
  - Symbol information remains on the host
  - Application binary remains on the host
  - Cleaning destroys all that
  - A rebuilt binary has a different UUID

Missing Symbols

Before iPhone SDK 4

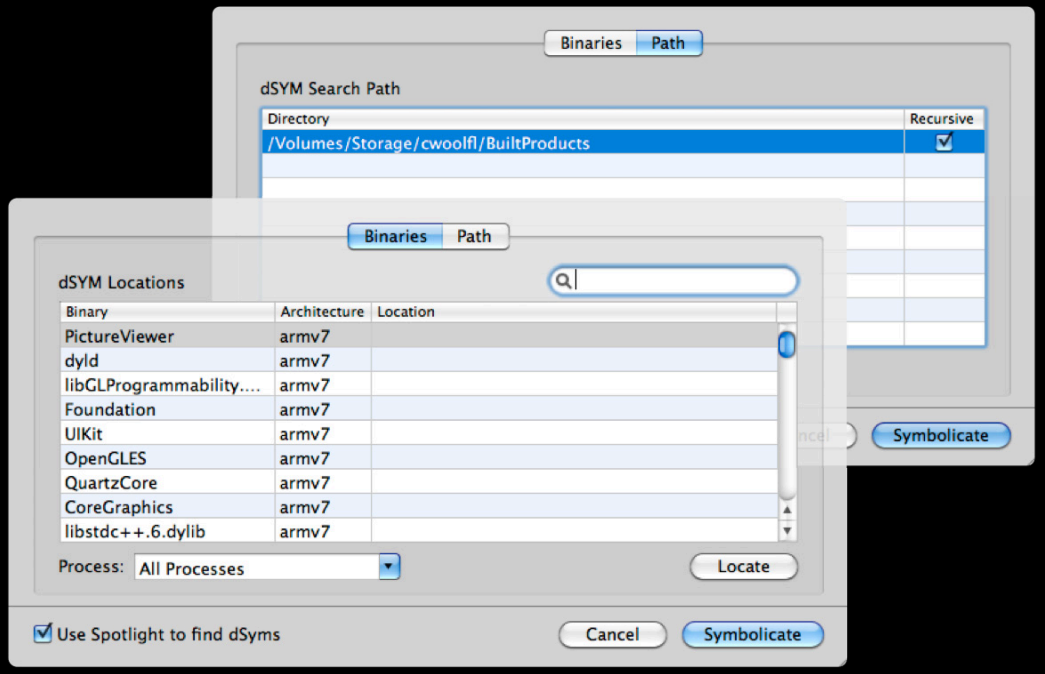
- Other reasons
  - Instruments uses Spotlight to find symbol information
  - Spotlight can't see into
    - Compressed archives
    - Some network shares
    - Places it's told not to look
    - /tmp

Missing Symbols

Enter Re-Symbolication

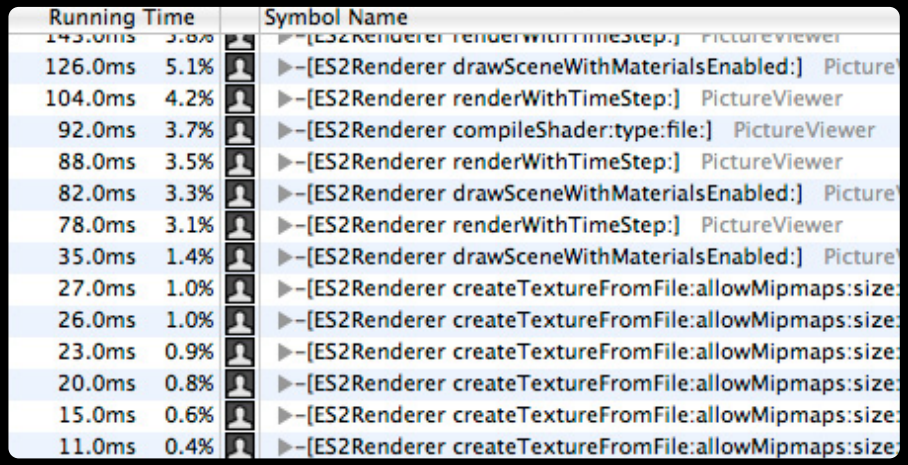
iOS 4

- Manually locate the missing symbol information
- Under the File menu
  - Re-Symbolicate Document
- Locate the dSYM
- Add dSYM search paths

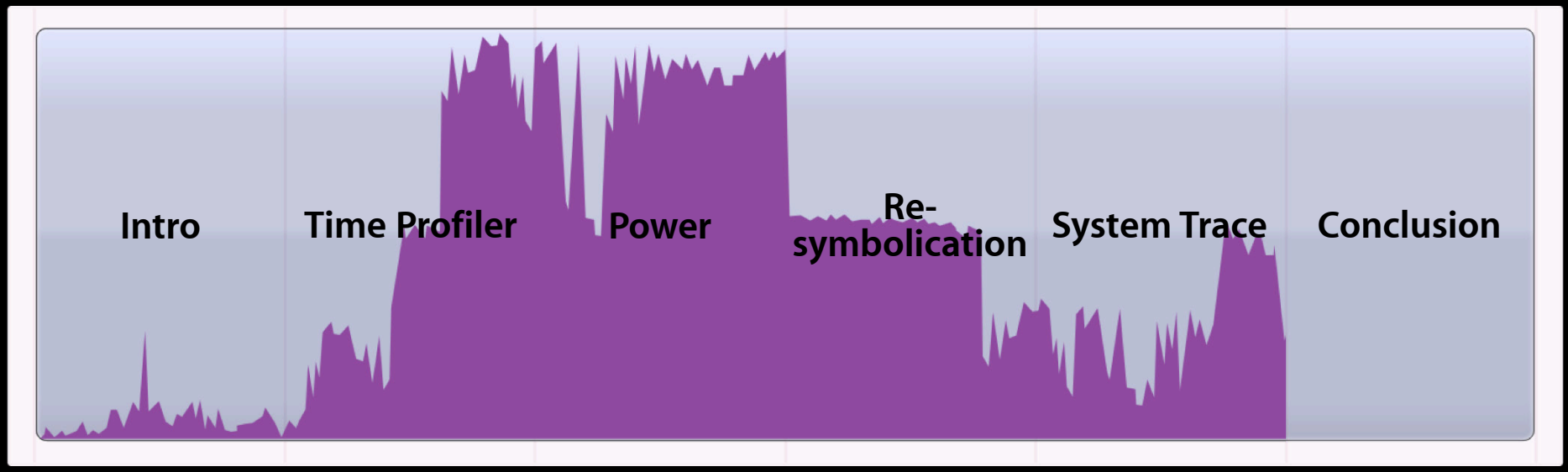


Re-Symbolicated

iOS 4



Running Time		Symbol Name
126.0ms	5.1%	-[ES2Renderer drawSceneWithMaterialsEnabled:] PictureViewer
104.0ms	4.2%	-[ES2Renderer renderWithTimeStep:] PictureViewer
92.0ms	3.7%	-[ES2Renderer compileShader:type:file:] PictureViewer
88.0ms	3.5%	-[ES2Renderer renderWithTimeStep:] PictureViewer
82.0ms	3.3%	-[ES2Renderer drawSceneWithMaterialsEnabled:] PictureViewer
78.0ms	3.1%	-[ES2Renderer renderWithTimeStep:] PictureViewer
35.0ms	1.4%	-[ES2Renderer drawSceneWithMaterialsEnabled:] PictureViewer
27.0ms	1.0%	-[ES2Renderer createTextureFromFile:allowMipmaps:size
26.0ms	1.0%	-[ES2Renderer createTextureFromFile:allowMipmaps:size
23.0ms	0.9%	-[ES2Renderer createTextureFromFile:allowMipmaps:size
20.0ms	0.8%	-[ES2Renderer createTextureFromFile:allowMipmaps:size
15.0ms	0.6%	-[ES2Renderer createTextureFromFile:allowMipmaps:size
11.0ms	0.4%	-[ES2Renderer createTextureFromFile:allowMipmaps:size


Intro
Time Profiler
Power
Re-
symbolication
System Trace
Conclusion

System Analysis

Motivation

Where your threads
spend time

How your threads interact
with other threads

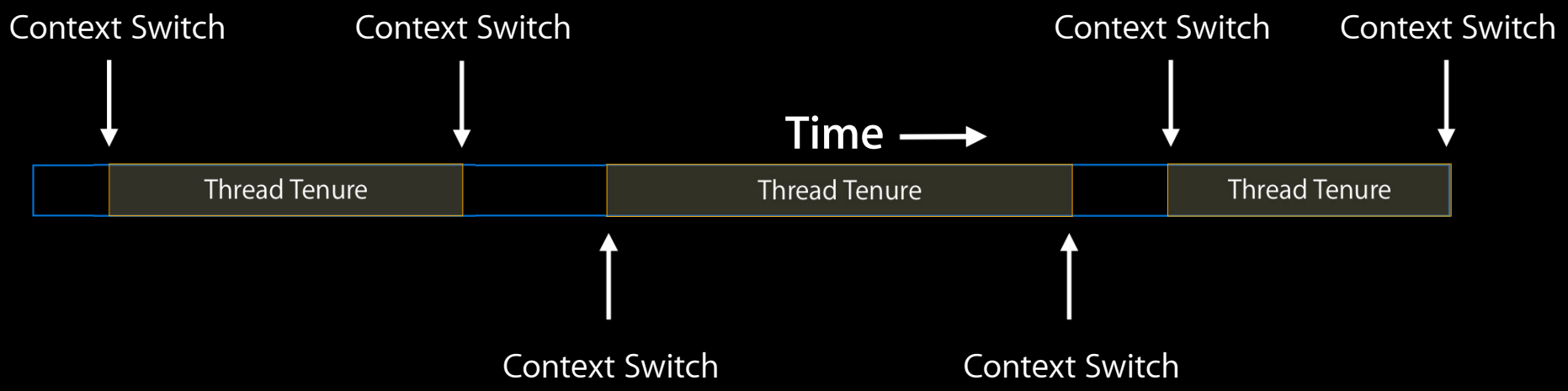
System Trace

What does it do?

- Shows all transitions between user to kernel code due to system events:
  - System calls
  - VM operations
  - Thread scheduling context switches

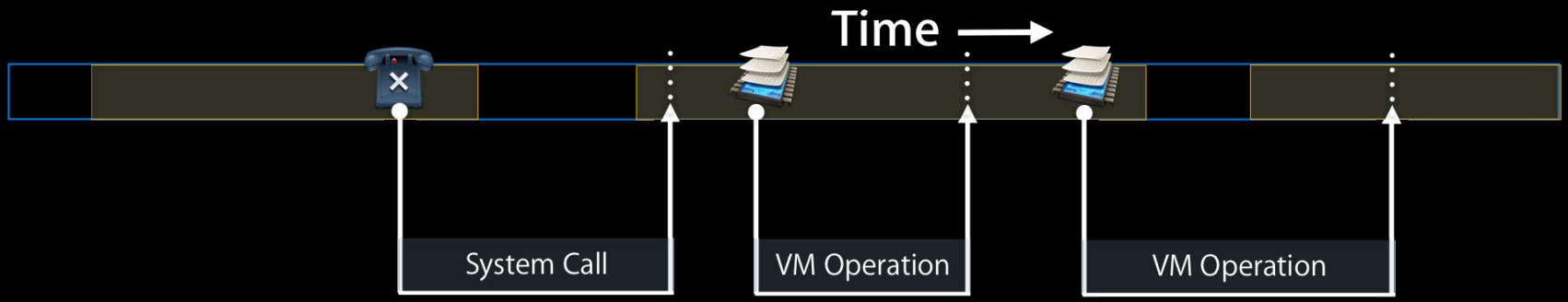
System Trace

How does it work?



System Trace

How does it work?



System Trace

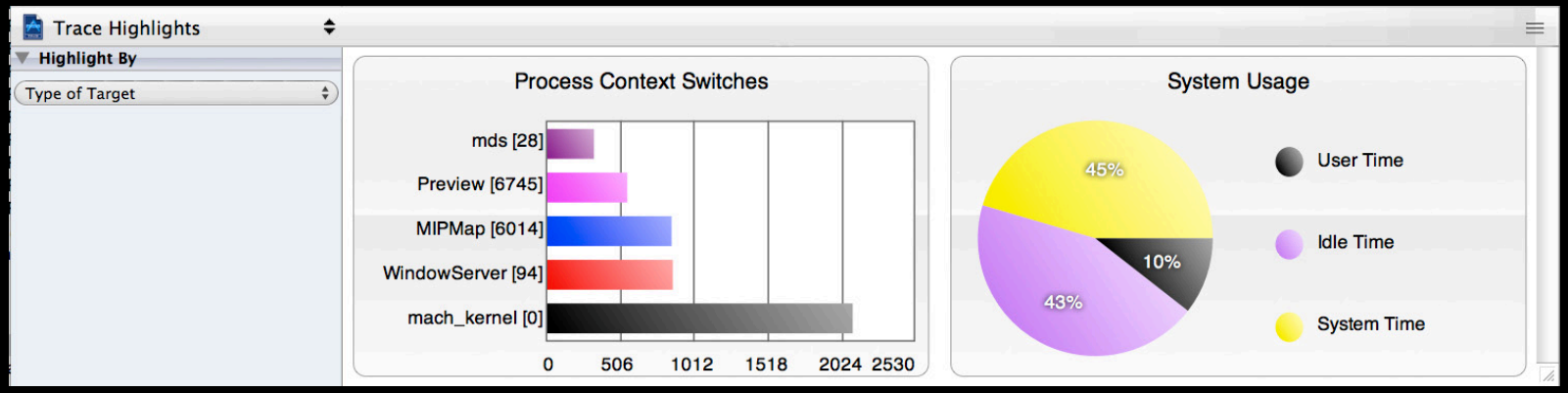
Scheduling Instrument

- Records all thread context switches
- Context switch reasons
- Threads running in user, system or idling
- Thread priorities

System Trace

Scheduling Instrument

- Rules of thumb
  - Minimize number of context switches
  - For throughput: maximize tenure length
  - For responsiveness: minimize tenure length



System Trace

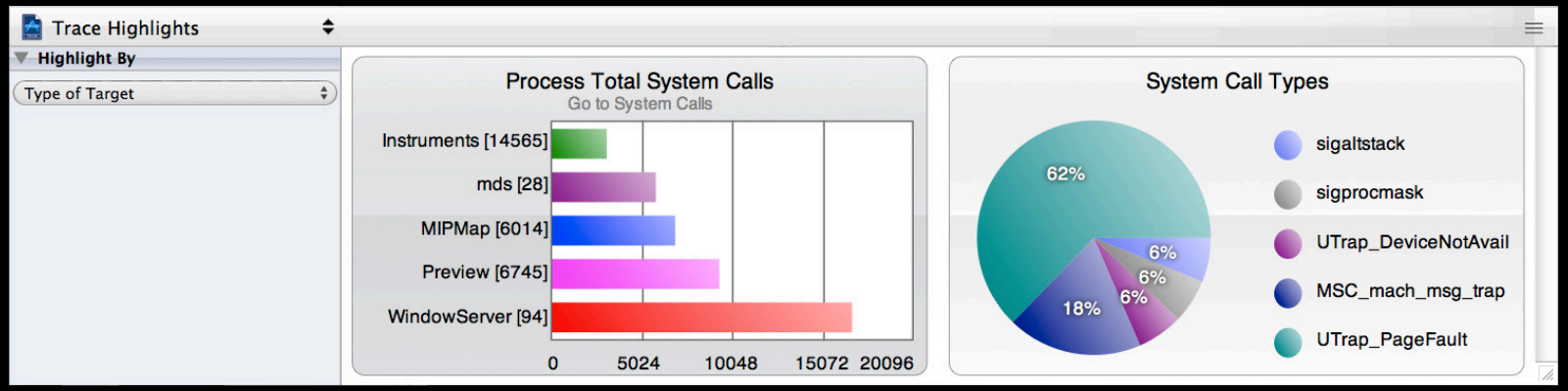
System Calls Instrument

- Records all system calls made to the kernel
  - e.g. read, select, locks
- May incur context switch
- Cause unwanted serialization

System Trace

System Calls Instrument

- Rules of thumb
  - Minimize number of system calls
  - Avoid slow blocking system calls



System Trace

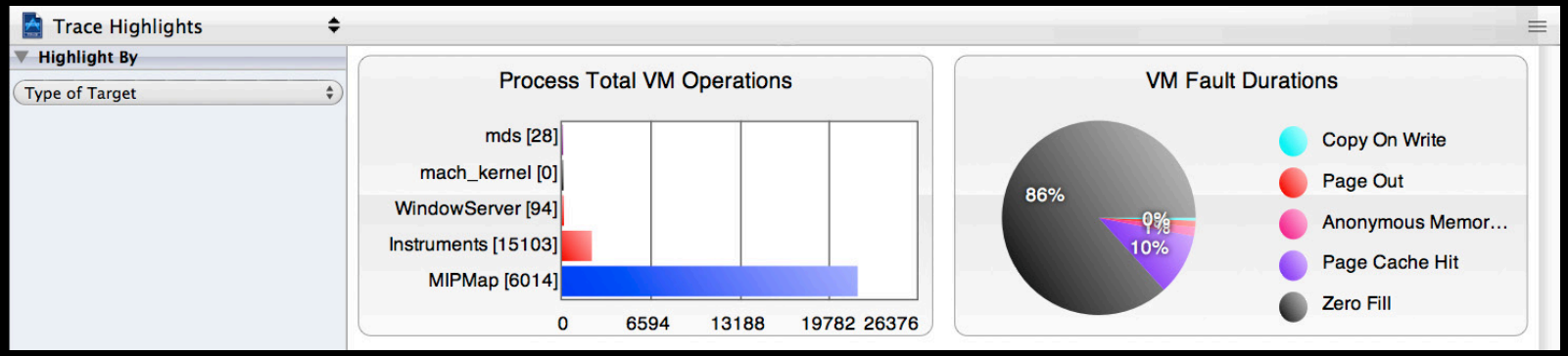
VM Operations Instrument

- Records uses of unmapped memory pages
  - First use of new memory allocations
  - First write to shared memory
  - Page-ins and outs from disk

System Trace

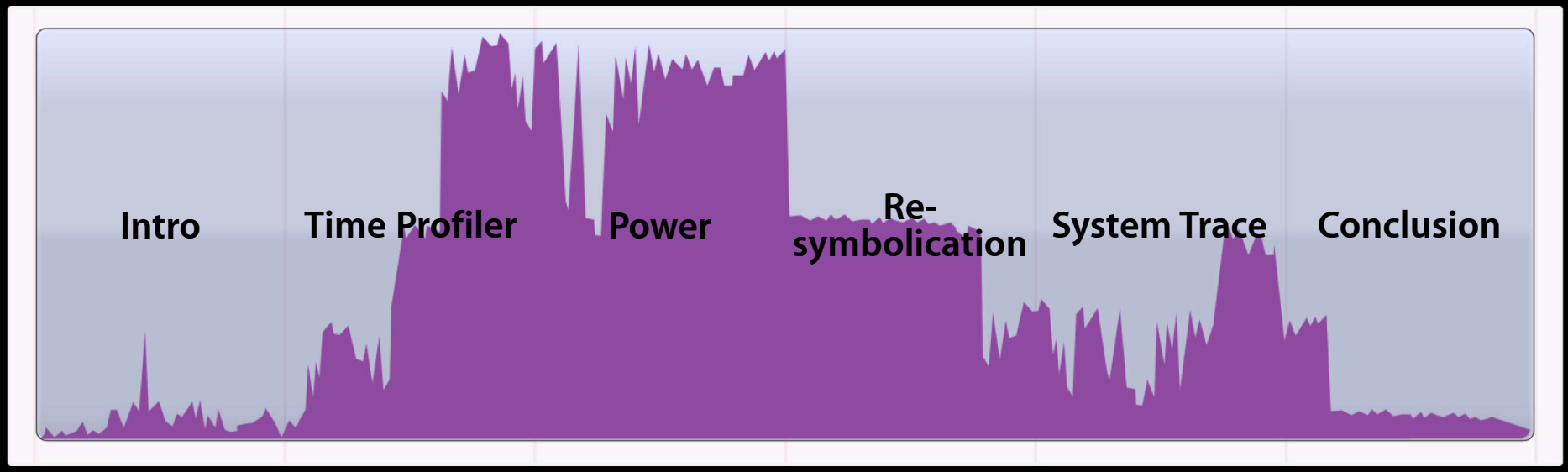
VM Operations Calls Instrument

- Rules of thumb
  - Minimize memory use generally reduces page faults
  - Reuse memory allocations
  - Pre-allocate memory for latency sensitive programs



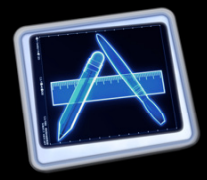
Demo

System Trace


Intro
Time Profiler
Power
Re-
symbolication
System Trace
Conclusion

Conclusion

Performance improvements are iterative



- Progressively optimize using Time Profiler
- Understand power characteristics of iOS 4 apps
- Examine interaction with the system

More Information

Michael Jurewitz
Developer Tools Evangelist
jurewitz@apple.com

Instruments Documentation
Instruments User Guide (Xcode documentation)

Apple Developer Forums
http://devforums.apple.com

Related Sessions

What's New in Instruments	Presidio Wednesday 11:30 AM
Advanced Memory Analysis with Instruments	Presidio Thursday 11:30 AM
Performance Optimization on iPhone OS	Presidio Thursday 2:00 PM
Advanced Performance Optimization on iPhone OS, Part 1	Mission Thursday 3:15 PM
Advanced Performance Optimization on iPhone OS, Part 2	Mission Friday 11:30AM

Related Labs

iPhone OS Performance Lab	Developer Tools Lab A Thursday 4:30PM
iPhone OS Performance Lab	Developer Tools Lab A Friday 9:00AM
Mac OS X Performance Lab	Application Frameworks Lab C Friday 11:30AM


WWDC10