



# Advanced Performance Analysis with Instruments

Lynne Salameh

# Today's Agenda

Intro

Time Profiler

Power

Re-  
symbolication

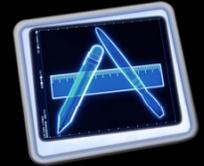
System Trace

Conclusion

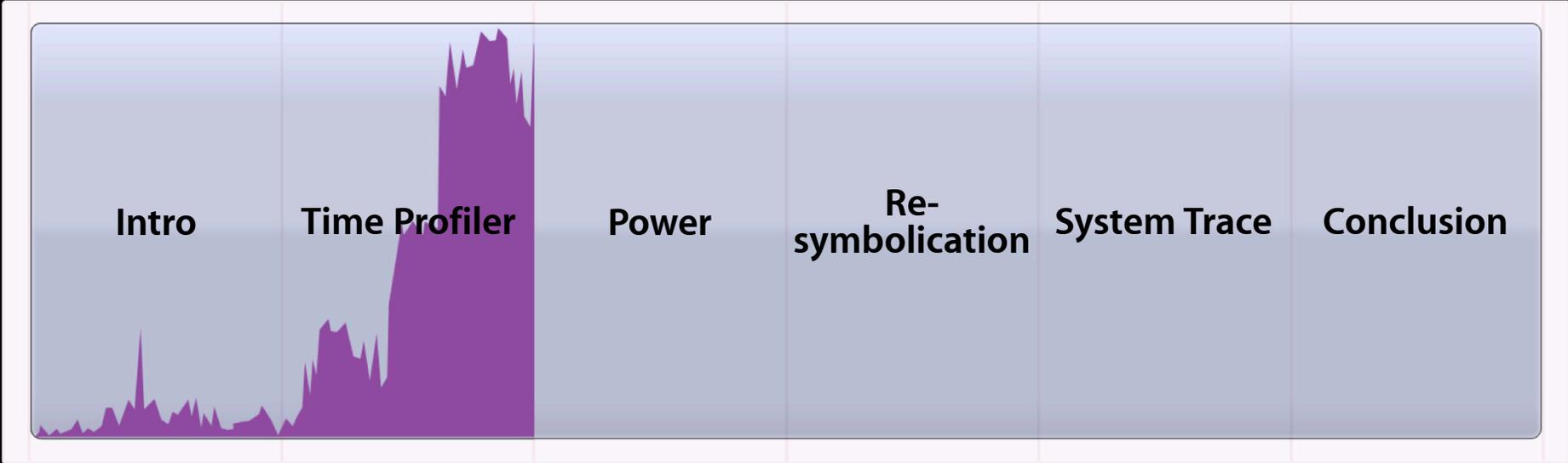


# Introduction

## Advanced performance analysis



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



# 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?

iOS 4

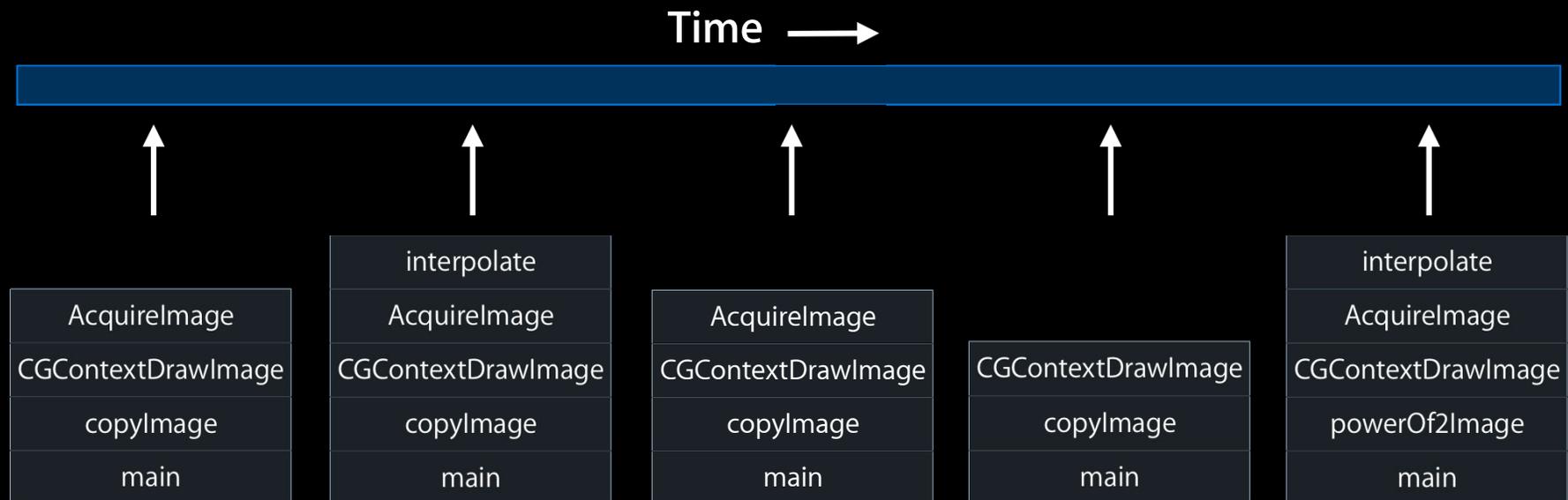


# Time Profiler

## How does it work?



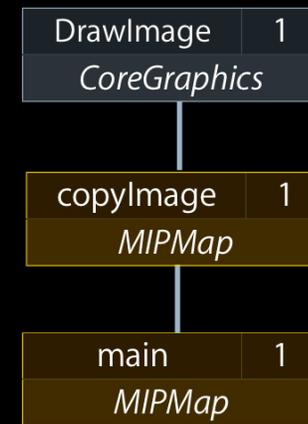
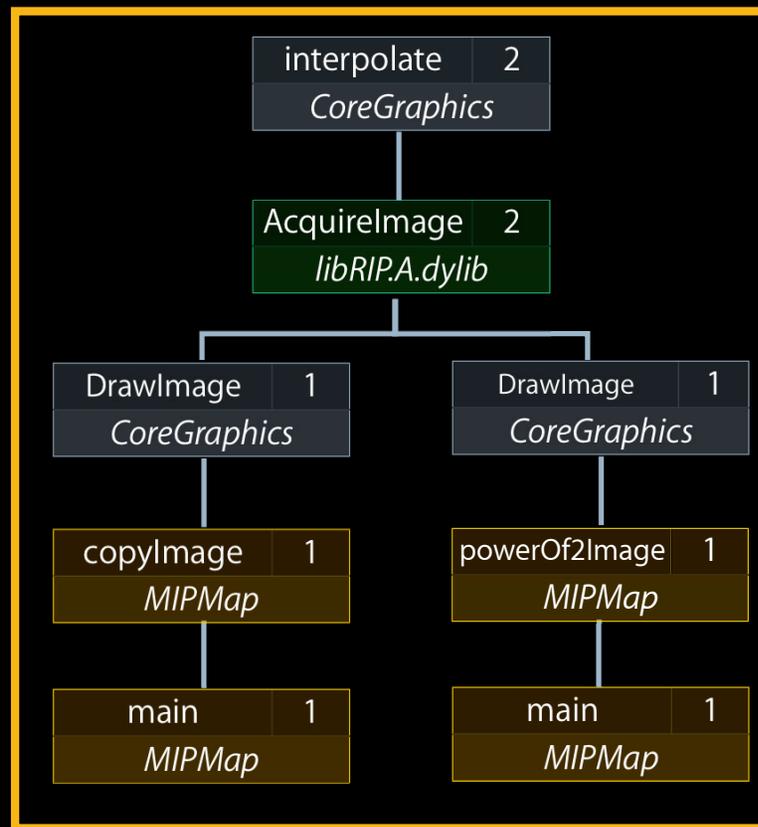
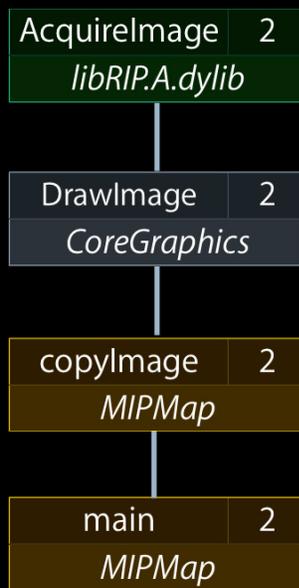
- Periodic samples of threads
- Gathers call stacks
  - Including kernel time and drivers





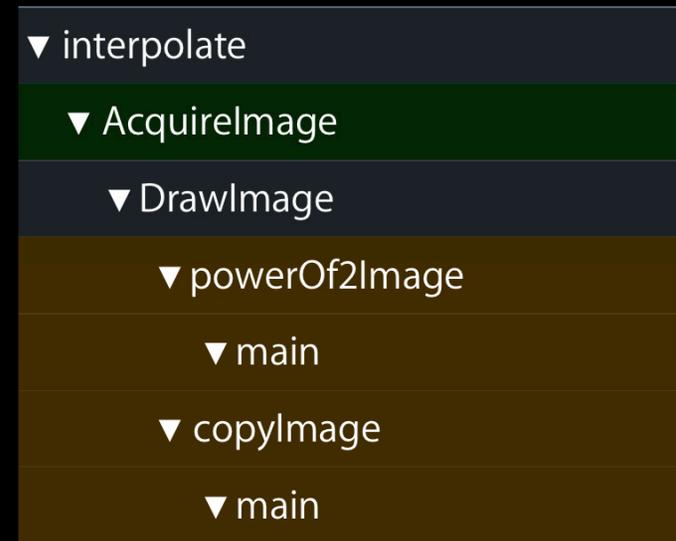
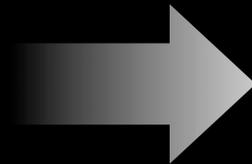
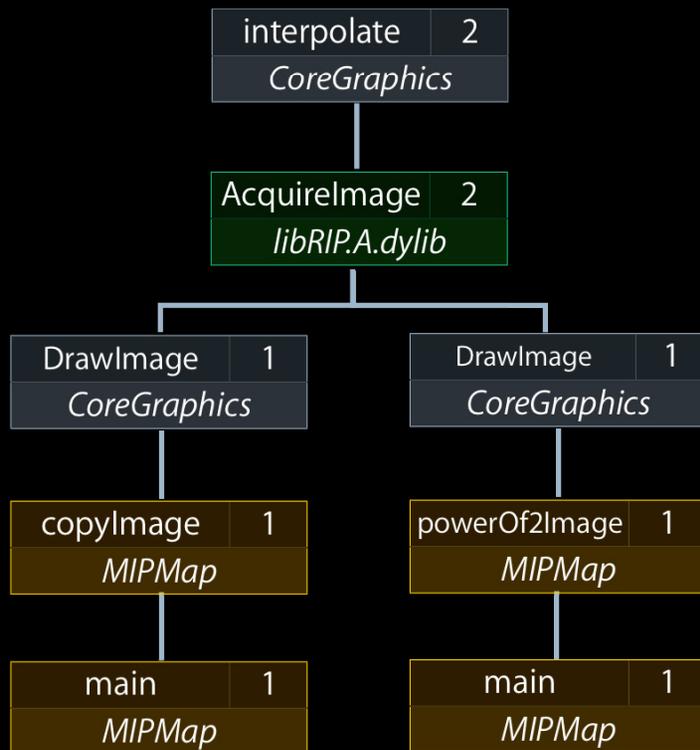
# Time Profiler

## Building call trees



# Time Profiler

## Building call trees



# Time Profiler

## 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

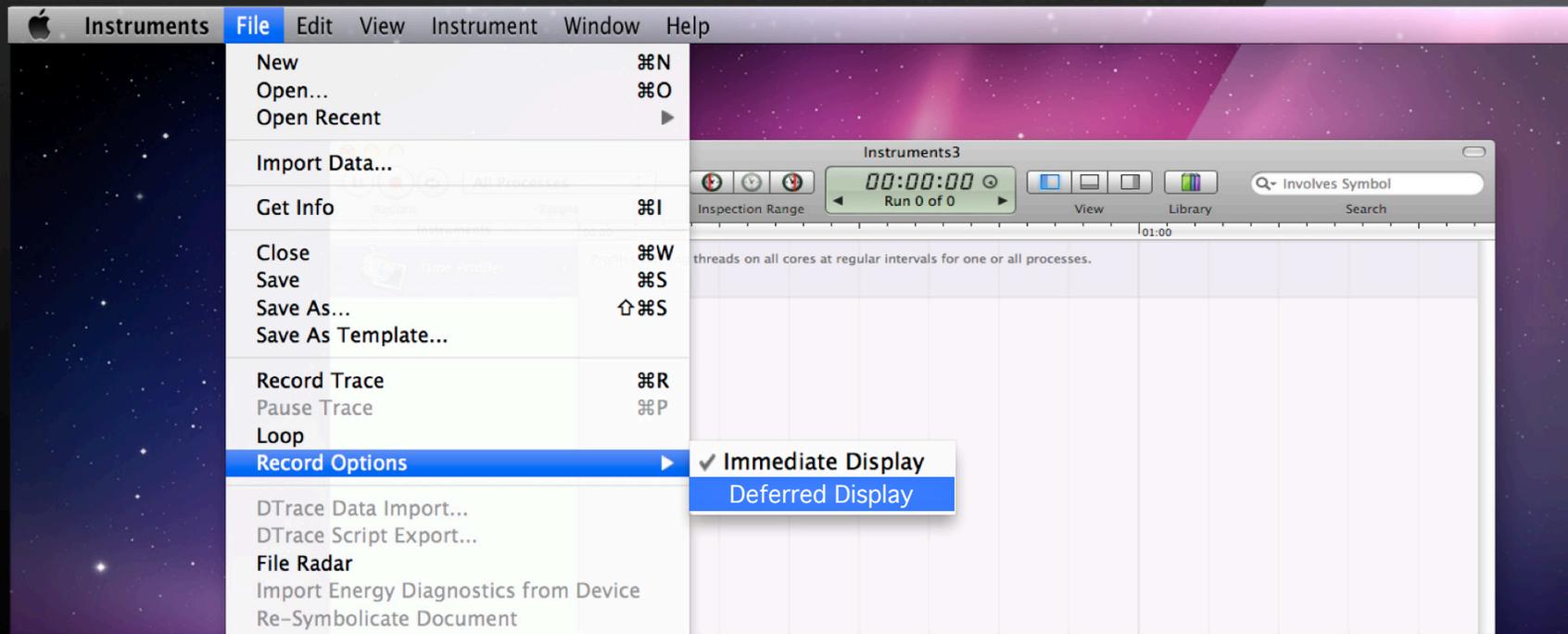


- Deferred mode—new recording style in Instruments
  - Disables track and detail views
  - Defers data processing until end

Minimize Observer Effect

# Time Profiler

## Deferred mode



# Time Profiler

## Deferred mode



The screenshot shows the Instruments application window with the 'File' menu open. The menu items are as follows:

- New ⌘N
- Open... ⌘O
- Open Recent ▶
- Import Data...
- Get Info ⌘I
- Close ⌘W
- Save ⌘S
- Save As... ⇧⌘S
- Save As Template...
- Record Trace ⌘R
- Pause Trace ⌘P
- Loop
- Record Options ▶**
  - Immediate Display
  - ✓ Deferred Display**
- DTrace Data Import...
- DTrace Script Export...
- File Radar
  - Import Energy Diagnostics from Device
  - Re-Symbolicate Document

The background shows the Instruments2 interface with a 'Recording' button and a search bar containing 'Involves Symbol'.

# Time Profiler

## Kernel backtraces



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

Heaviest Stack Trace	
Thread 0xf7d4 : Main Thread	
	1994 main
	1967 -[ImageProcessor start:]
	1967 copyImage
	1967 CGContextDrawImage
	204 CGContextReadGetBytesAtOffset
	204 memcpy

User

Kernel

# 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



Heaviest Stack Trace	
Thread 0xf7d4 : Main Thread	
	1994 main
	1967 -[ImageProcessor start:]
	1967 copyImage
	1967 CGContextDrawImage
	204 CGContextReadGetBytesAtOffset
	204 memcpy
	98 user_trap
	98 vm_fault
	96 vm_fault_page
	94 vnode_pagein
	67 vm_page_grab

# Time Profiler

## Stack compression



Heaviest Stack Trace	
Thread 0xf7d4 : Main Thread	
	1994 main
	1967 -[ImageProcessor start:]
	1967 copyImage
	1967 CGContextDrawImage
	204 CGContextReadGetBytesAtOffset
	204 memcpy
	98 user_trap
	67 vm_page_grab

# Time Profiler

## Stack compression



Heaviest Stack Trace	
Thread 0xf7d4 : Main Thread	
	1994 main
	1967 -[ImageProcessor start:]
	1967 copyImage
	1967 CGContextDrawImage
	67 vm_page_grab

# 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 Call Tree Samples XRP\_network\_output main.c

Line	Code	Percentage
105	for (h = 0; h < XRP_HIDDEN_NODE_COUNT; h++)	1.9%
106	{	
107	double input_sum = net->hidden_biases[h];	1.7%
108	for (i = 0; i < XRP_INPUT_NODE_COUNT; i++)	5.4%
109	{	
110	const double weight =	
111	net->weight_hidden_to_input[h][i];	7.2%
112		
113	input_sum += inputs[i] * weight;	22.0%
114	}	
115		
116	scratch->net_input_to_hidden[h] = input_sum;	
117		
118	scratch->output_of_hidden[h] = tanh(input_sum);	
119	}	
120		
121	for (o = 0; o < XRP_OUTPUT_NODE_COUNT; o++)	
122	{	
123	double net_input_to_output = net->output_biases[o];	
124		
125	for (h = 0; h < XRP_HIDDEN_NODE_COUNT; h++)	
126	{	
127	const double weight =	
128	net->weight_output_to_hidden[o][h];	2.3%
129		
130	net_input_to_output +=	3.7%
131	scratch->output_of_hidden[h] * weight;	

**Heaviest Backtraces**

1 Hit

- start
- XRP\_network\_output

**Instructions**

```
mov ebx, dword ptr [rbp-24]
movsd xmm0, [rbp-40]
call 0x00000000000000c0
mov rdx, qword ptr [rbp-96]
movsxd rax, rbx
movsd [word ptr [rdx+rax*8], xmm0
```

# Time Profiler

## Disassembly view



Time Profiler    Call Tree    Samples    XRP\_network\_output

main.c

Line	Code	Percentage
105	for (h = 0; h < XRP_HIDDEN_NODE_COUNT; h++)	1.9%
106	{	
107	double input_sum = net->hidden_biases[h];	1.7%
108	for (i = 0 ; i < XRP_INPUT_NODE_COUNT ; i++)	5.4%
109	{	
110	const double weight =	
111	net->weight_hidden_to_input[h][i];	7.2%
112		
113	input_sum += inputs[i] * weight;	22.0%
114	}	
115		
116	scratch->net_input_to_hidden[h] = input_sum;	2.3%
117		
118	scratch->output_of_hidden[h] = tanh(input_sum);	47.7%
119	}	
120		
121	for (o = 0 ; o < XRP_OUTPUT_NODE_COUNT ; o++)	0.7%
122	{	
123	double net_input_to_output = net->output_biases[o];	0.4%
124		
125	for (h = 0; h < XRP_HIDDEN_NODE_COUNT; h++)	1.7%
126	{	
127	const double weight =	
128	net->weight_output_to_hidden[o][h];	2.3%
129		
130	net_input_to_output +=	3.7%
131	scratch->output_of_hidden[h] * weight;	

# Time Profiler

## Disassembly view



Time Profiler Call Tree Samples XRP\_network\_output

NeuralExp

Address	Instruction	Operand	Percentage
41	+0x82	mulsd xmm1, [rbp-48]	0.9%
42	+0x87	movsd xmm0, [rbp-40]	4.5%
43	+0x8c	addsd xmm0, xmm1	0.0%
44	+0x90	movsd [rbp-40], xmm0	6.5%
45	+0x95	inc qword ptr [rbp-20]	2.0%
46	+0x98	cmp byte ptr [rbp-20], byte 2	1.5%
47	+0x9c	jle XRP_network_output+0x49	1.9%
48	+0x9e	mov eax, dword ptr [rbp-24]	
49	+0xa1	mov rcx, qword ptr [rbp-96]	
50	+0xa5	movsxd rdx, rax	
51	+0xa8	mov rax, qword ptr [rbp-40]	0.9%
52	+0xac	mov qword ptr [rcx+rdx*8+8], rax	1.4%
53	+0xb1	mov ebx, dword ptr [rbp-24]	0.8%
54	+0xb4	movsd xmm0, [rbp-40]	
55	+0xb9	call dyld_stub_tanh	46.2%
56	+0xbe	mov rdx, qword ptr [rbp-96]	0.8%
57	+0xc2	movsxd rax, rbx	
58	+0xc5	movsd [word ptr [rdx+rax*8], xmm0	
59	+0xca	inc qword ptr [rbp-24]	0.8%
60	+0xcd	cmp byte ptr [rbp-24], byte 2	0.2%
61	+0xd1	jle XRP_network_output+0x2b	0.7%
62	+0xd7	mov dword ptr [rbp-28], dword 0	
63	+0xde	jmp XRP_network_output+0x184	
64	+0xe3	mov eax, dword ptr [rbp-28]	0.2%
65	+0xe6	mov rdx, qword ptr [rbp-72]	
66	+0xea	cwde	
67	+0xec	mov rax, qword ptr [rdx+rax*8+240]	
68	+0xf4	mov qword ptr [rbp-56], rax	0.2%
69	+0xf8	mov dword ptr [rbp-24], dword 0	
70	+0xff	jmp XRP_network_output+0x151	

# Time Profiler

## Disassembly view



Time Profiler | Call Tree | Samples | XRP\_network\_output

NeuralExp

Address	Instruction	Operand	Percentage
41	+0x82	mulsd xmm1, [rbp-48]	0.9
42	+0x87	movsd xmm0, [rbp-40]	4.5
43	+0x8c	addsd xmm0, xmm1	0.0
44	+0x90	movsd [rbp-40], xmm0	6.5
45	+0x95	inc qword ptr [rbp-20]	2.0
46	+0x98	cmp byte ptr [rbp-20], byte 2	1.5
47	+0x9c	jle XRP_network_output+0x49	1.9
48	+0x9e	mov eax, dword ptr [rbp-24]	
49	+0xa1	mov rcx, qword ptr [rbp-96]	
50	+0xa5	movsxd rdx, rax	
51	+0xa8	mov rax, qword ptr [rbp-40]	0.9
52	+0xac	mov qword ptr [rcx+rdx*8+8], rax	1.4
53	+0xb1	mov ebx, dword ptr [rbp-24]	0.8%
54	+0xb4	movsd xmm0, [rbp-40]	
55	+0xb9	call dyld_stub_tanh	46.2%
56	+0xbe	mov rdx, qword ptr [rbp-96]	0.8%
57	+0xc2	movsxd rax, rbx	
58	+0xc5	movsd [word ptr [rdx+rax*8], xmm0	
59	+0xca	inc qword ptr [rbp-24]	0.8%
60	+0xcd	cmp byte ptr [rbp-24], byte 2	0.2%
61	+0xd1	jle XRP_network_output+0x2b	0.7%
62	+0xd7	mov dword ptr [rbp-28], dword 0	
63	+0xde	jmp XRP_network_output+0x184	
64	+0xe3	mov eax, dword ptr [rbp-28]	0.2%
65	+0xe6	mov rdx, qword ptr [rbp-72]	
66	+0xea	cwde	
67	+0xec	mov rax, qword ptr [rdx+rax*8+240]	
68	+0xf4	mov qword ptr [rbp-56], rax	0.2%
69	+0xf8	mov dword ptr [rbp-24], dword 0	
70	+0xff	jmp XRP_network_output+0x151	

- View as Percentage
- View as Value
- View as Default
- Refresh
- Constrain Call Tree Selection
- Line Correction
- Show ISA reference guide

# Time Profiler

## Disassembly view



EM64T ISA Reference

Instructions

- movapd
- andpd
- movd**
- movapd
- subq
- xorpd
- cmpq
- ja
- movapd
- mulsd
- cvtsd2si
- addq
- movq
- sarq
- andq
- shlq
- movapd
- mulsd
- cvtsi2sd
- movapd
- subsd
- movapd
- movsd
- mulsd
- movsd

### MOVD/MOVQ—Move Doubleword

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

**Flags Affected**  
None.

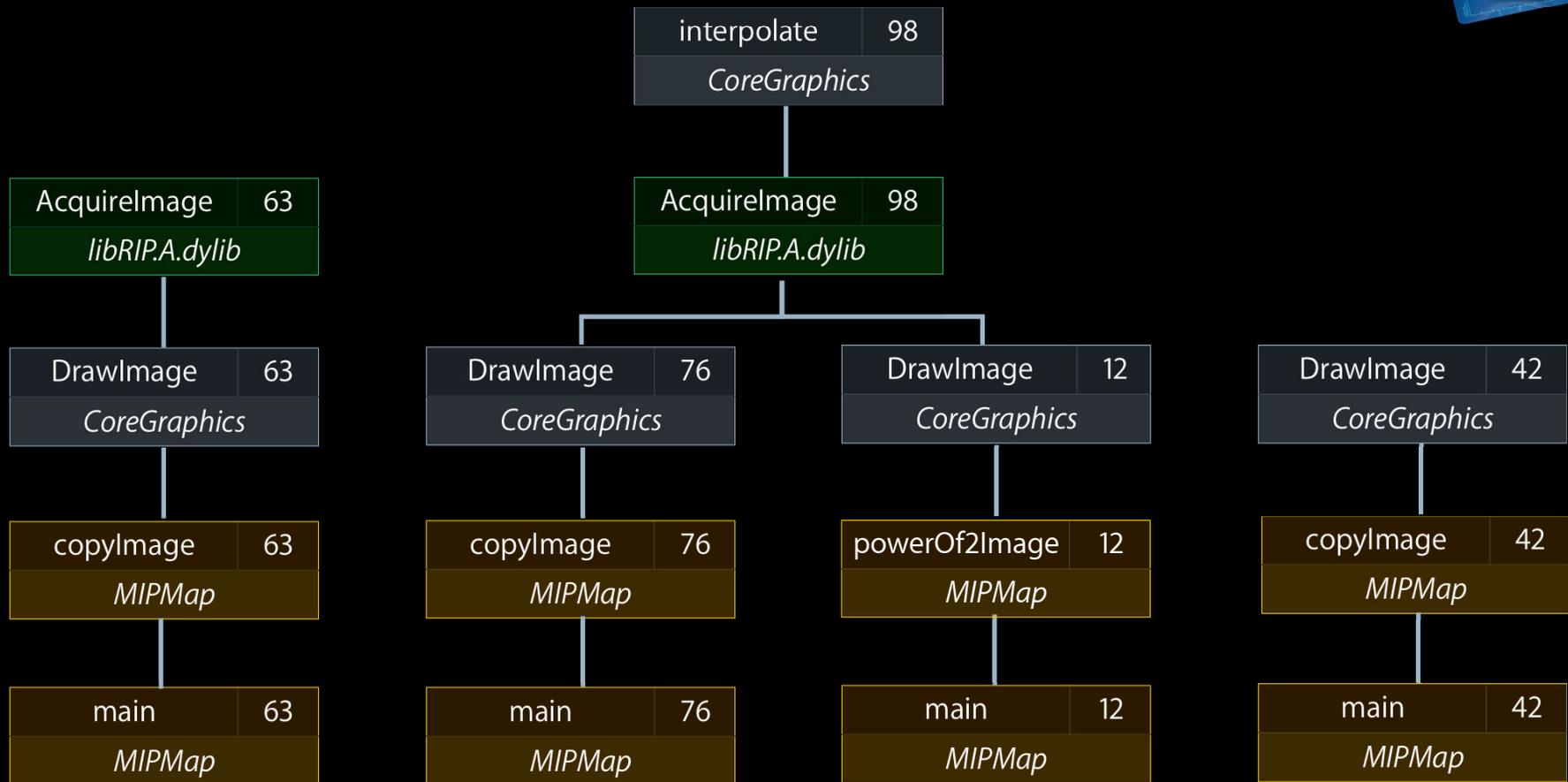
**IA-32e Mode Operation**  
Promoted to 64-bits.  
Default operand size 32-bits.  
Enables access to new registers R8-R15.

movd Page 325 of 601 copyright

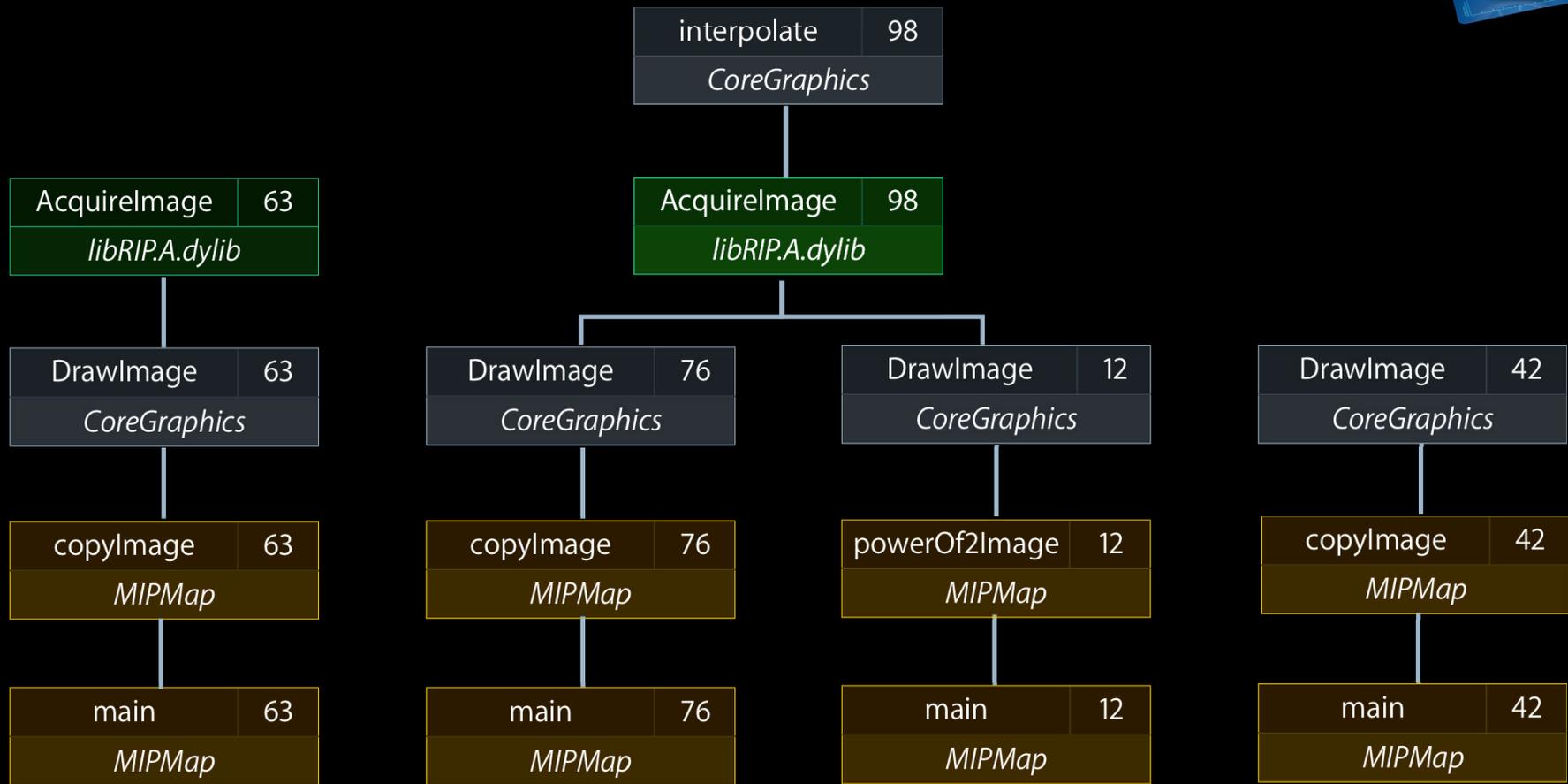
# Demo

## Time Profiler

# Data Mining

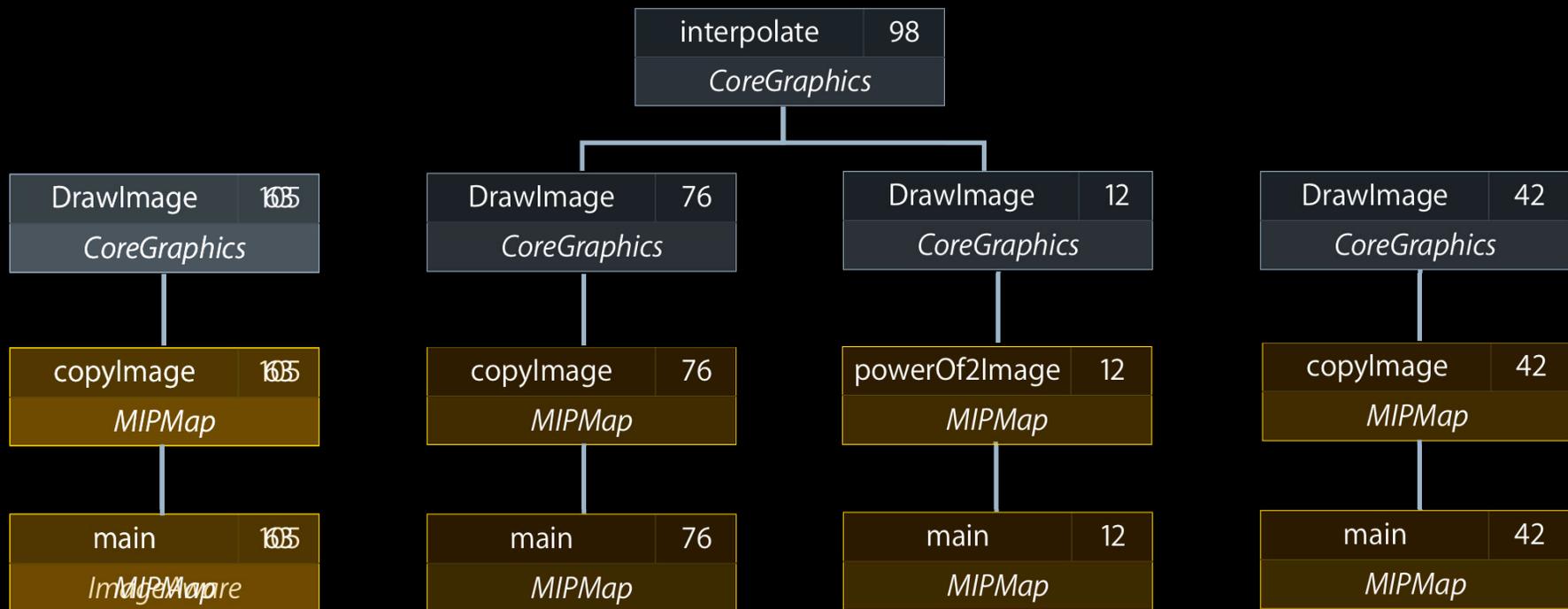


# Data Mining



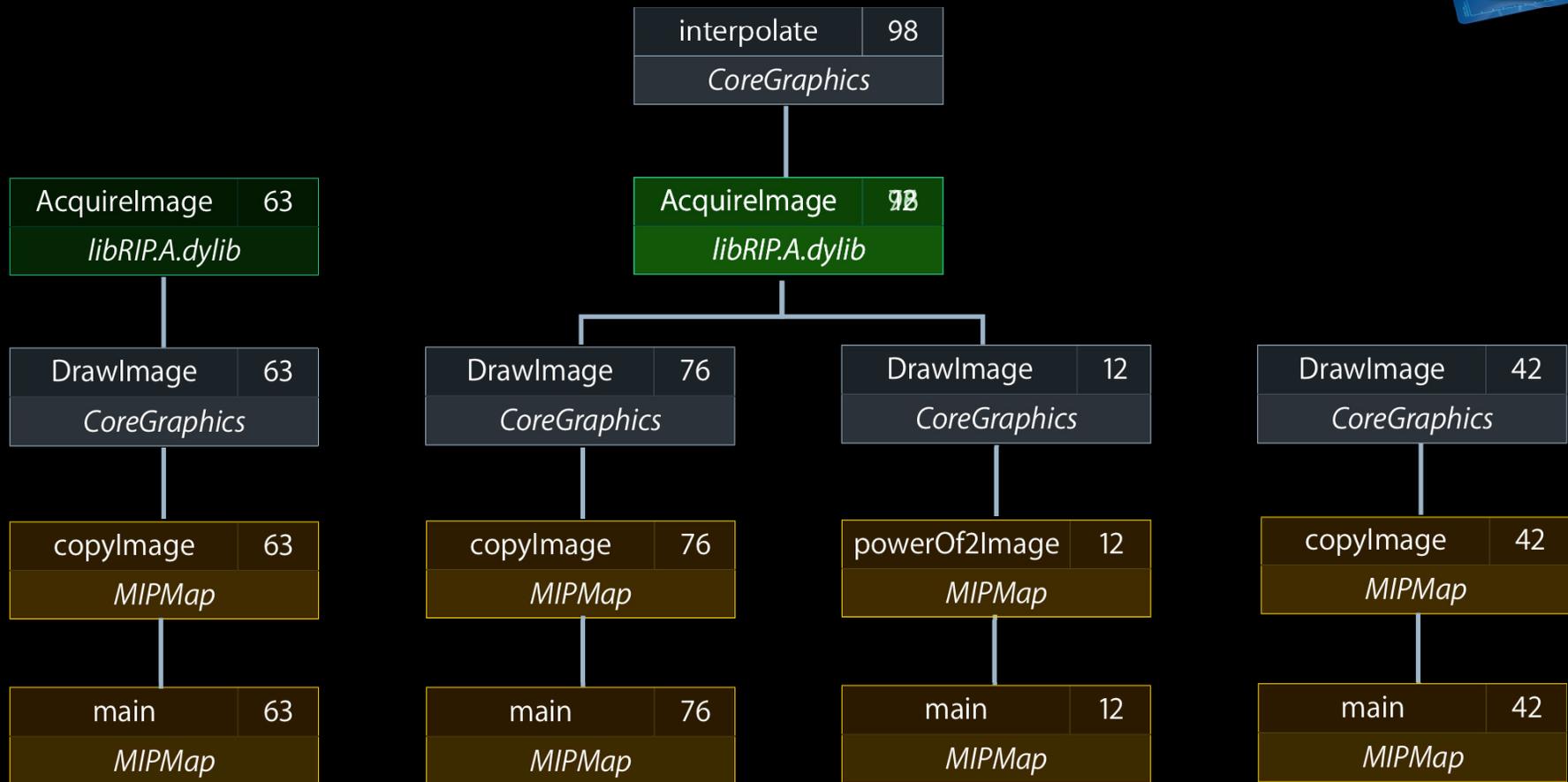
Charge symbol to caller

# Data Mining



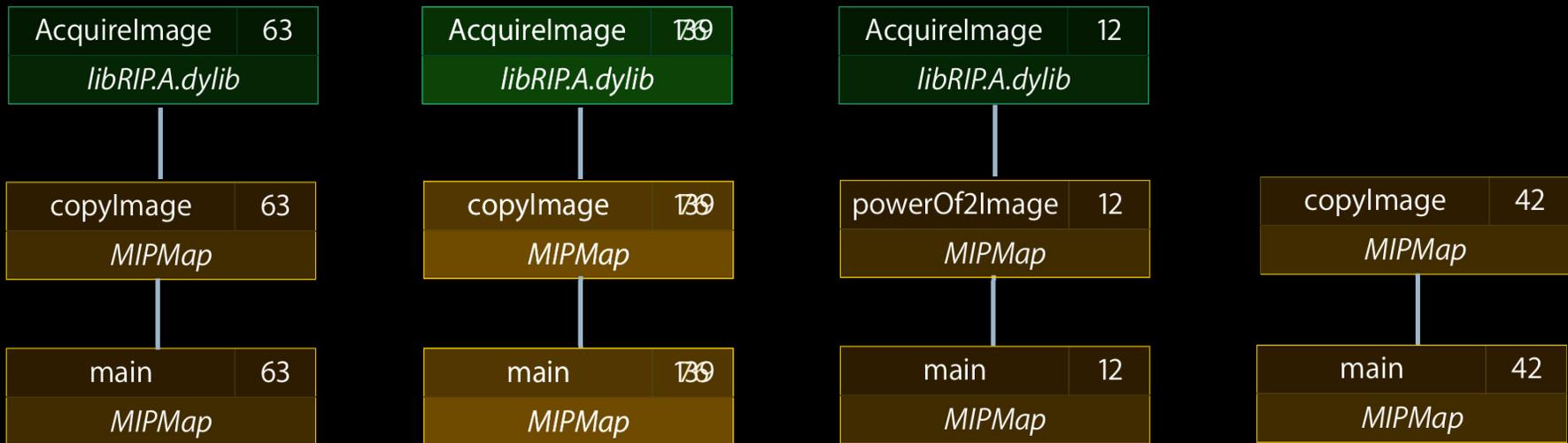
Charge symbol to caller

# Data Mining



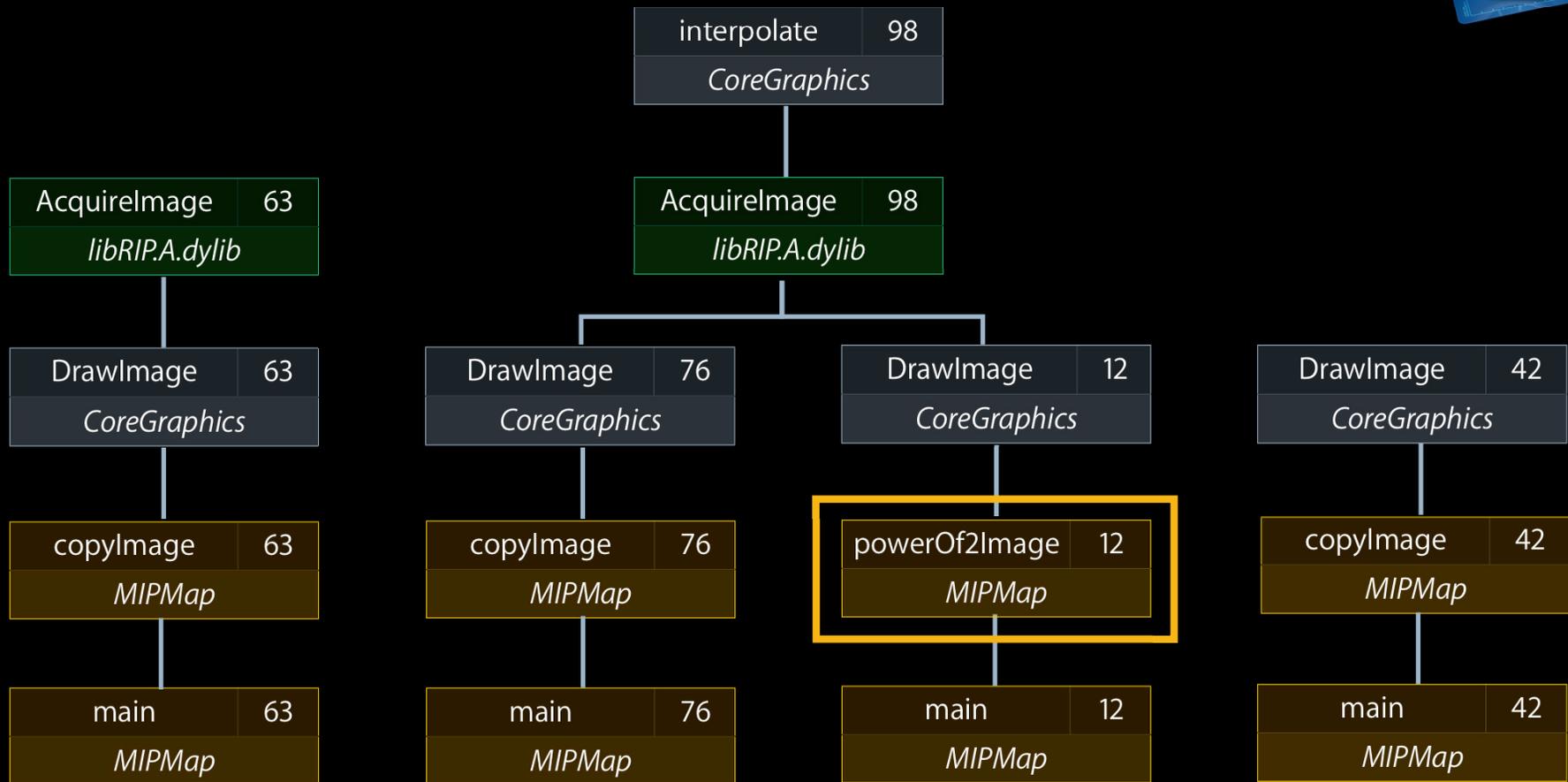
Charge library to caller

# Data Mining



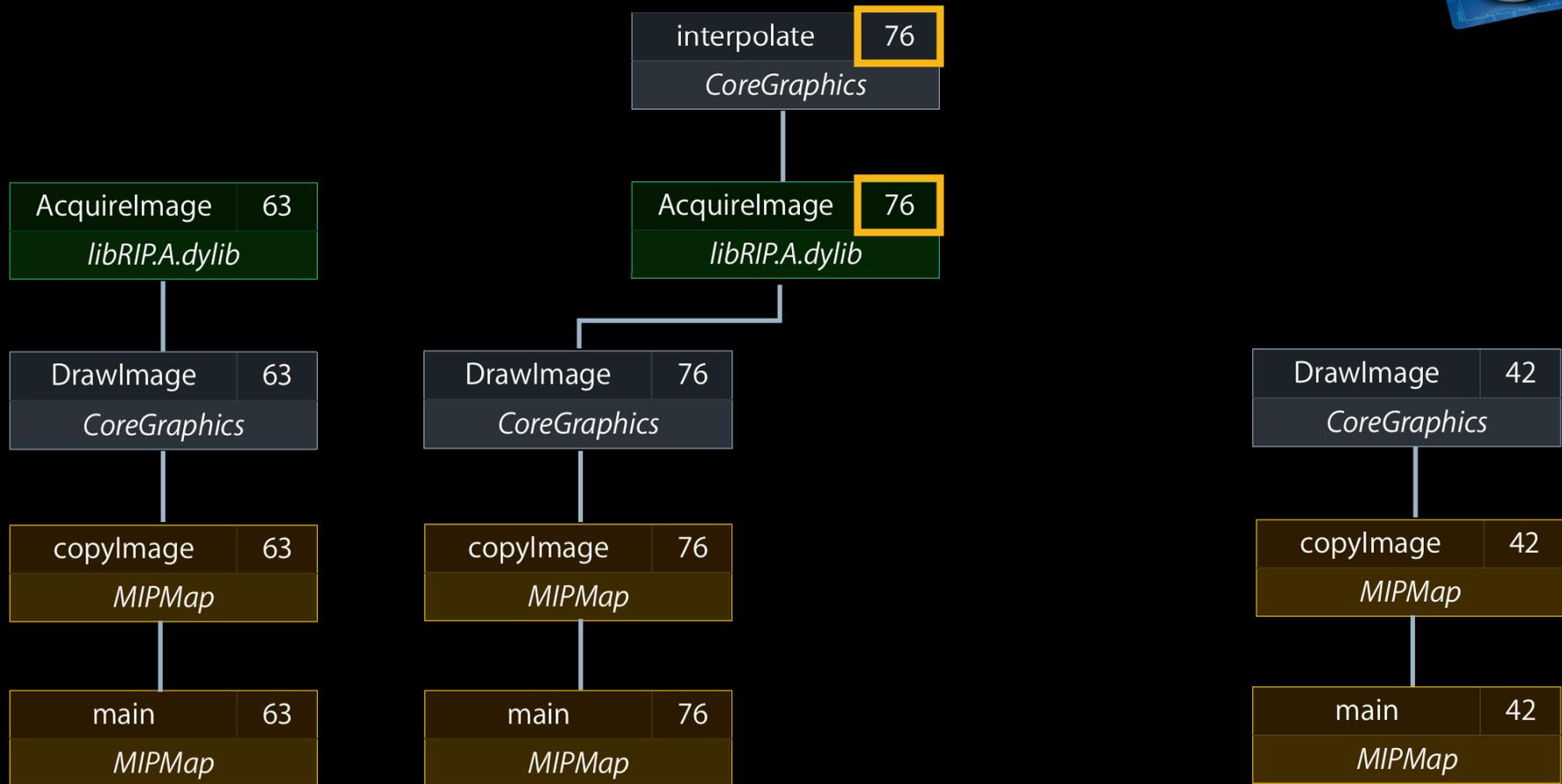
Charge library to caller

# Data Mining



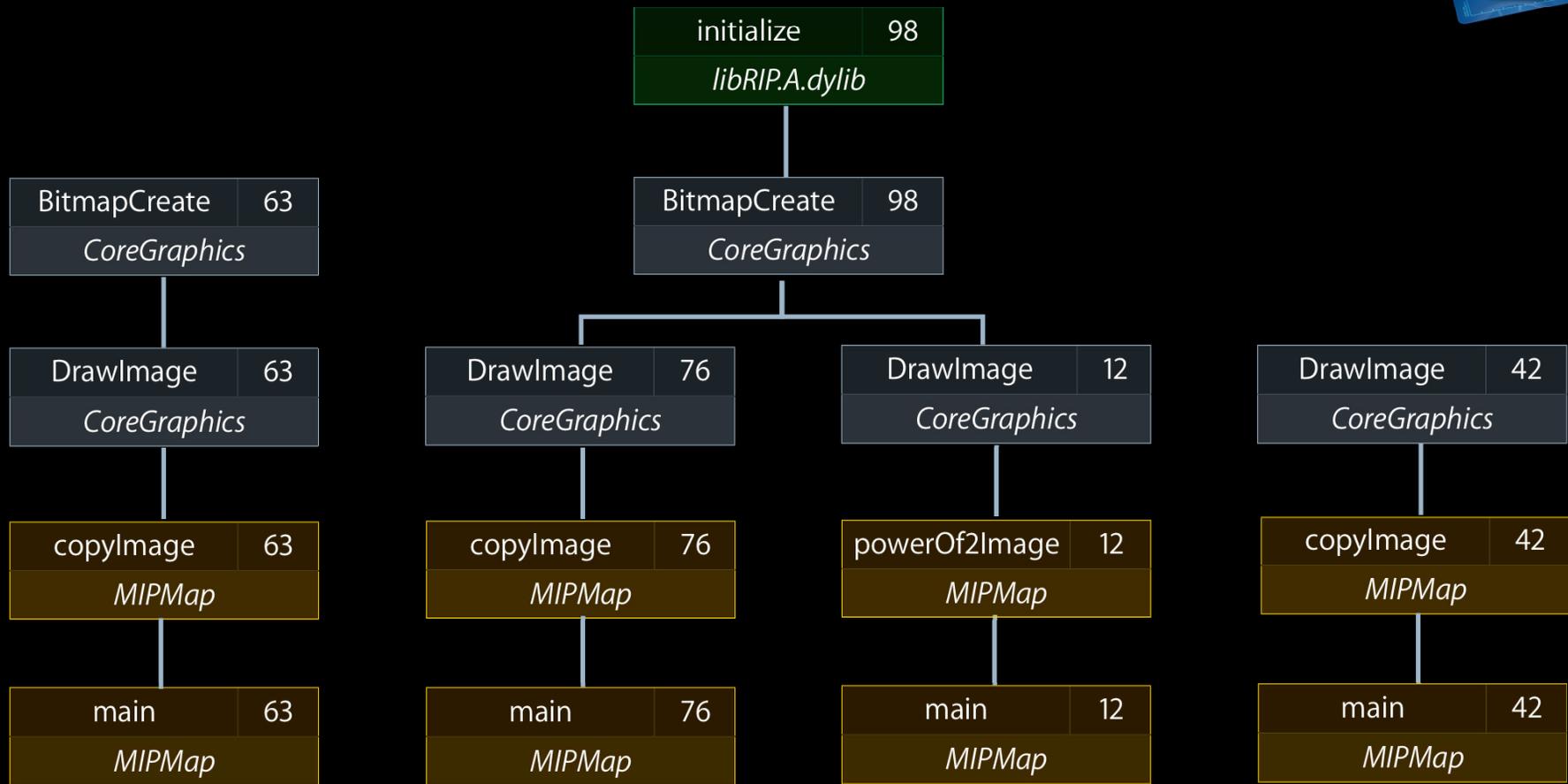
Prune symbol and subtree

# Data Mining



Prune symbol and subtree

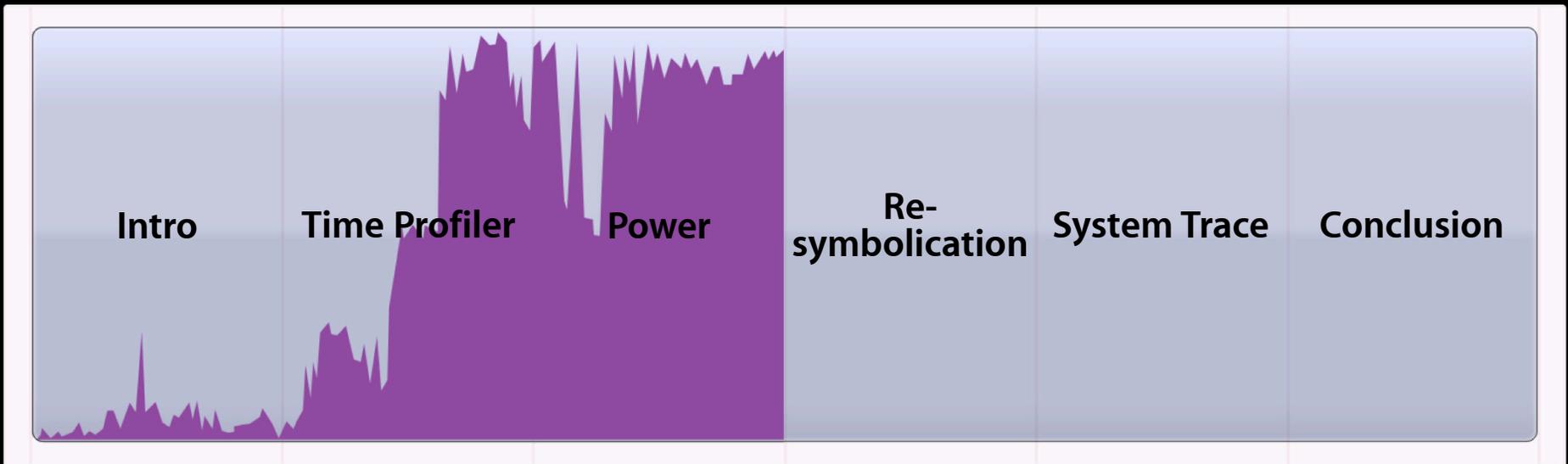
# Data Mining



Flatten library to boundaries

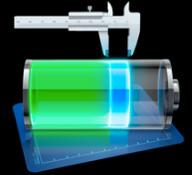
# Advanced Performance Analysis with Instruments

Chad Woolf

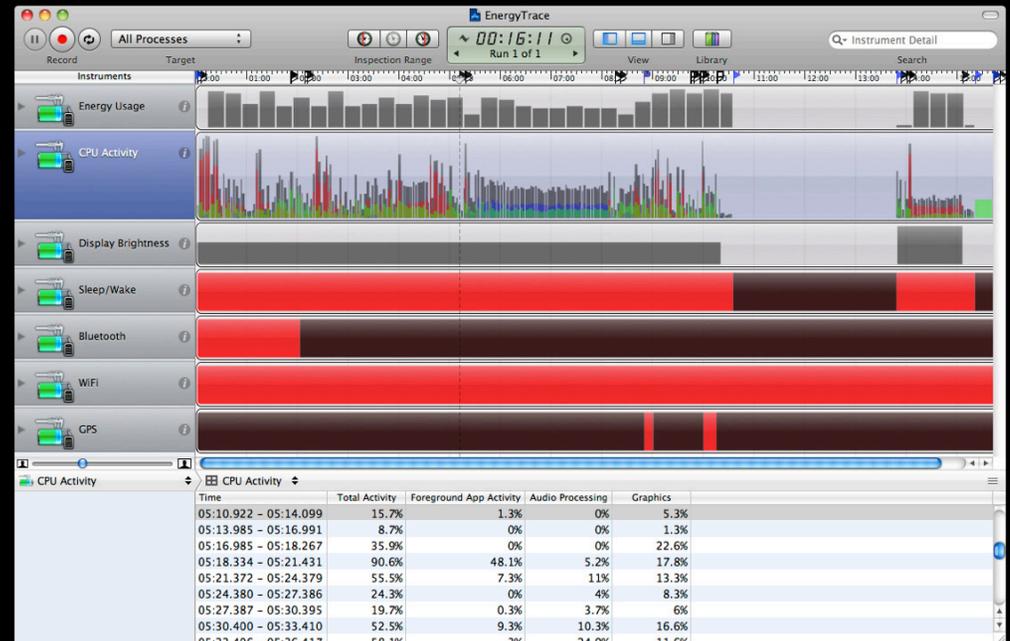


# iPhone SDK 4

## Power analysis tools



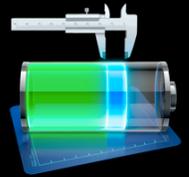
- Energy usage (power)
- CPU activity
- Sleep/Wake
- Display brightness
- Radios
- Energy diagnostics template



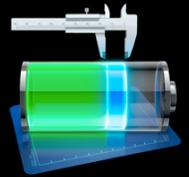
# Tricks of the Trade

## 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

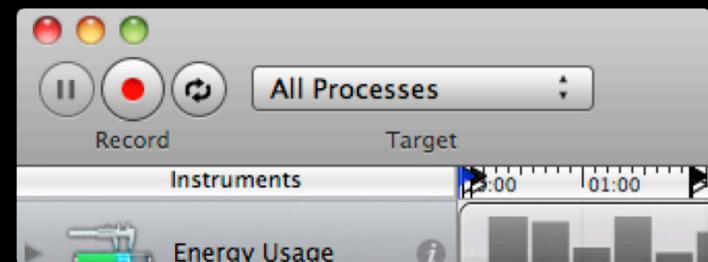


Record in the field



or

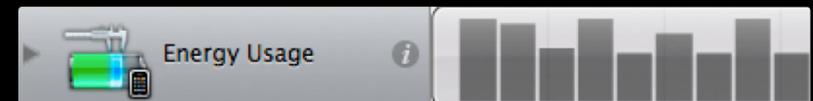
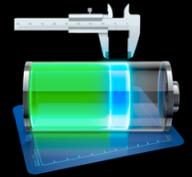
Record while connected



# Energy Usage

## True power measurement

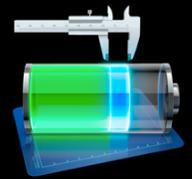
- Records energy flowing from the battery
- Energy usage factor from 1 to 20
  - low number → long battery life
  - high number → short battery life
- Works with
  - iPhone 3GS
  - iPhone 4
  - iPod Touch second and third generation



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



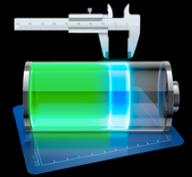
- CPU activity is a big part of your energy profile
- Power scales with CPU load
  - More load = more power
- Measurements
  - Total activity
  - Foreground app activity
  - Graphics/render (springboard)
  - Audio/video (mediaserverd)



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

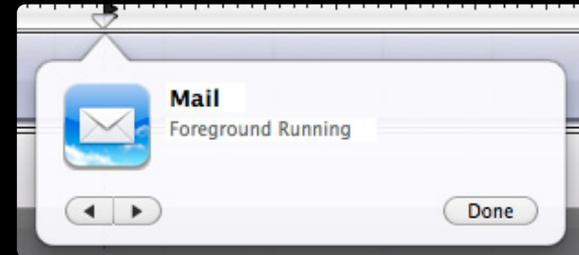
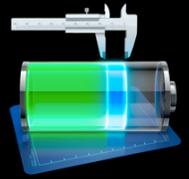


- Graphics/render service
  - CoreAnimation
- Audio/video service
  - AV Foundation
  - Media Player Framework
  - OpenAL

# CPU Activity

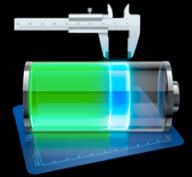
## Tracking important system activity

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



# Radio Instrumentation

## Track radio states



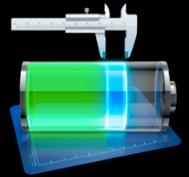
- Bluetooth
  - Enabled/disabled
- Wi-Fi
  - Enabled
    - More efficient location
    - More efficient data transfer
  - Disabled
    - Data transfers must use 3G or EDGE



# 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

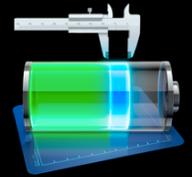
- Turning on logging
- Importing a trace
- Finding interesting events like
  - A wake caused by a push notification
  - Watching a video on YouTube
  - GPS activity

# Demo

Power Instrumentation

# Tricks of the Trade

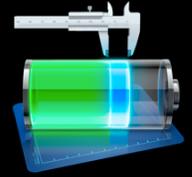
## Controlling your environment



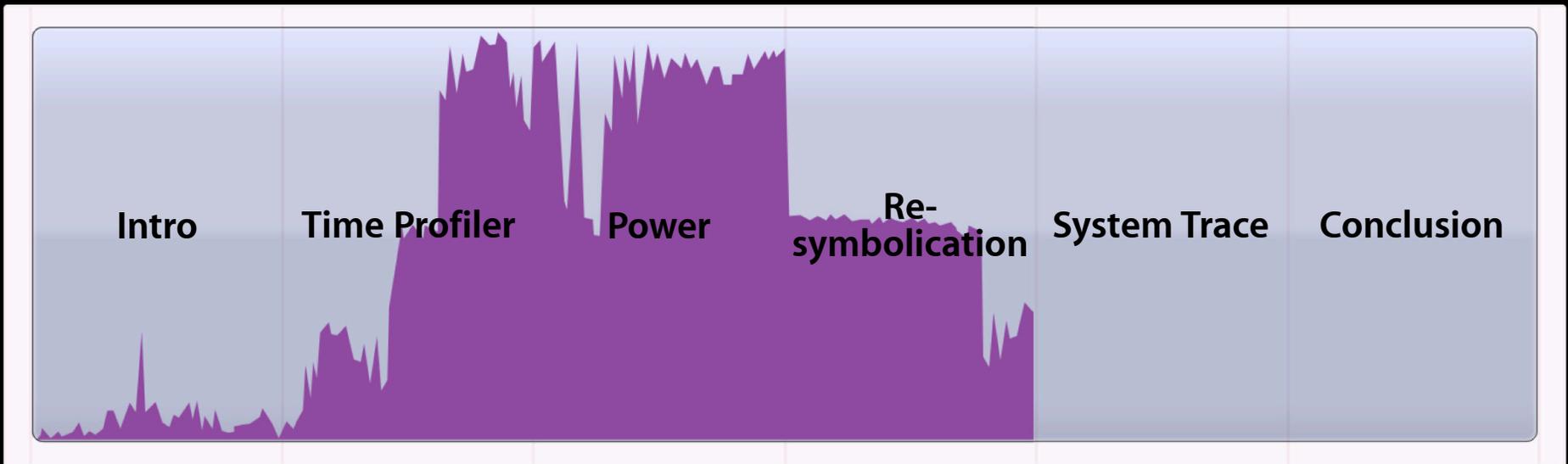
- Controlling iOS 4 features
  - Mail fetching (disable)
  - Push notifications (disable)
  - Auto-dimming (disable)
- Radios
  - Disable Wi-Fi to test 3G or EDGE transfers
  - Disable Bluetooth
- Temperature affects energy usage

# Power Analysis Tools

## Summary



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



# 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

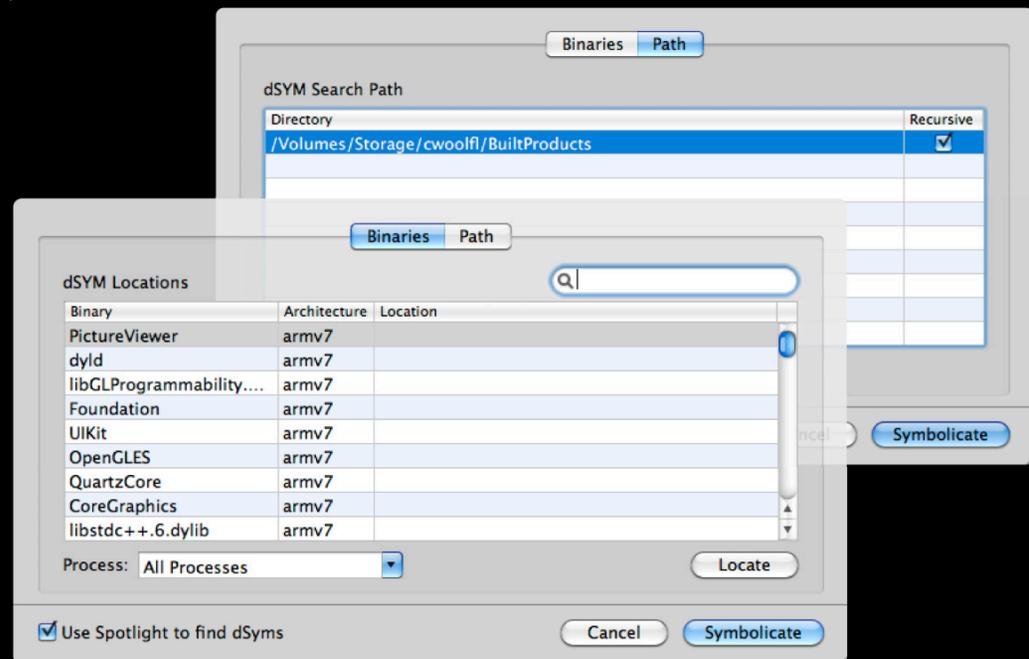
## Before iPhone SDK 4

- 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

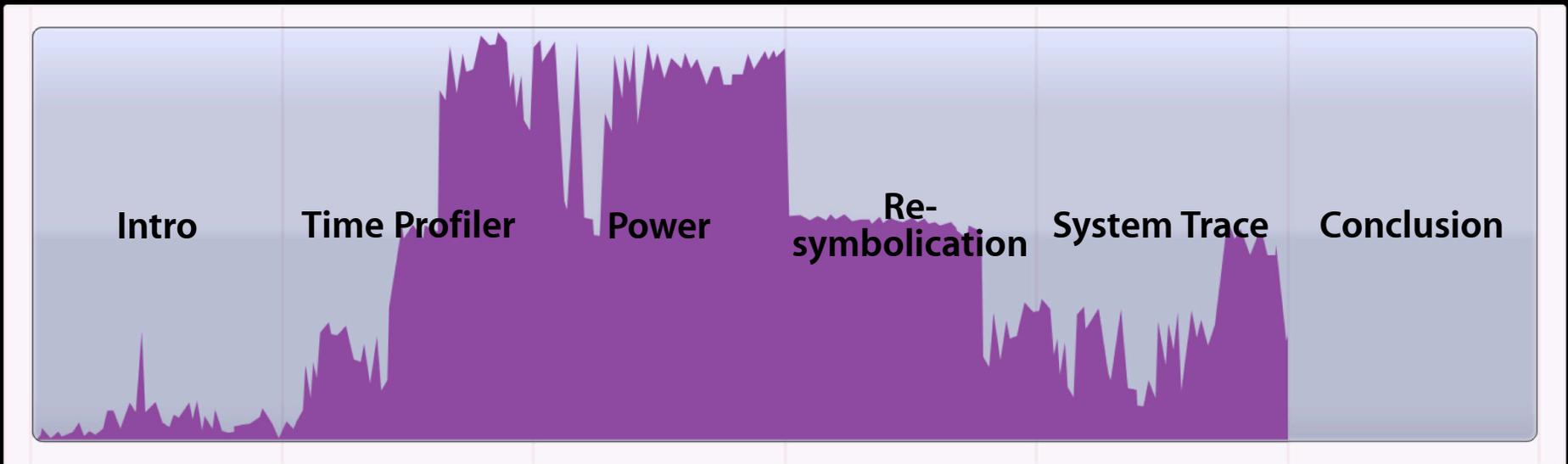
- Manually locate the missing symbol information
- Under the File menu
  - Re-Symbolicate Document
- Locate the dSYM
- Add dSYM search paths



# Re-Symbolicated

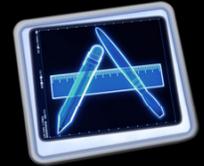
iOS 4

Running Time		Symbol Name
143.0ms	3.8%	-[ES2Renderer renderWithTimeStep:] PictureViewer
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:] PictureViewer
26.0ms	1.0%	-[ES2Renderer createTextureFromFile:allowMipmaps:size:] PictureViewer
23.0ms	0.9%	-[ES2Renderer createTextureFromFile:allowMipmaps:size:] PictureViewer
20.0ms	0.8%	-[ES2Renderer createTextureFromFile:allowMipmaps:size:] PictureViewer
15.0ms	0.6%	-[ES2Renderer createTextureFromFile:allowMipmaps:size:] PictureViewer
11.0ms	0.4%	-[ES2Renderer createTextureFromFile:allowMipmaps:size:] PictureViewer



# System Analysis

## Motivation



Where your threads  
spend time



How your threads interact  
with other threads

# System Trace

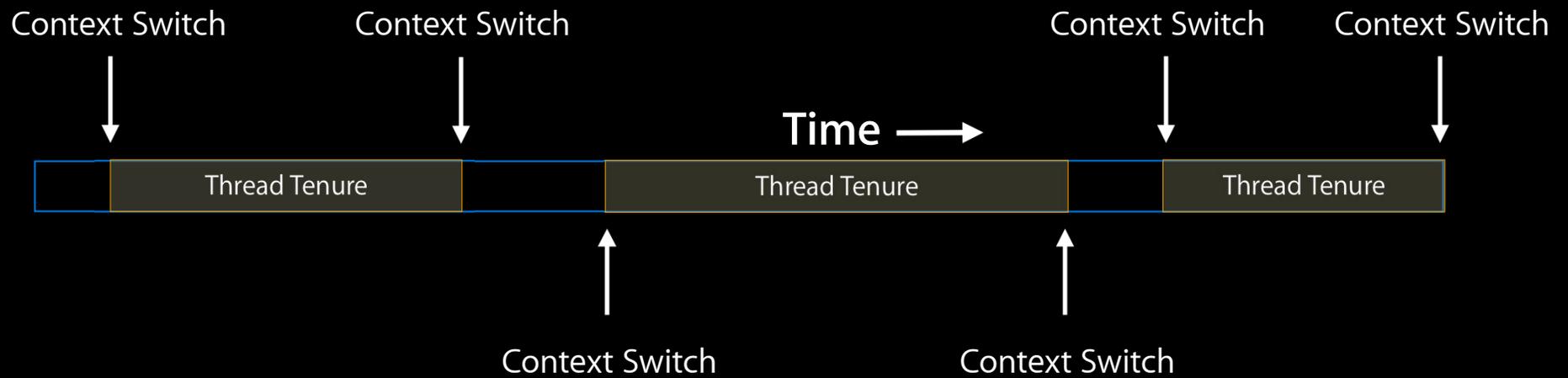
## What does it do?



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

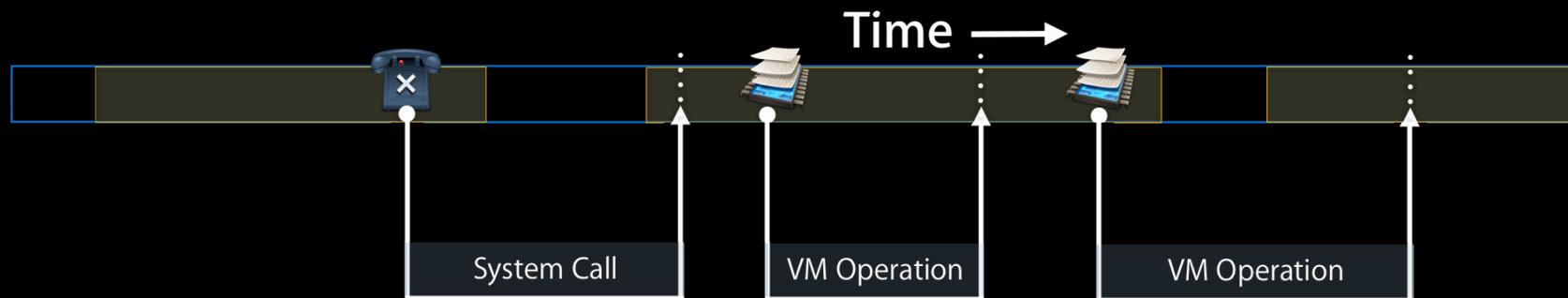
# System Trace

## How does it work?



# System Trace

How does it work?



# System Trace

## Scheduling Instrument



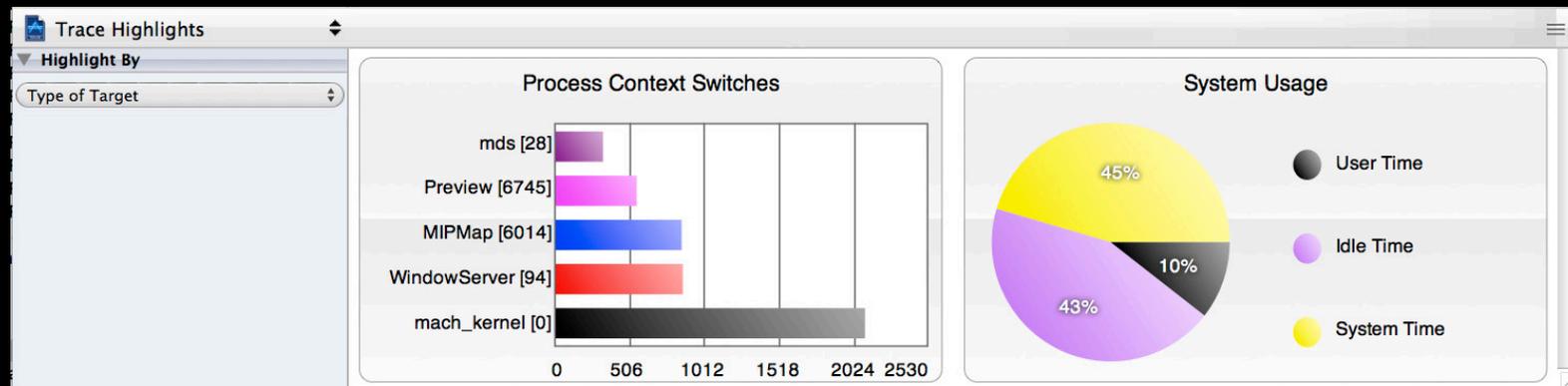
- Records all thread context switches
- Context switch reasons
- Threads running in user, system or idling
- Thread priorities

# 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



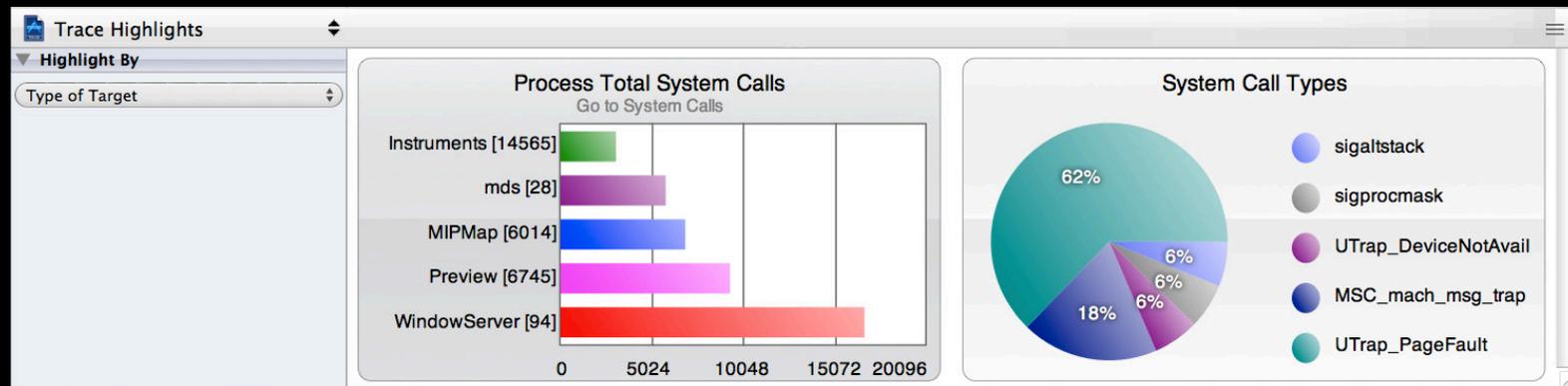
- Records all system calls made to the kernel
  - e.g. read, select, locks
- May incur context switch
- Cause unwanted serialization

# System Trace

## System Calls Instrument



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



# System Trace

## VM Operations Instrument



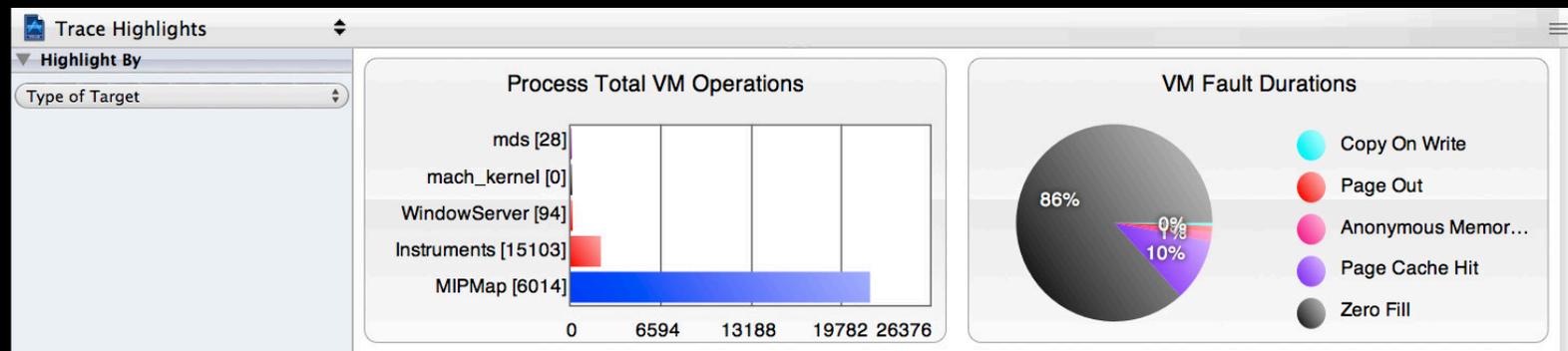
- Records uses of unmapped memory pages
  - First use of new memory allocations
  - First write to shared memory
  - Page-ins and outs from disk

# System Trace

## VM Operations Calls Instrument

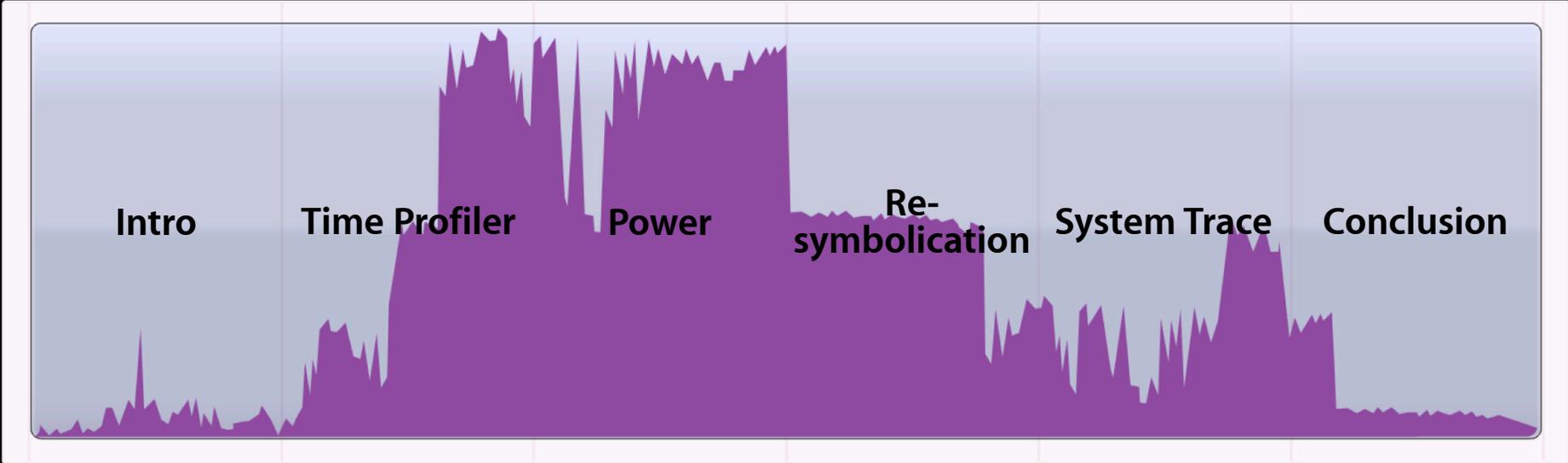


- Rules of thumb
  - Minimize memory use generally reduces page faults
  - Reuse memory allocations
  - Pre-allocate memory for latency sensitive programs



# Demo

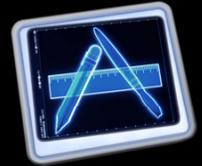
## System Trace



# 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](mailto: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



