

Learning Instruments

Profiling your App

Session 409

David M. O'Rourke

Performance Tools Engineering Manager

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

What You'll Learn Today

What You'll Learn Today

- Elements of performance

What You'll Learn Today

- Elements of performance
- Methodology to improve performance

What You'll Learn Today

- Elements of performance
- Methodology to improve performance
- Instruments tour

What You'll Learn Today

- Elements of performance
- Methodology to improve performance
- Instruments tour
- iOS App optimization demonstrations

What Is Performance?

Fast



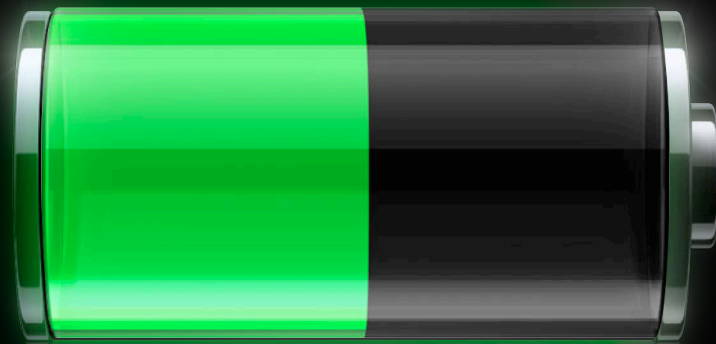
Responsive



Efficient



Power



Power

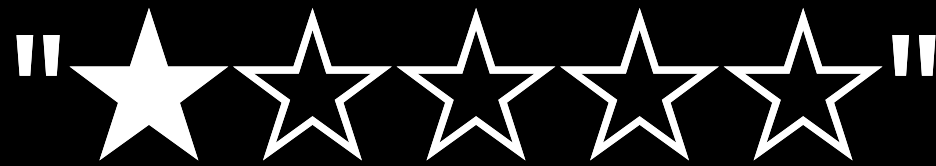


"Slow and buggy"

"Crashes all the time"







Performance Is a Feature of Your App

Performance Is a Feature of Your App

- Xcode to author/build your app

Performance Is a Feature of Your App

- Xcode to author/build your app
- Interface Builder to design your app

Performance Is a Feature of Your App

- Xcode to author/build your app
- Interface Builder to design your app
- Instruments to profile your app
 - Optimize performance
 - Reduces crashes and terminations
 - Can improve power usage

Performance Profile Process

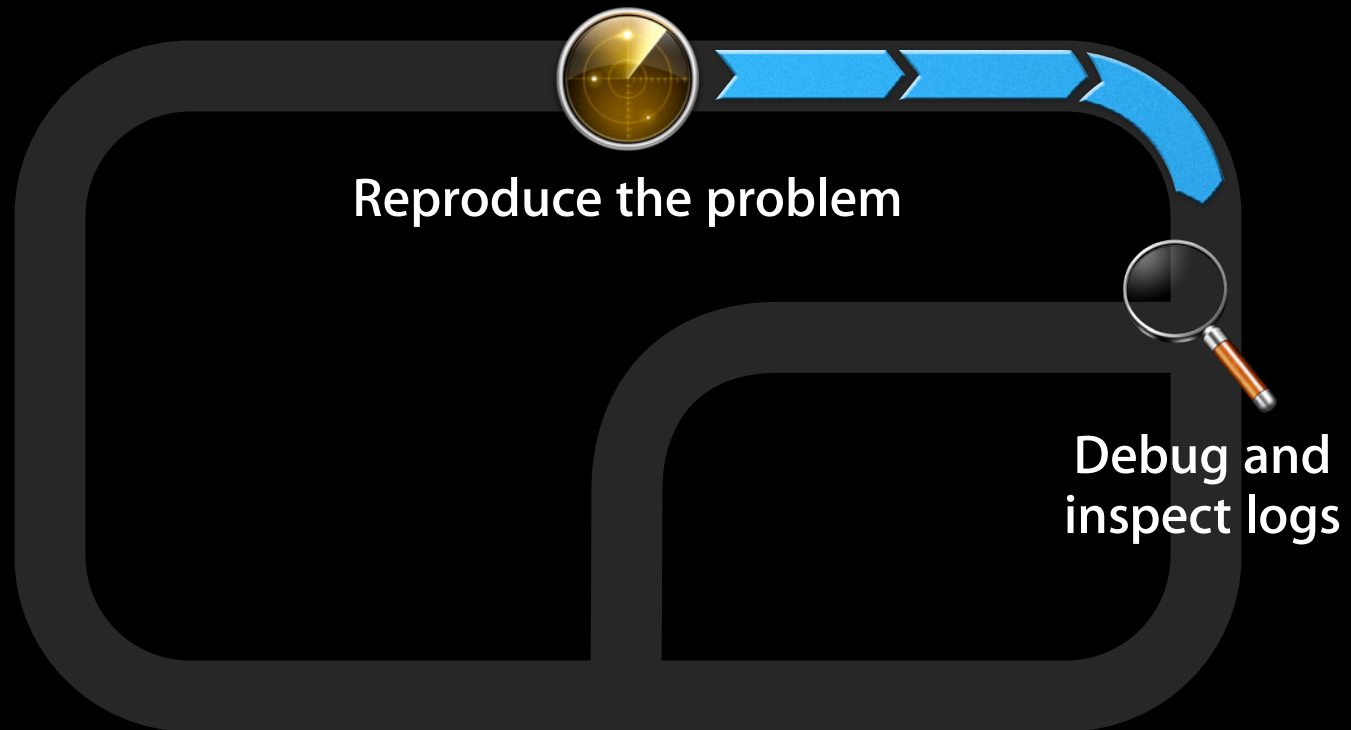
Debug Process

Debug Process

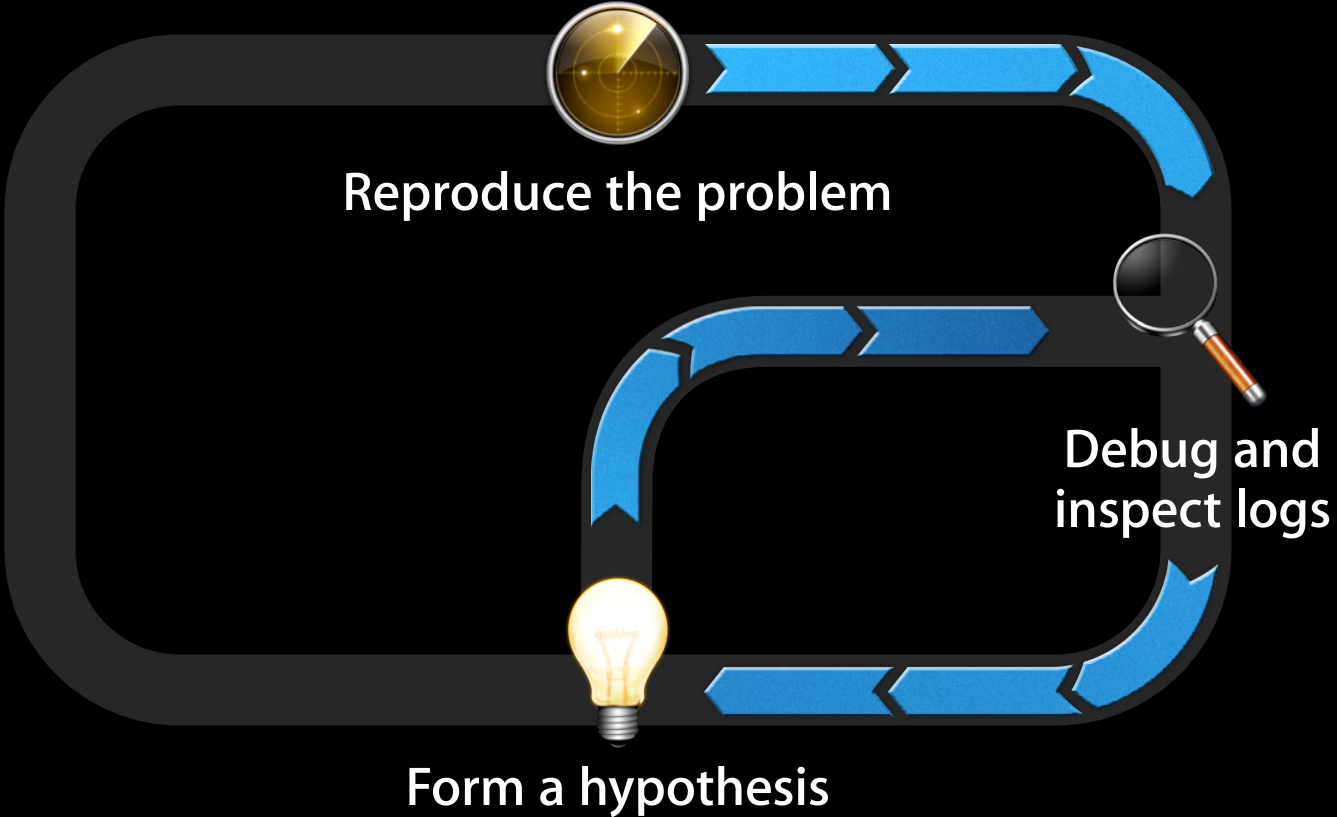


Reproduce the problem

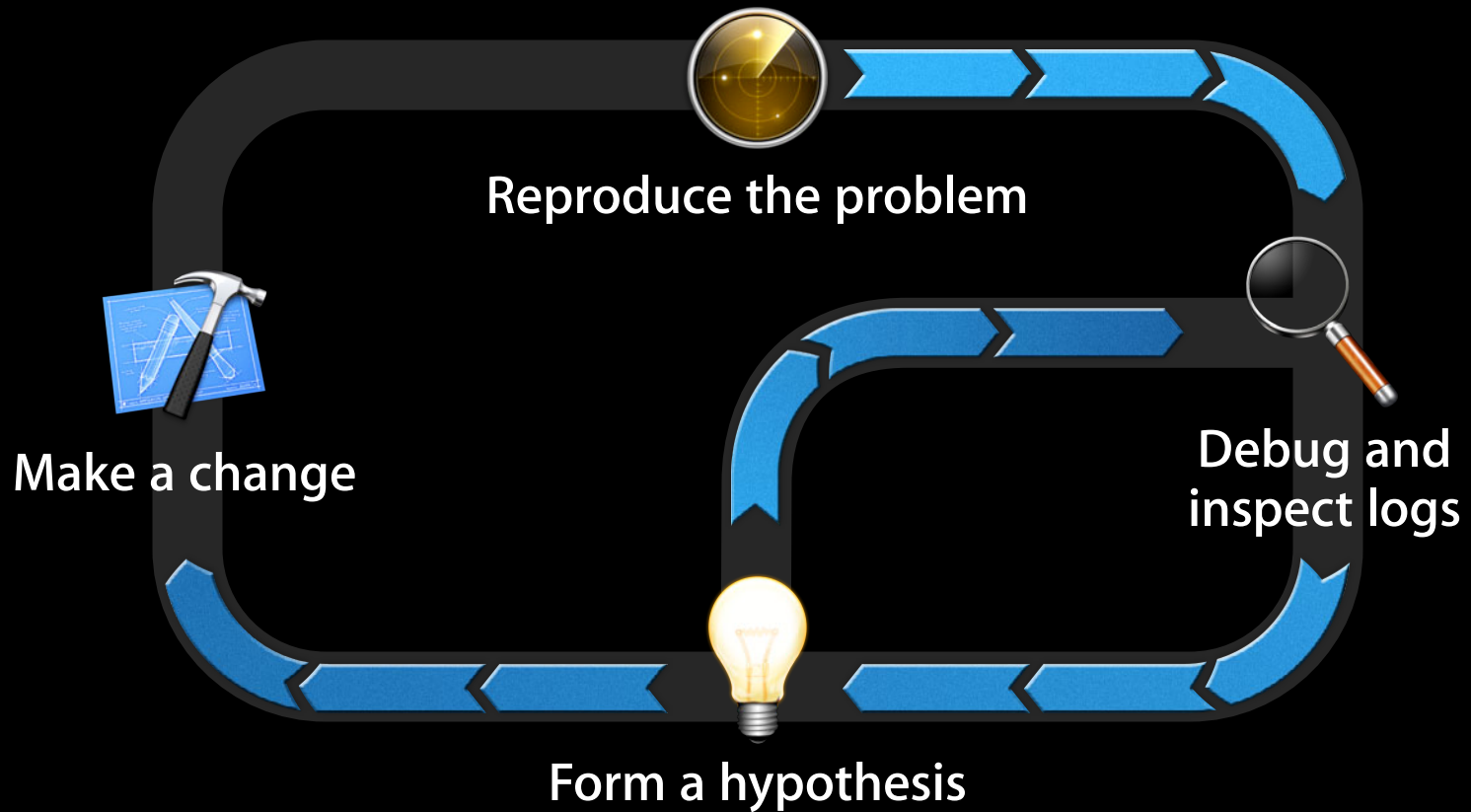
Debug Process



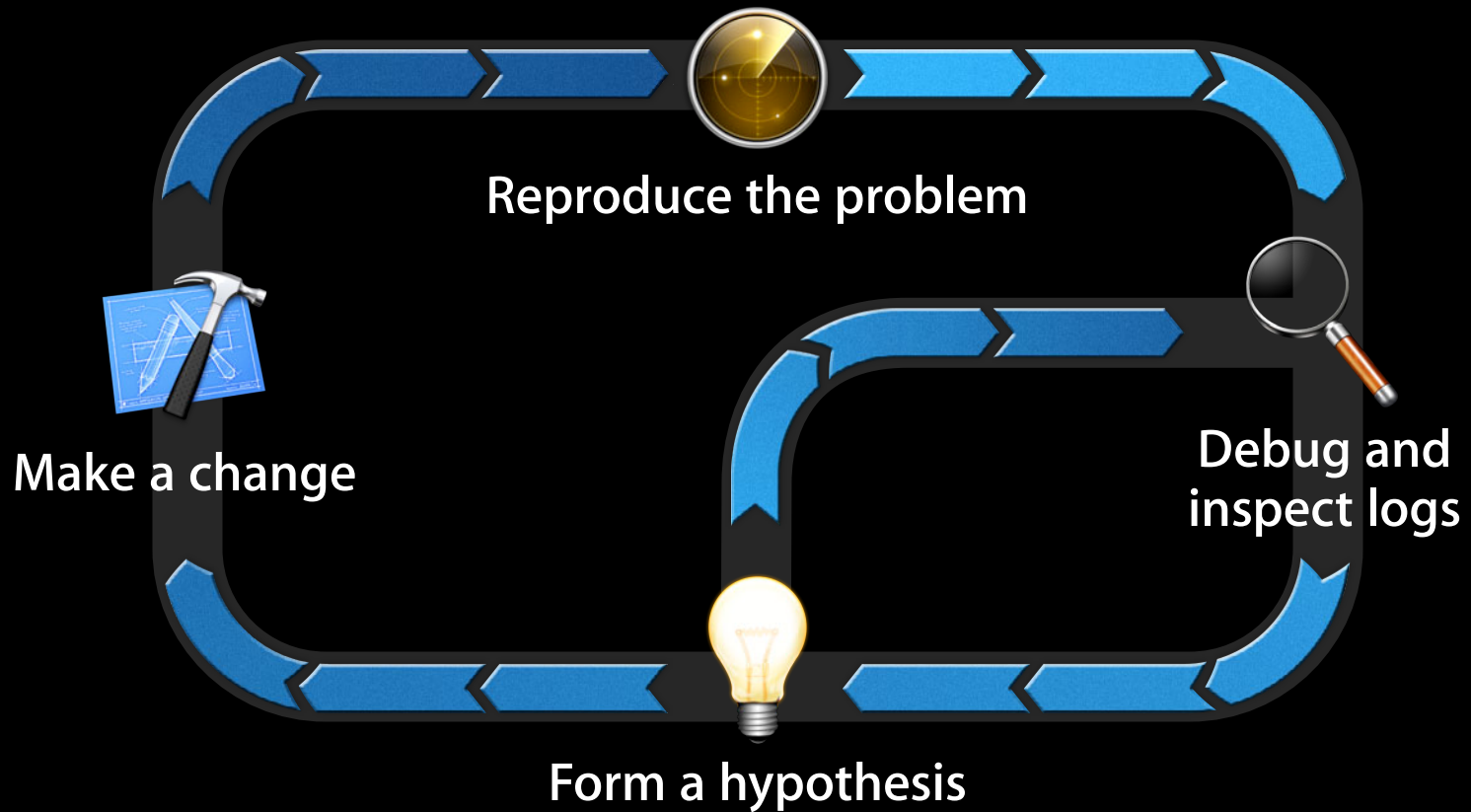
Debug Process



Debug Process



Debug Process



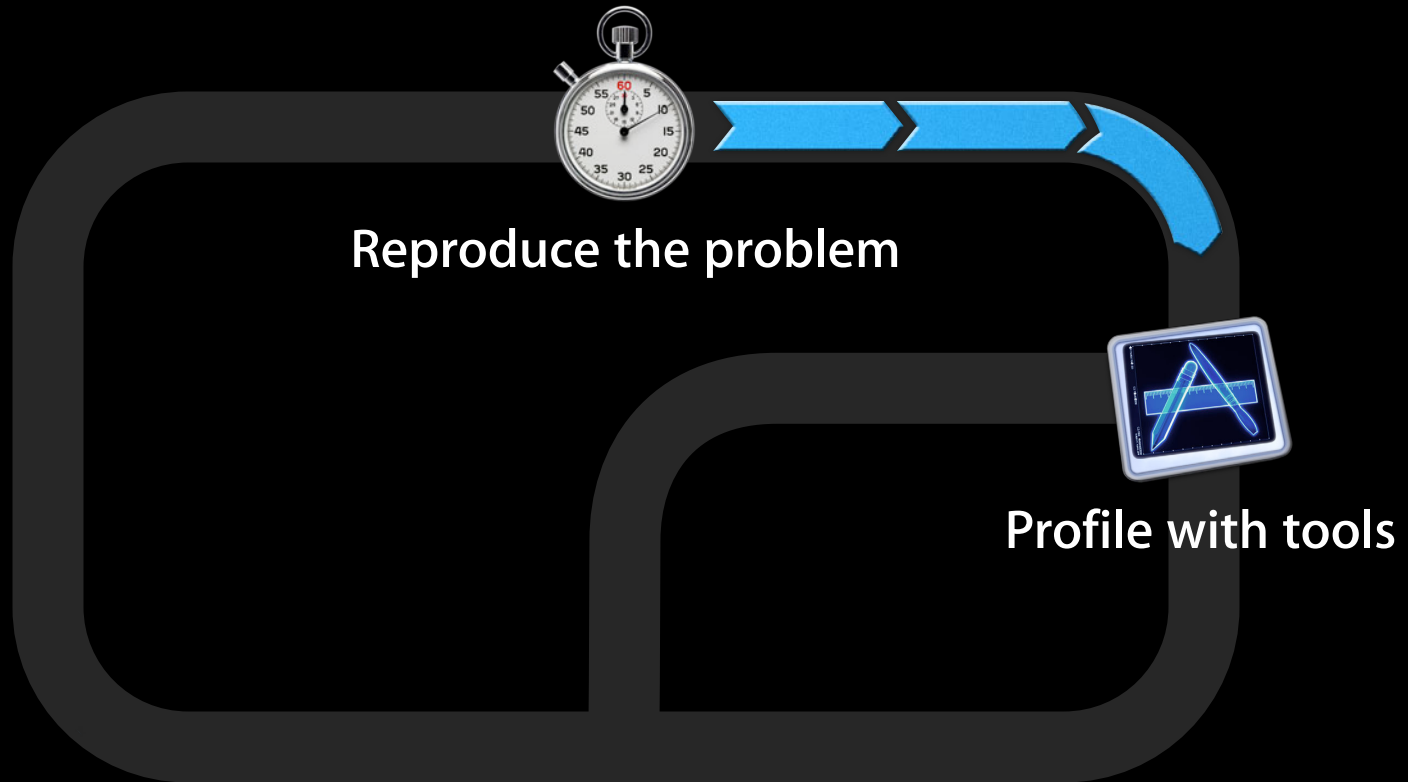
Profile Process

Profile Process

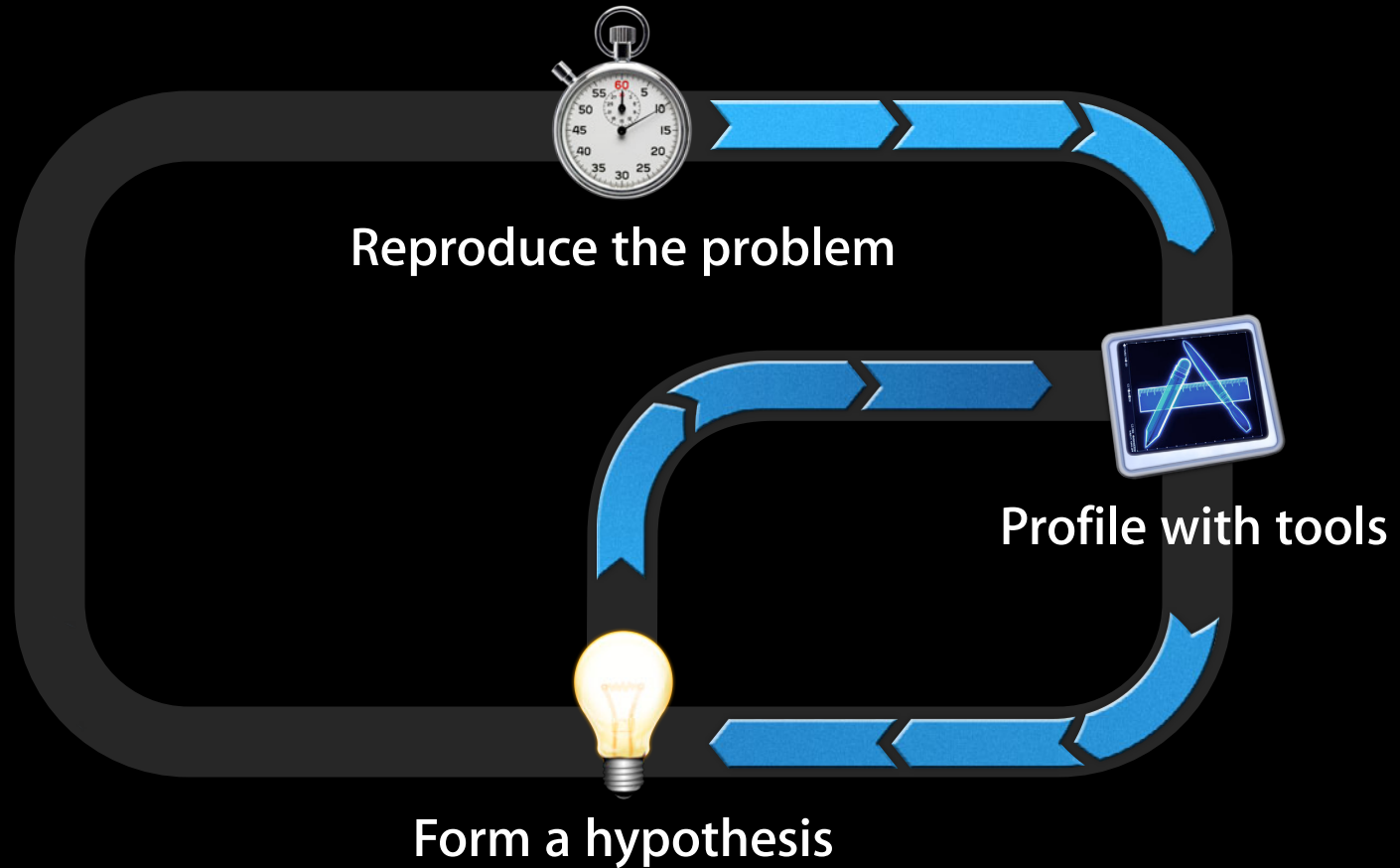


Reproduce the problem

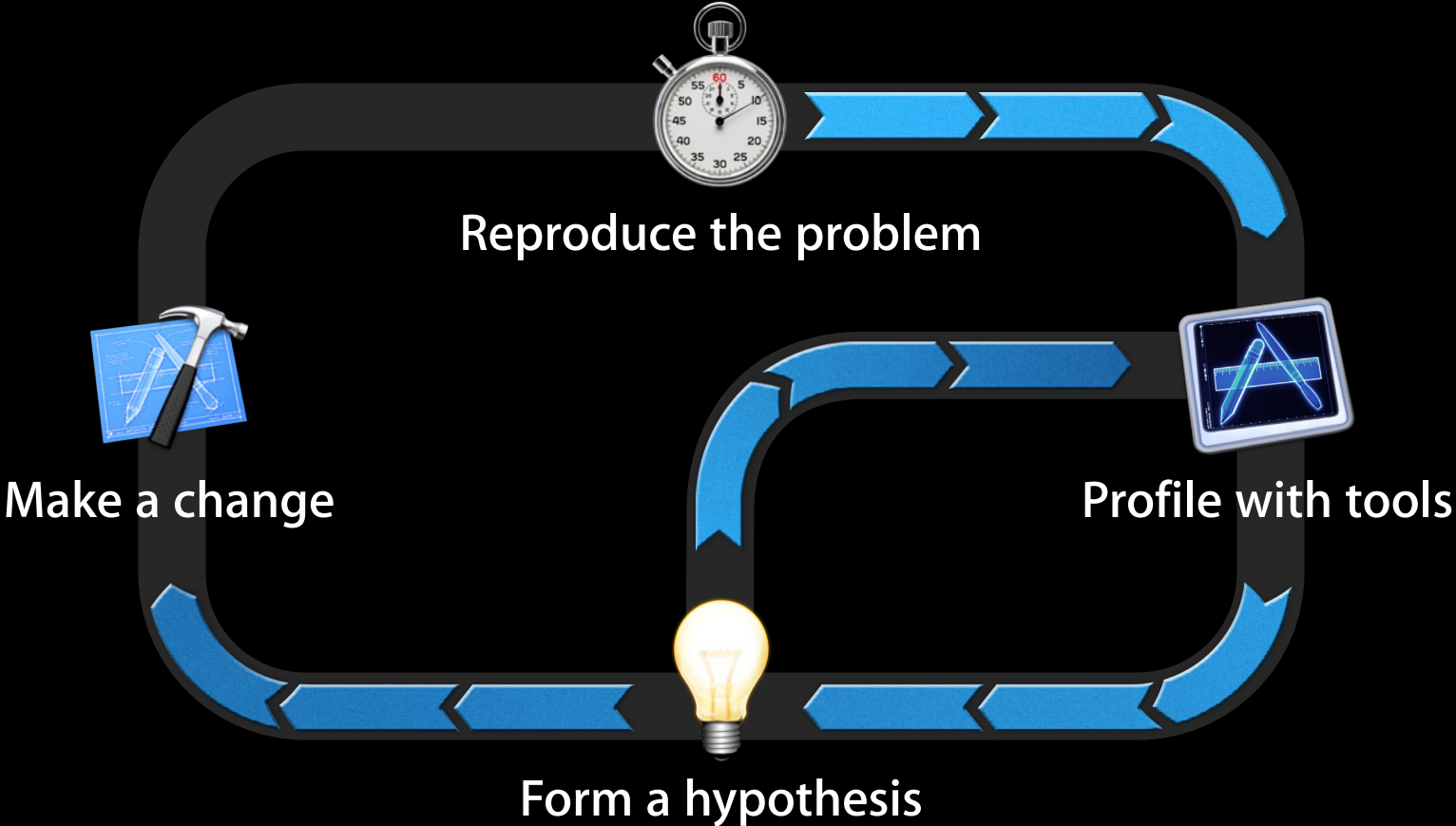
Profile Process



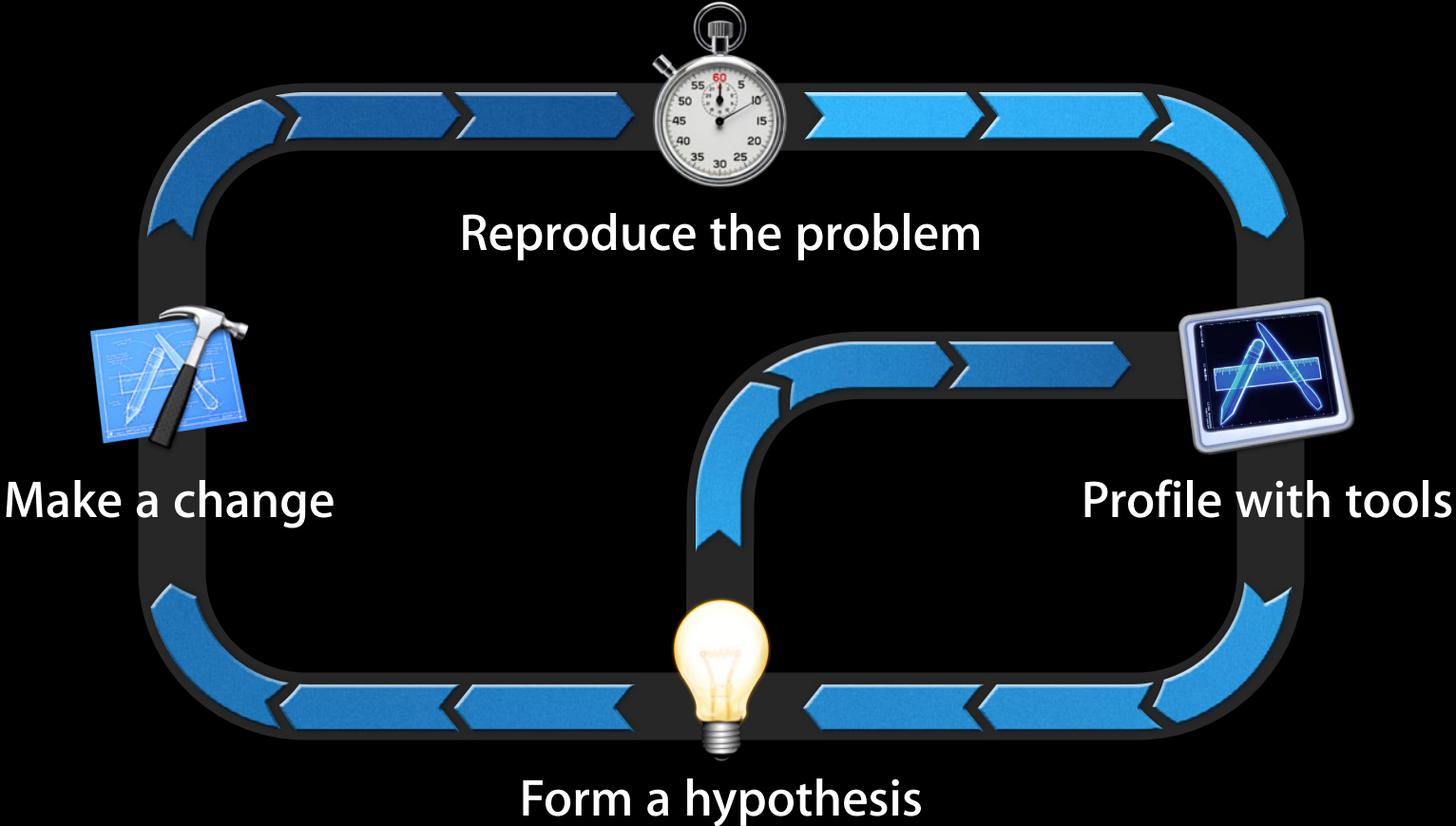
Profile Process



Profile Process



Profile Process



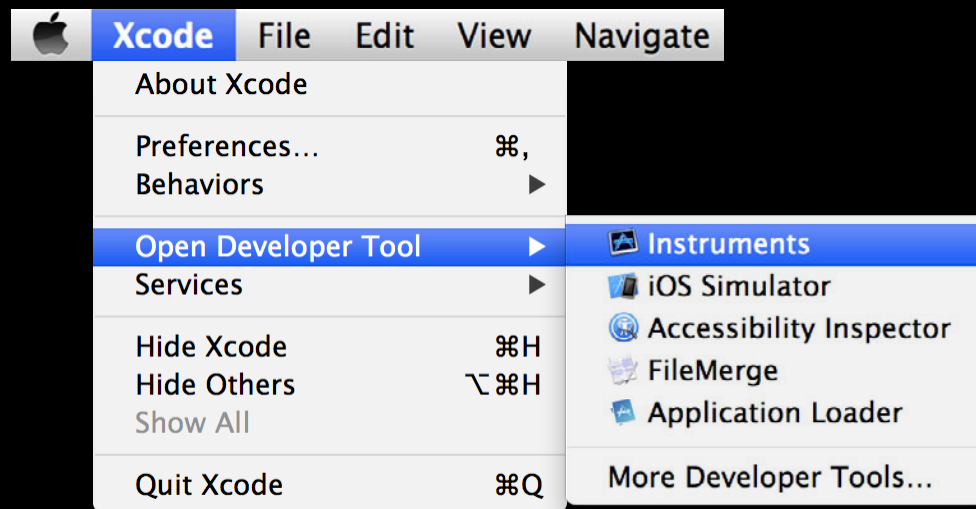
Where Can I Find Instruments?

Where Can I Find Instruments?

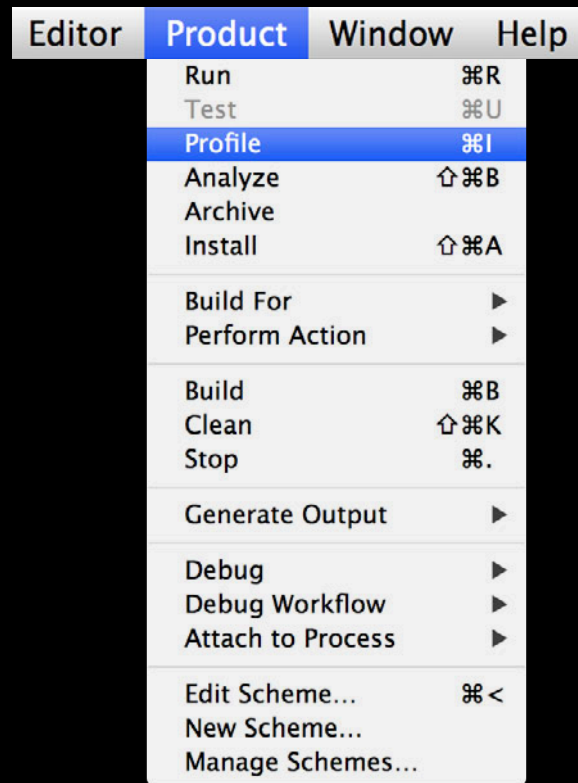


How Do I Access Instruments?

How Do I Access Instruments?

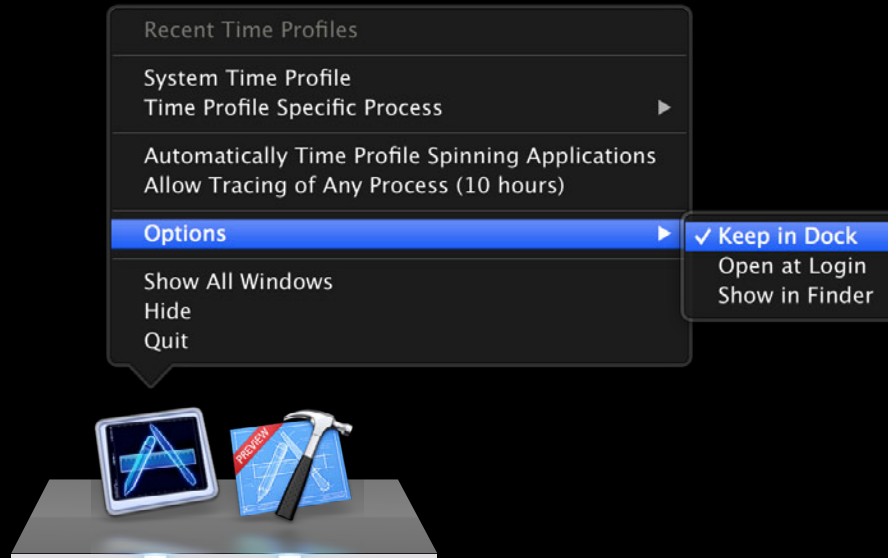


How Do I Access Instruments?

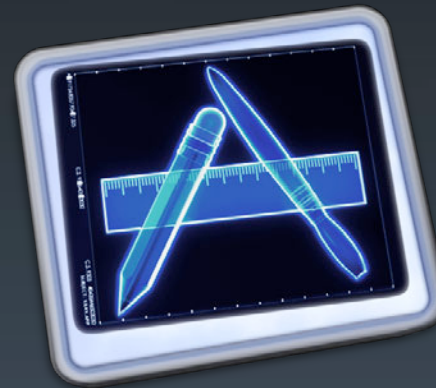


How Do I Access Instruments?

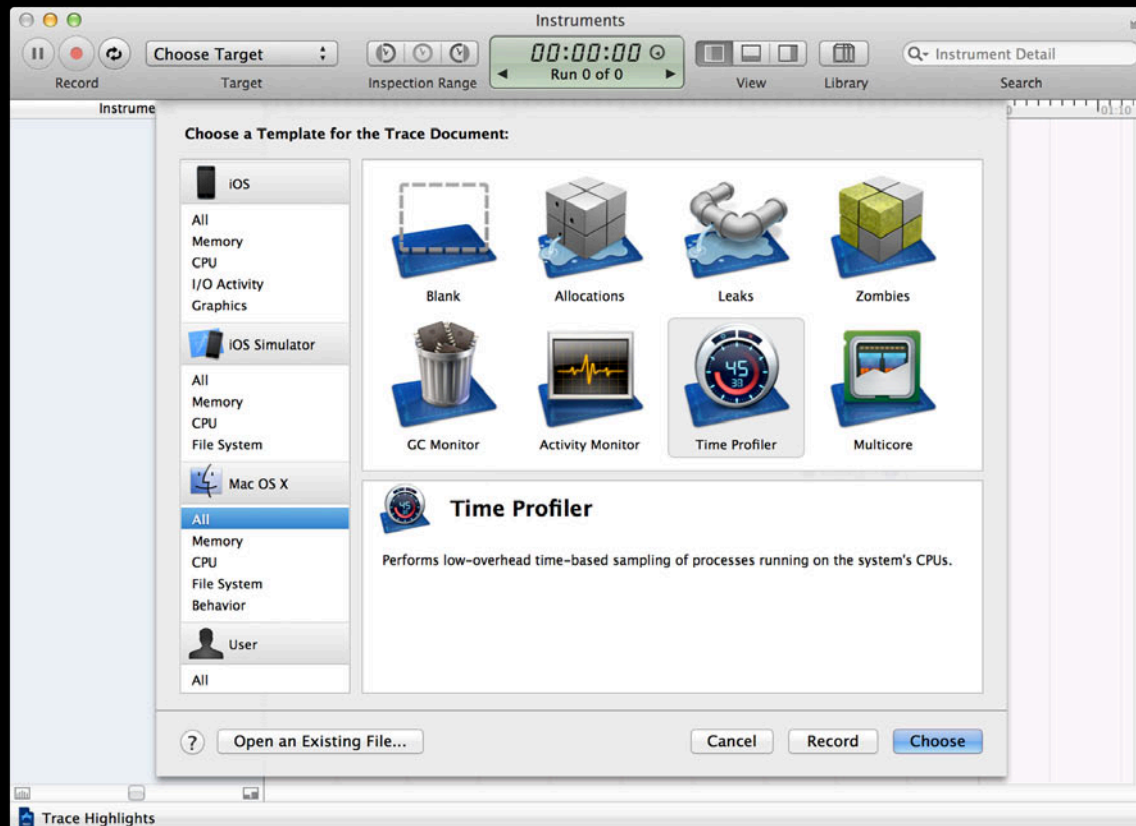
- Pro Tip
 - Right-click and choose Keep in Dock



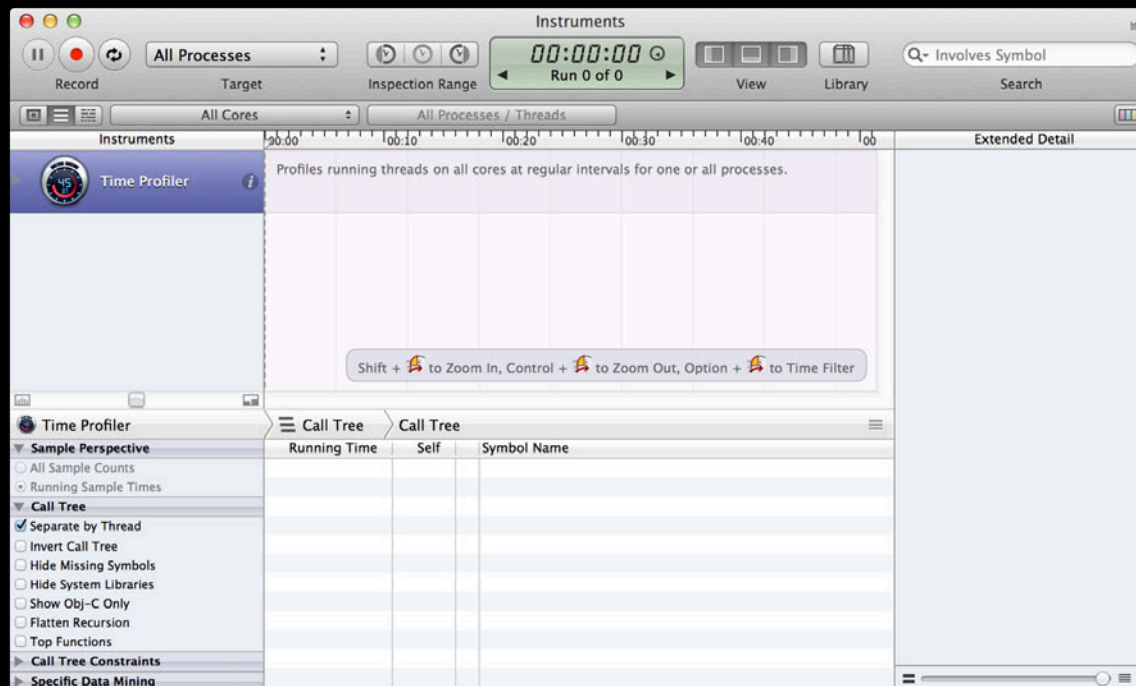
Instruments Tour



Document Model and Templates

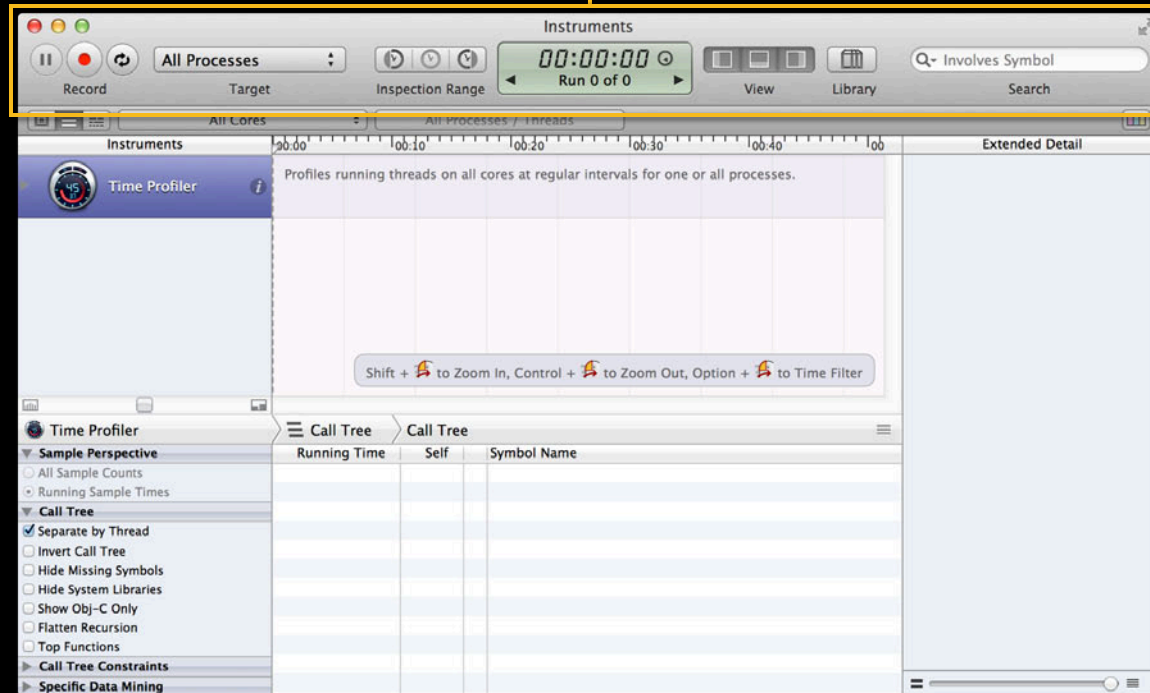


Tour



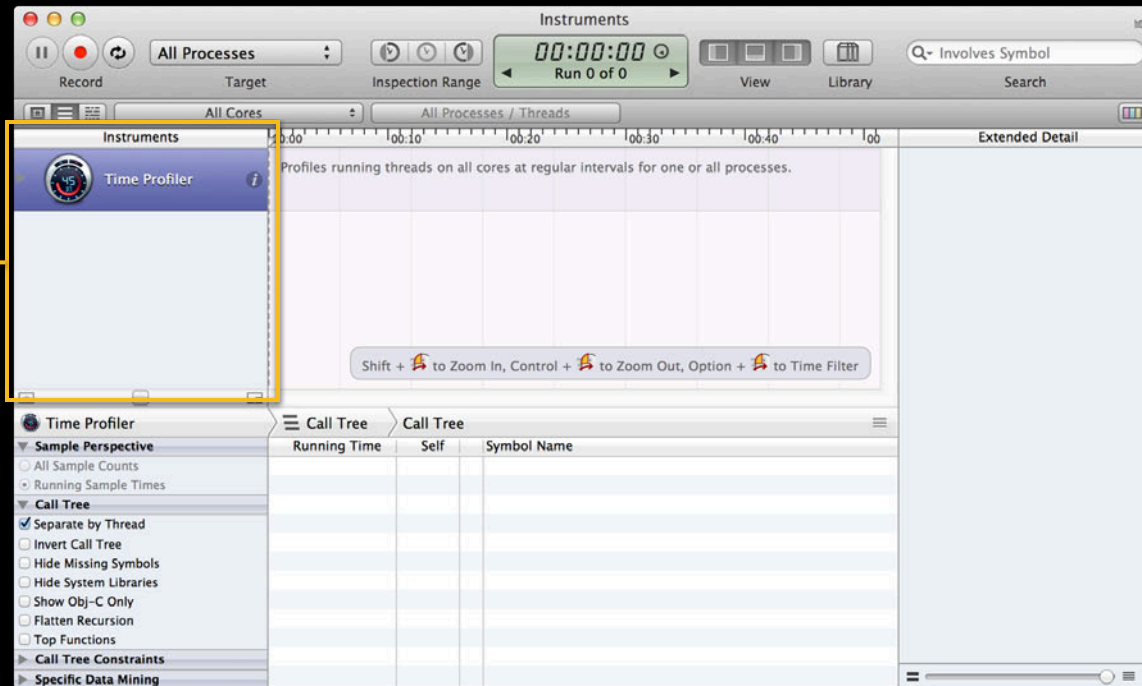
Tour

Toolbar

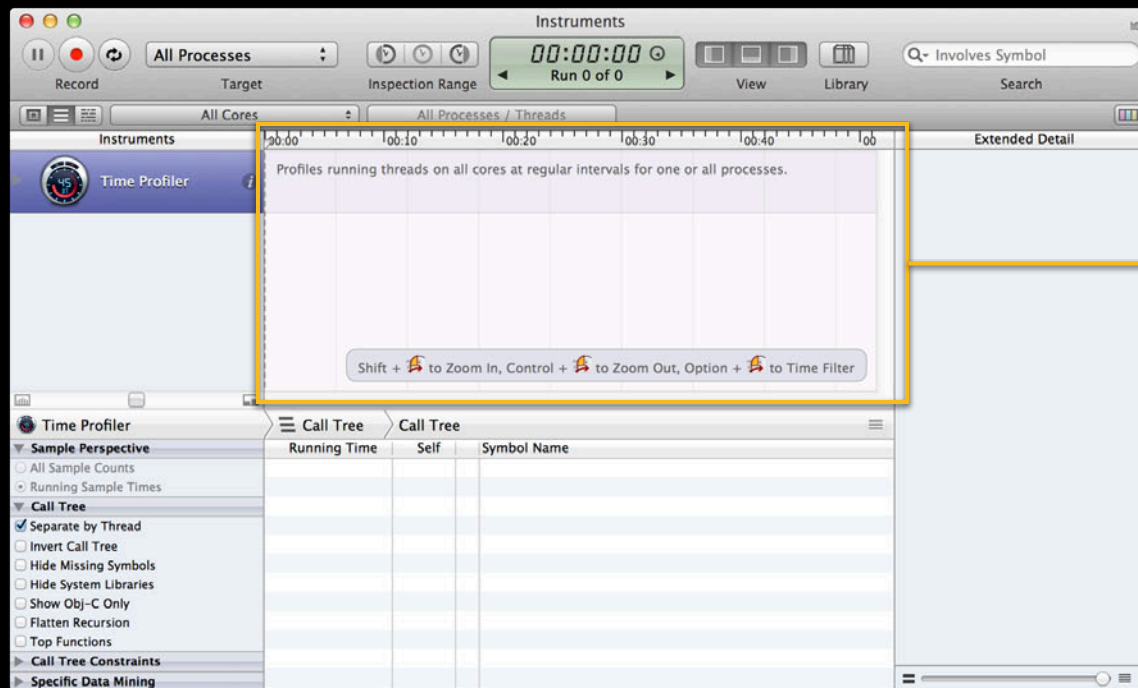


Tour

Strategies

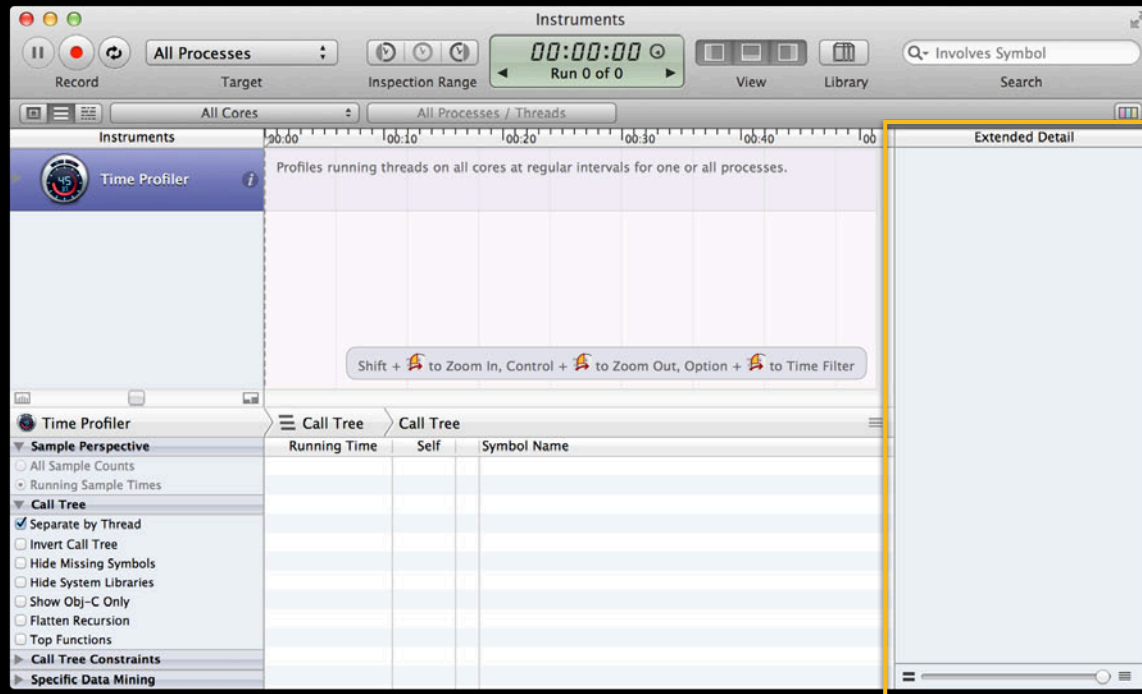


Tour



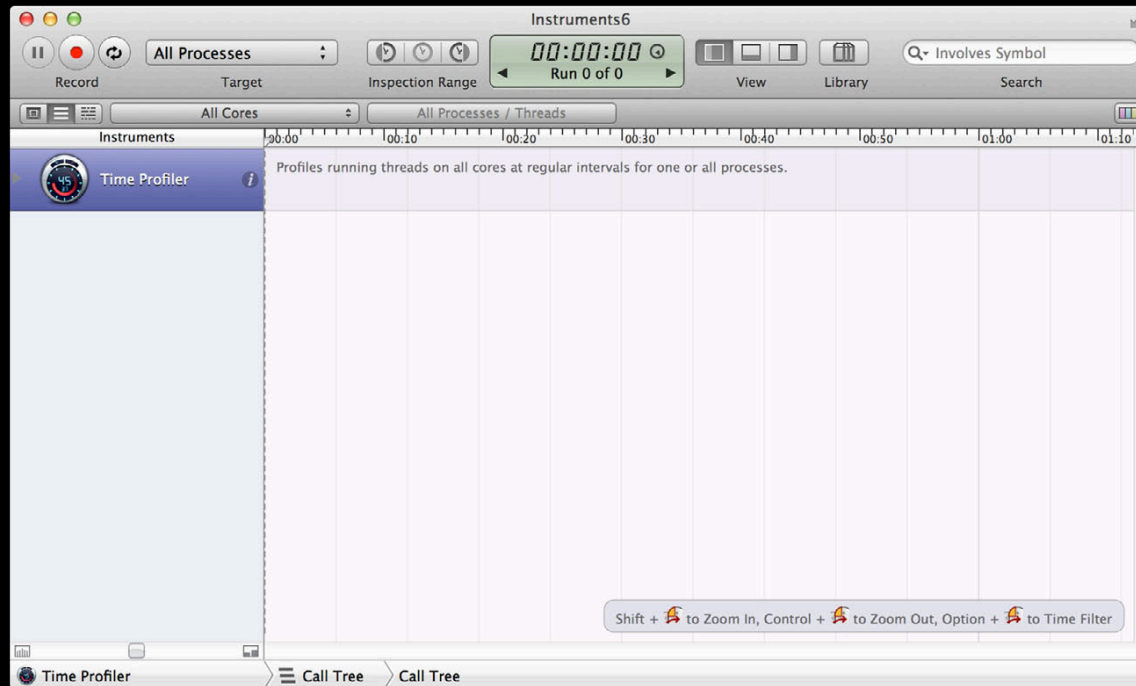
Timeline

Tour

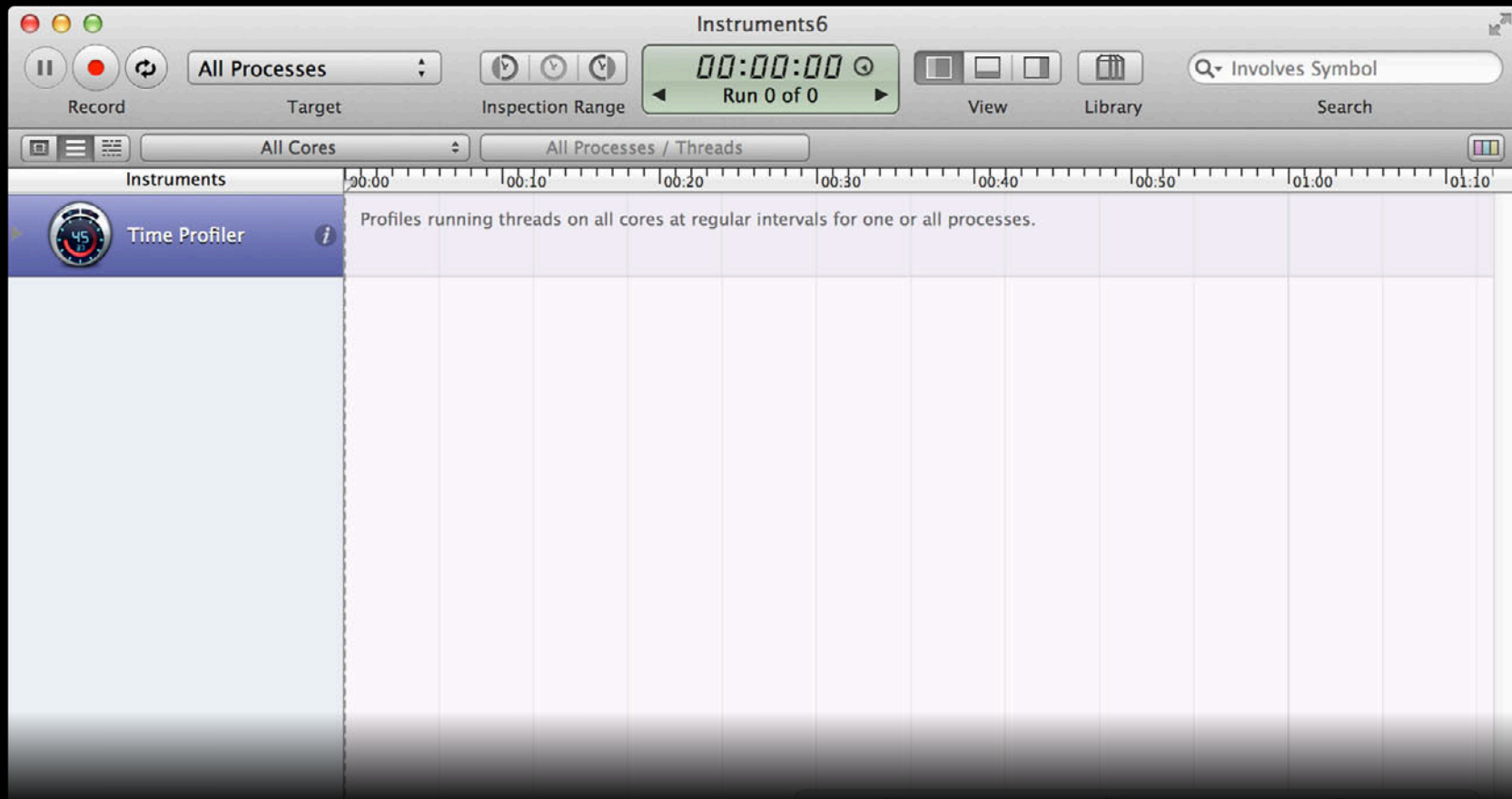


Extended Details

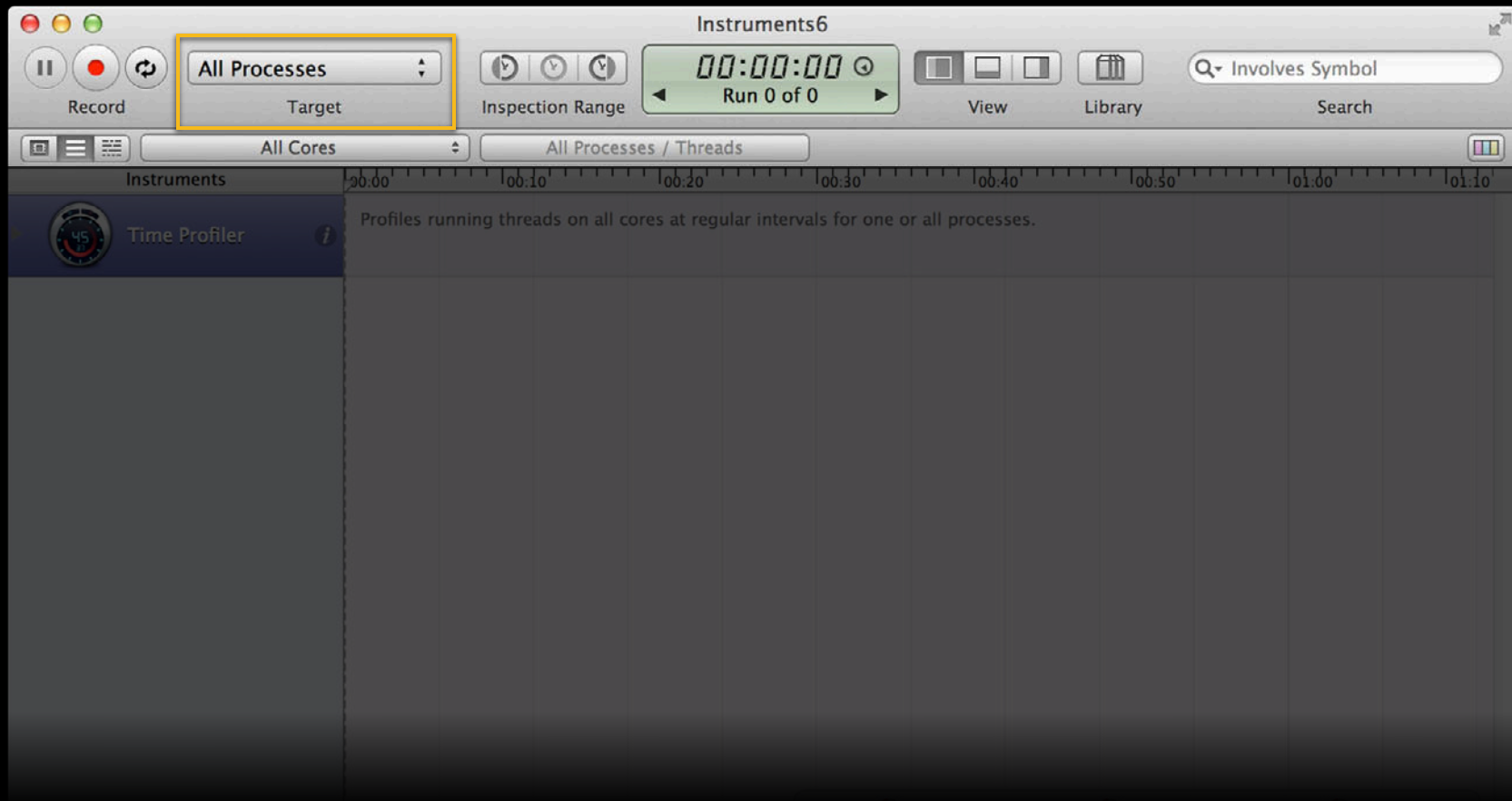
Toolbar



Toolbar

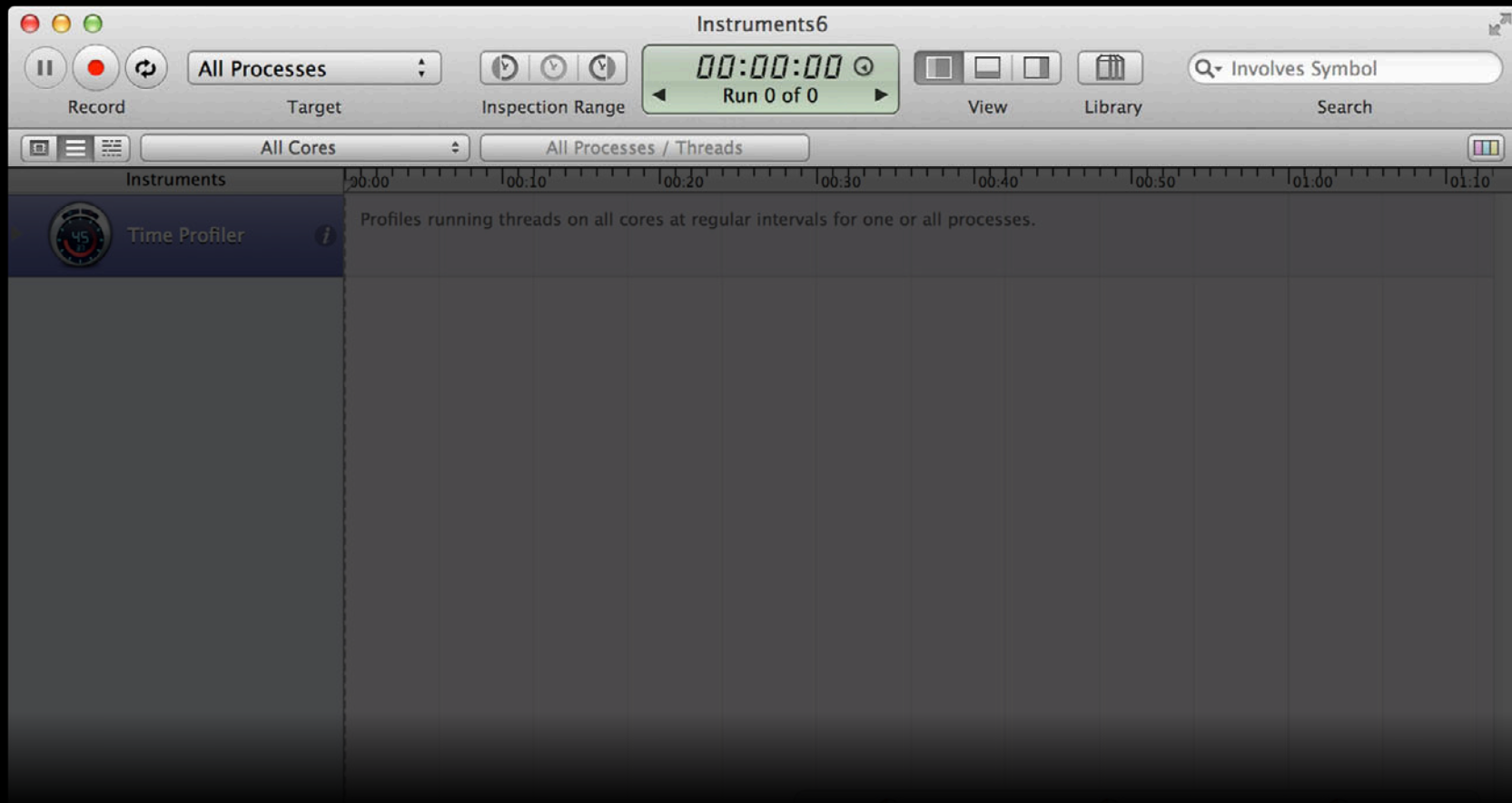


Toolbar



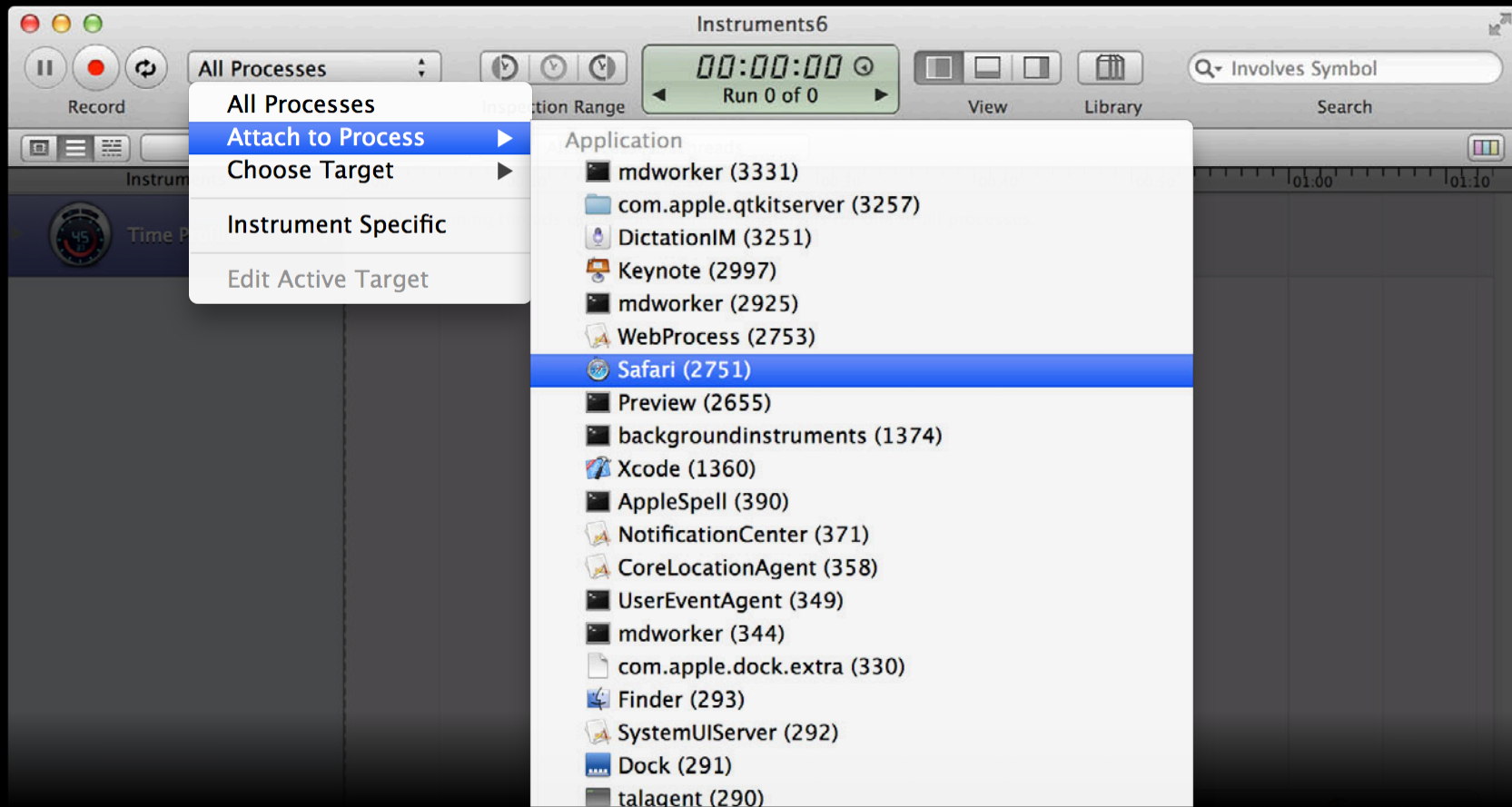
Toolbar

Target menu

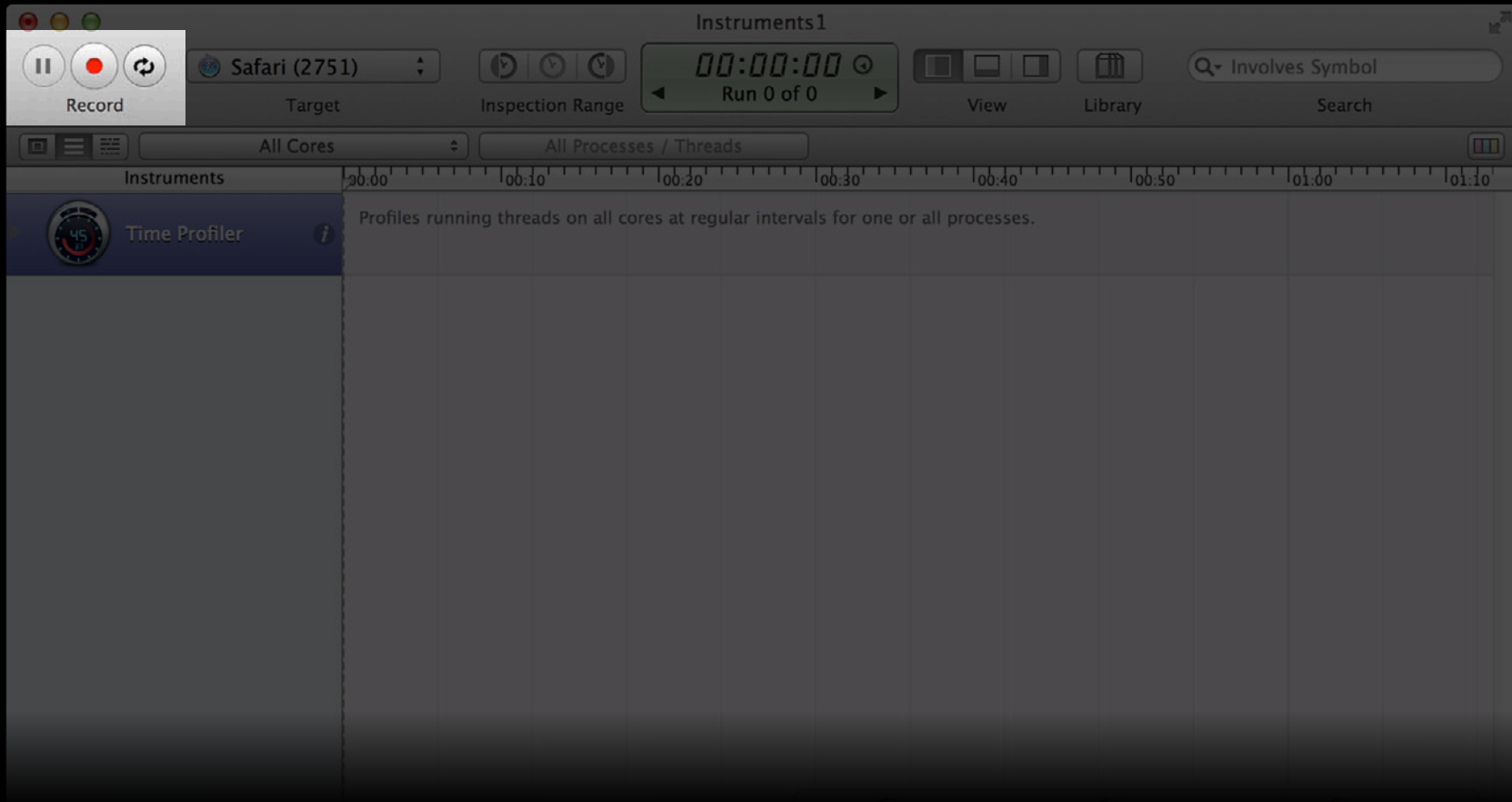


Toolbar

Target menu



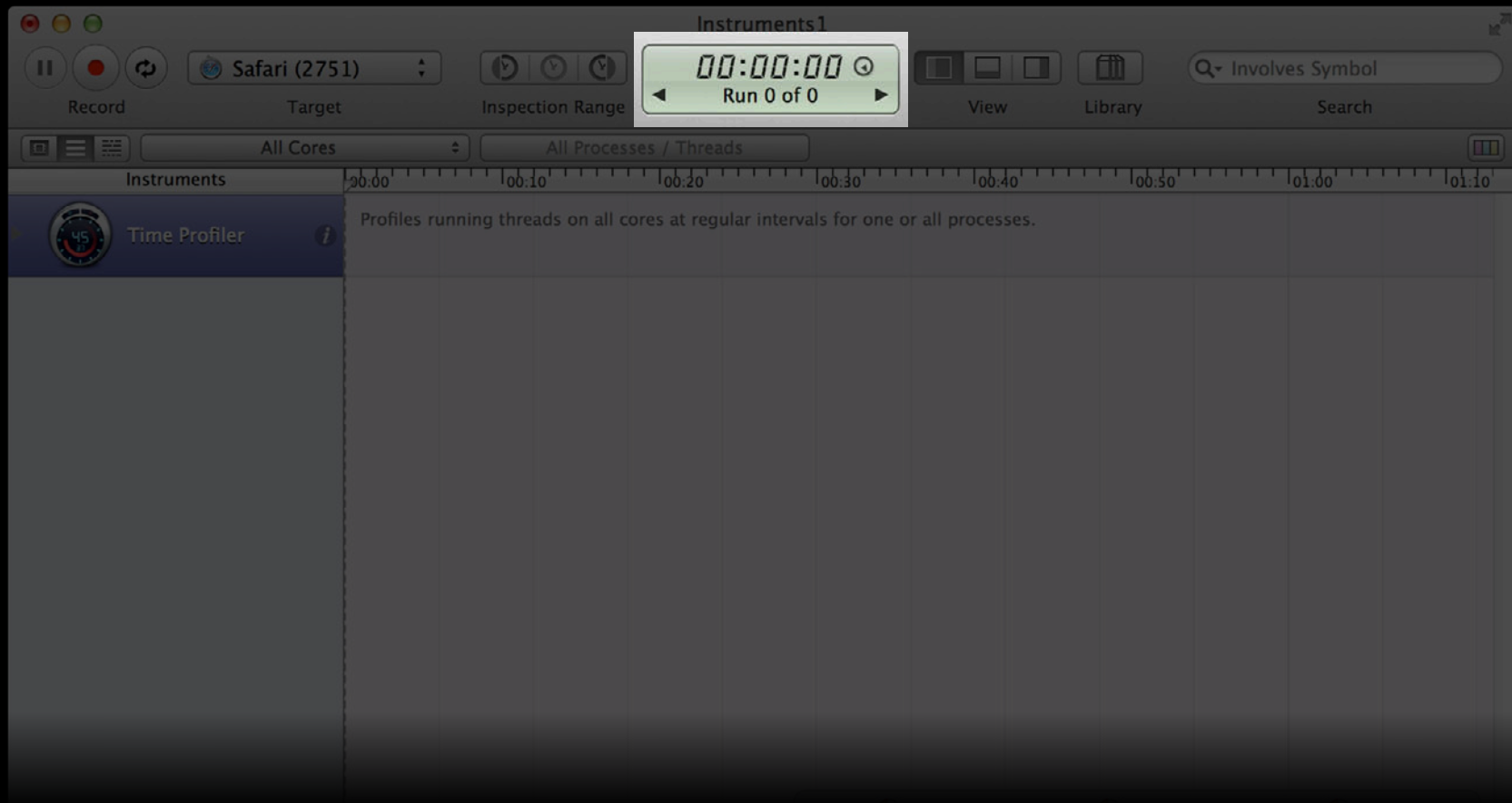
Toolbar



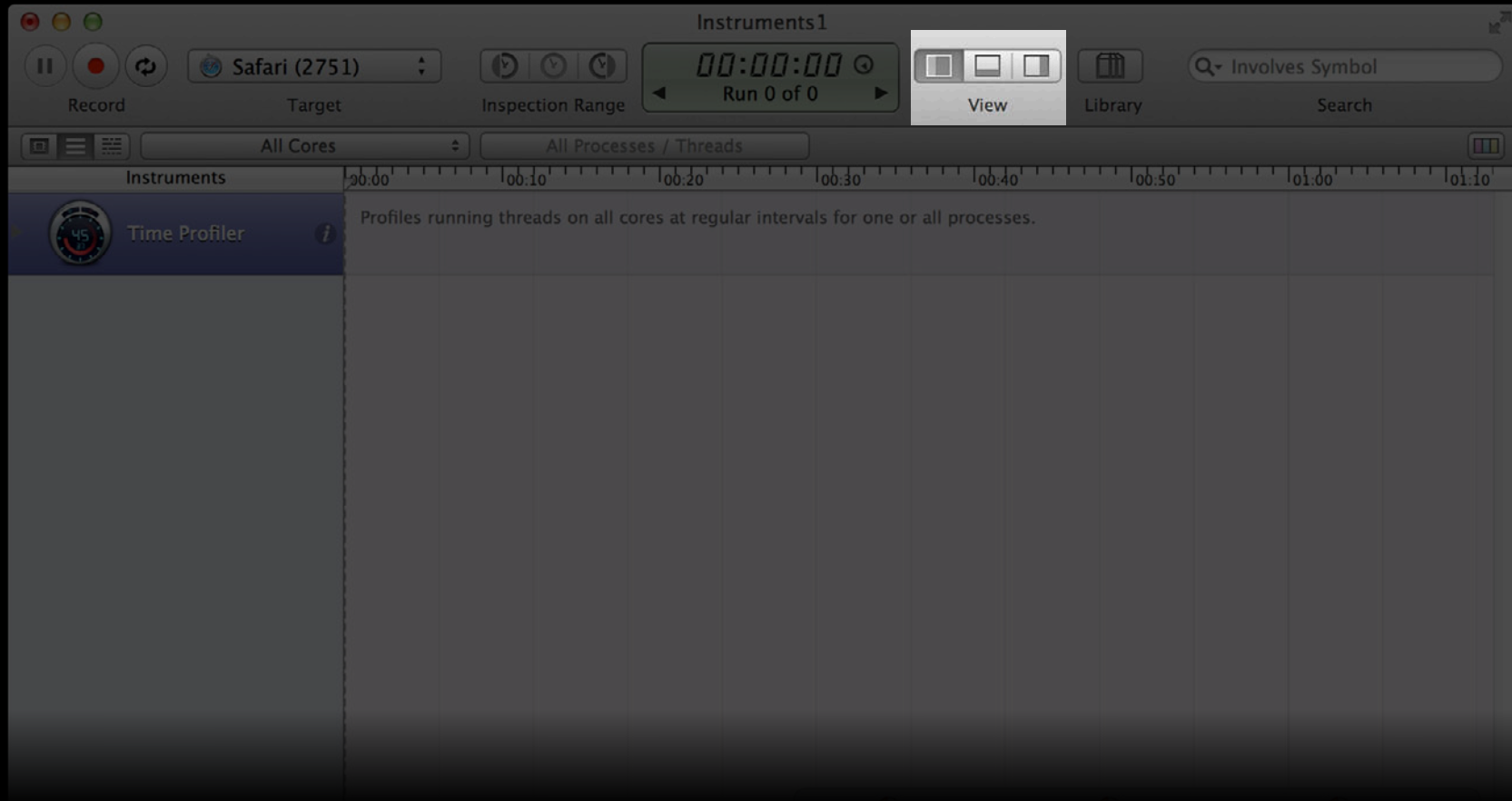
Toolbar



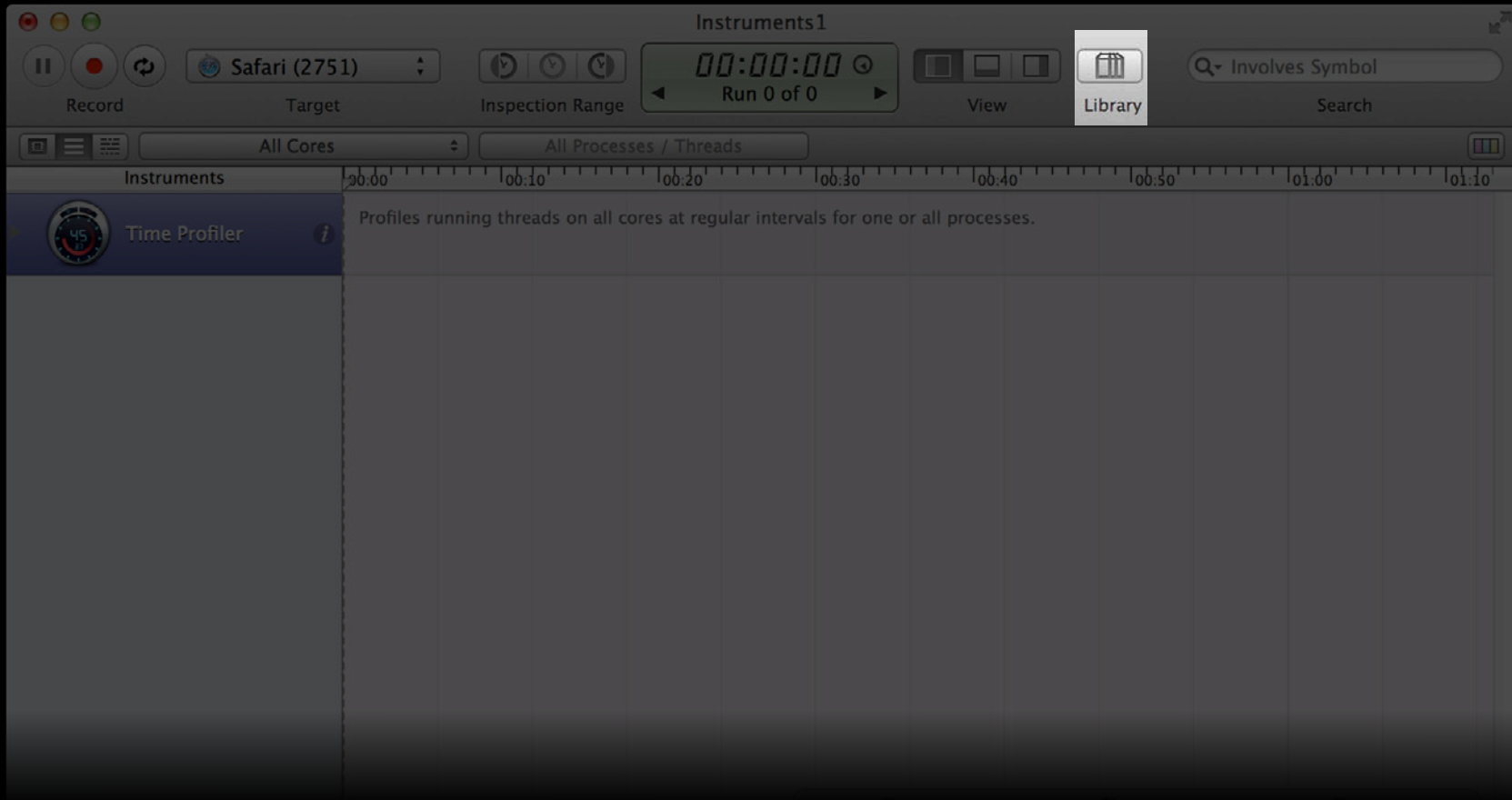
Toolbar



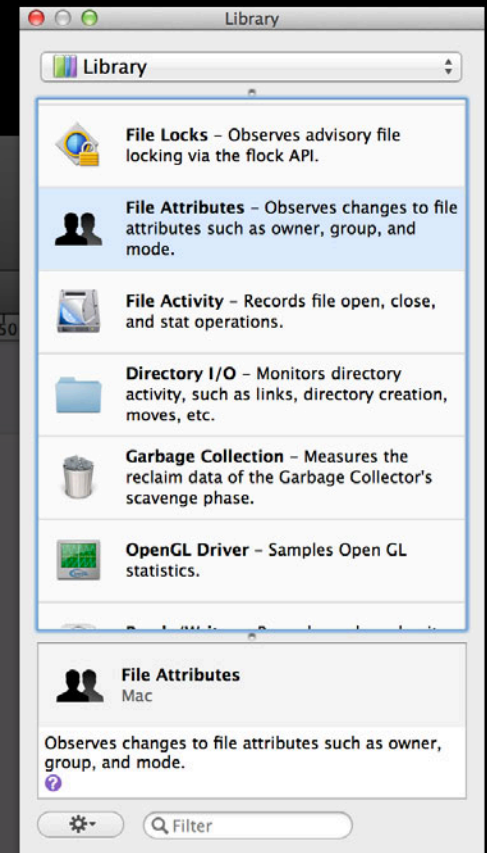
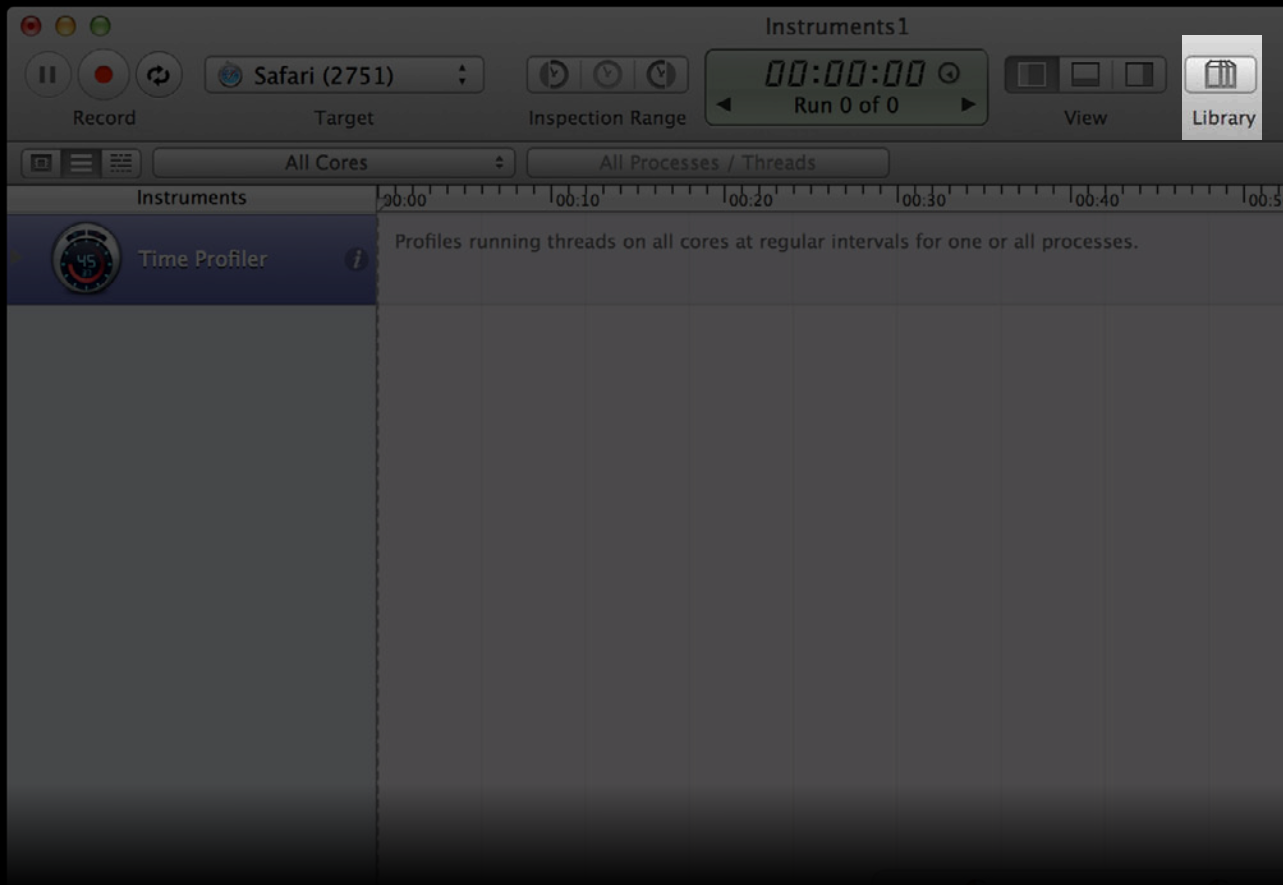
Toolbar



Toolbar

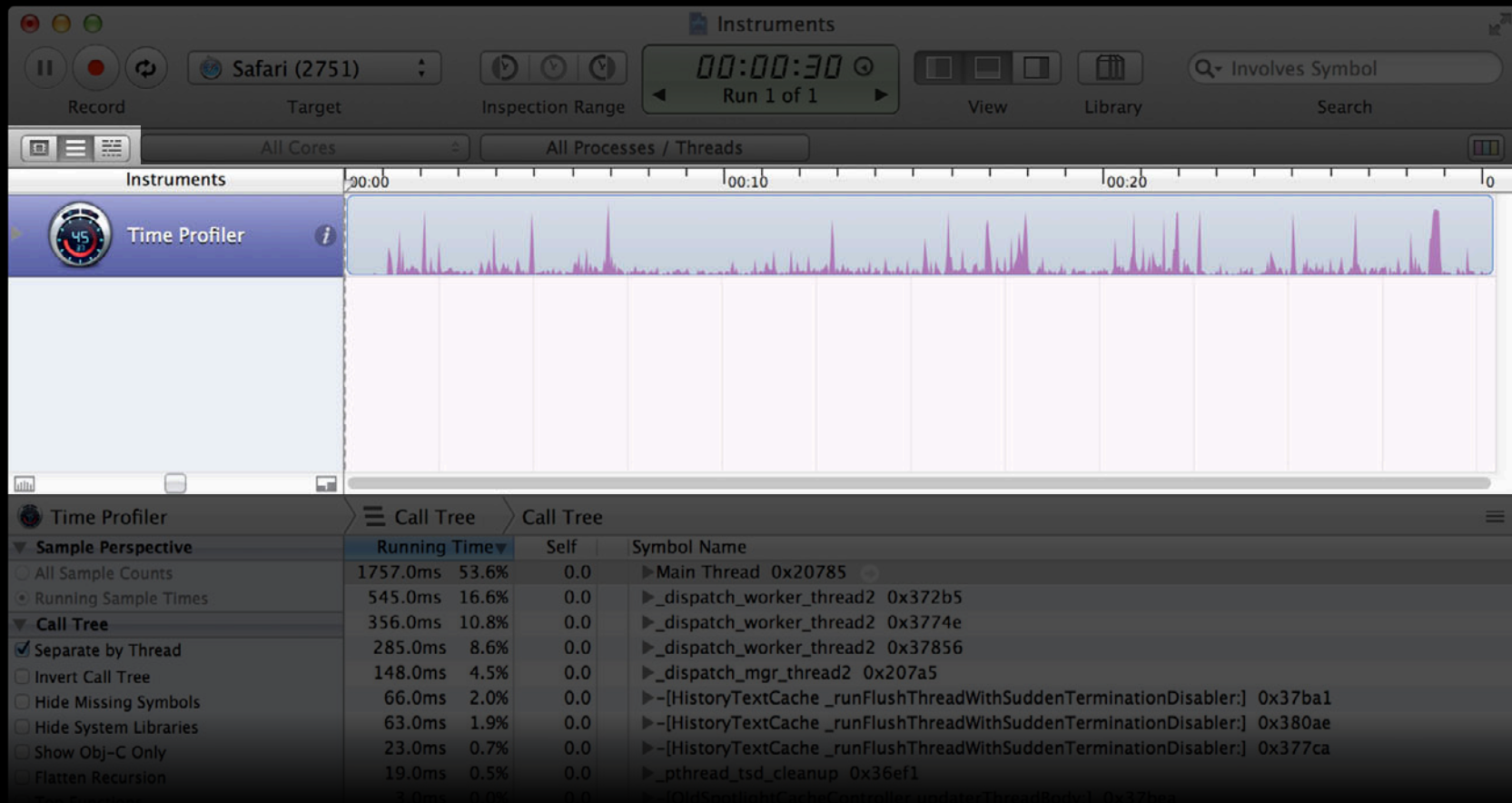


Toolbar



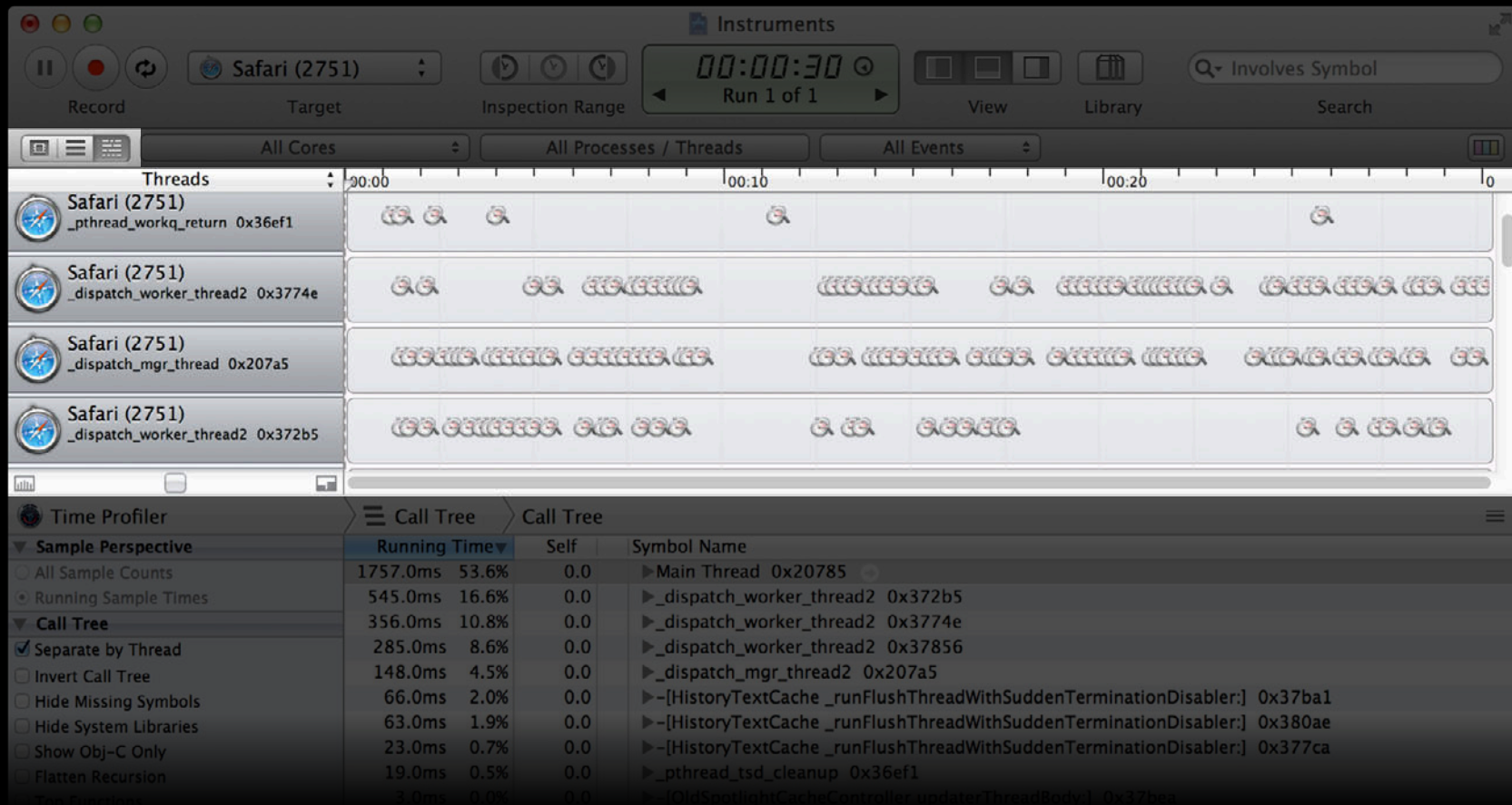
Timeline

Instrument strategy



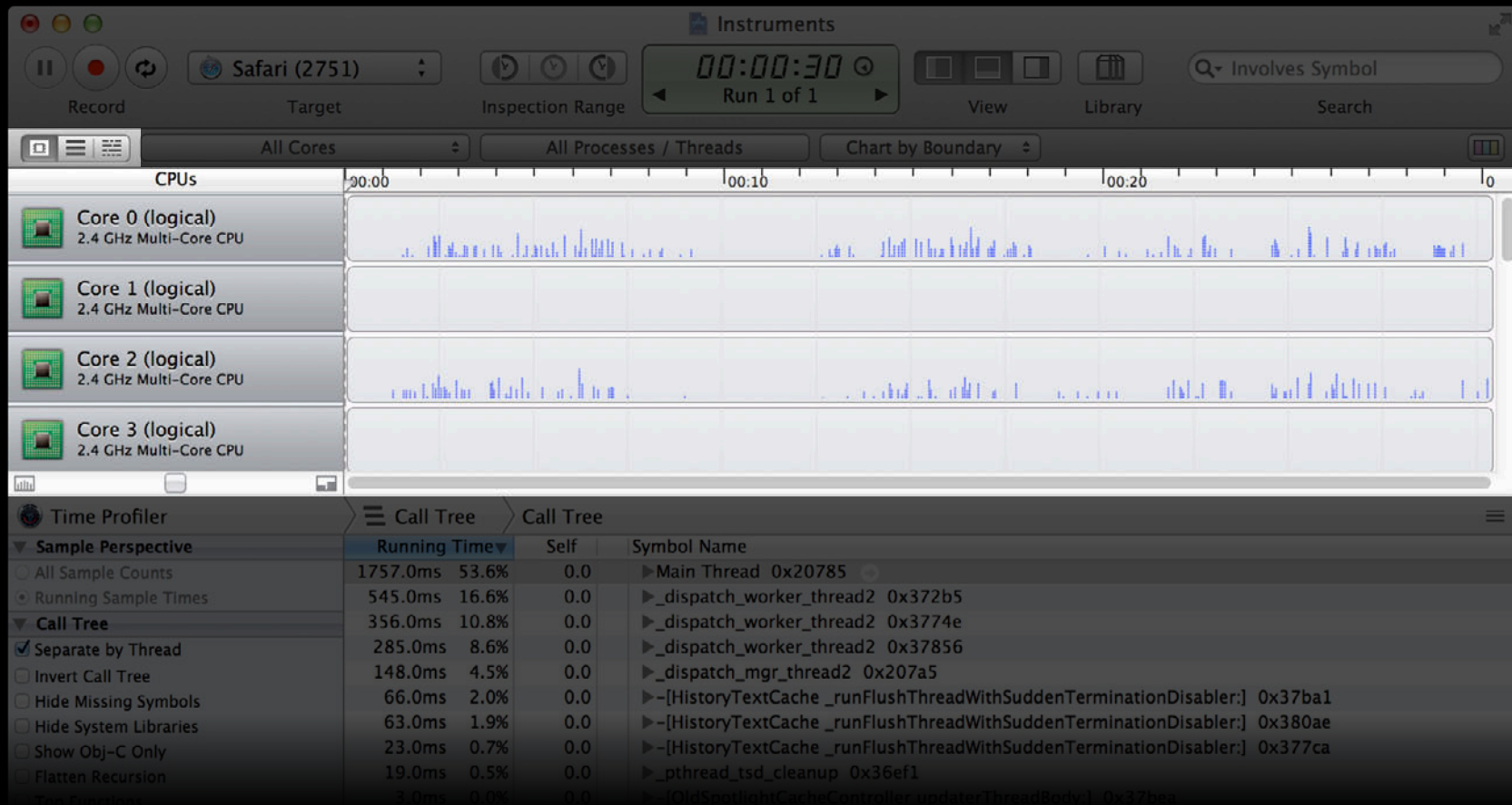
Timeline

Threads strategy



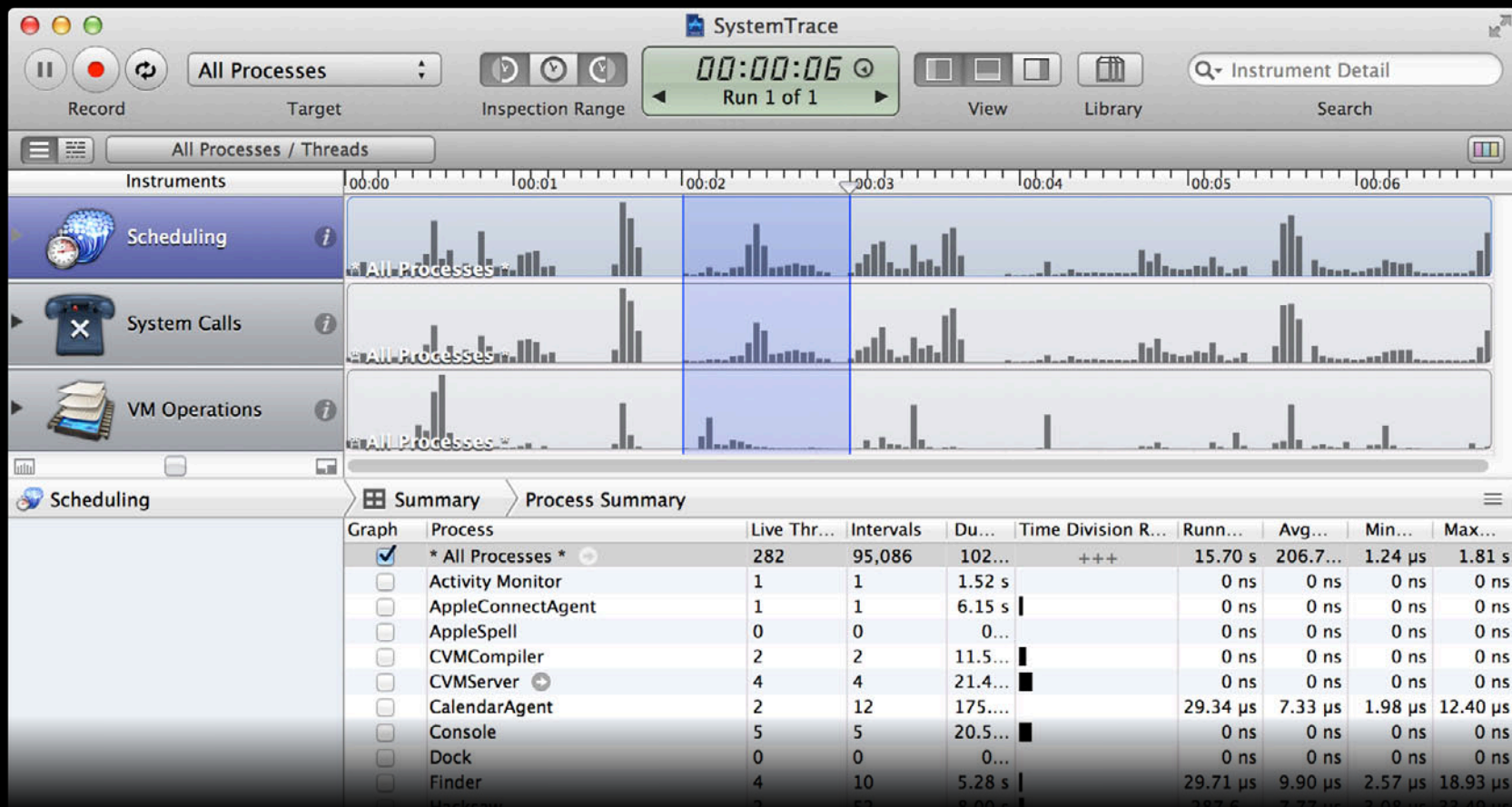
Timeline

CPU strategy



Timeline and Filtering

System Trace



Detail Pane

Call Tree

The screenshot shows the Instruments application interface. At the top, there are controls for recording (Safari (2751)), target, inspection range (00:00:30 Run 1 of 1), and search (Involves Symbol). Below this is a timeline with a purple spike graph. The main area is divided into two panes: the Time Profiler on the left and the Call Tree on the right. The Call Tree pane shows a list of function calls with columns for Running Time, Self, and Symbol Name.

Running Time	Self	Symbol Name
1757.0ms	53.6%	0.0
545.0ms	16.6%	0.0
356.0ms	10.8%	0.0
285.0ms	8.6%	0.0
148.0ms	4.5%	0.0
66.0ms	2.0%	0.0
63.0ms	1.9%	0.0
23.0ms	0.7%	0.0
19.0ms	0.5%	0.0
3.0ms	0.0%	0.0

Detail Pane

Call Tree

The screenshot shows the Instruments application interface. At the top, there are controls for recording (Safari (2751)), target, inspection range (00:00:30 Run 1 of 1), and search (Involves Symbol). Below this is a timeline with a Time Profiler graph. The bottom pane is split into two views: 'Sample Perspective' and 'Call Tree'. The 'Call Tree' view is currently selected and shows a list of function calls with columns for 'Self', 'Symbol Name', and 'Time'. A context menu is open over the 'Call Tree' view, showing options: 'Call Tree' (checked), 'Sample List', and 'Console'.

Self	Symbol Name	Time
0.0	▶ Main Thread 0x20785	0.0
356.0ms	▶ _dispatch_worker_thread2 0x372b5	10.8%
285.0ms	▶ _dispatch_worker_thread2 0x3774e	8.6%
148.0ms	▶ _dispatch_mgr_thread2 0x207a5	4.5%
66.0ms	▶-[HistoryTextCache_runFlushThreadWithSuddenTerminationDisabler:] 0x37ba1	2.0%
63.0ms	▶-[HistoryTextCache_runFlushThreadWithSuddenTerminationDisabler:] 0x380ae	1.9%
23.0ms	▶-[HistoryTextCache_runFlushThreadWithSuddenTerminationDisabler:] 0x377ca	0.7%
19.0ms	▶_pthread_tsd_cleanup 0x36ef1	0.5%
3.0ms	▶-[HistoryTextCacheController_updateThreadBody:] 0x37baa	0.0%

Detail Pane

Sample List

The screenshot shows the Instruments application interface. At the top, the target is 'Safari (2751)' and the run time is '00:00:30 Run 1 of 1'. The 'Time Profiler' instrument is selected, showing a purple flame graph. Below the graph, the 'Sample List' pane is active, displaying a table of call stack samples.

#	Timestamp	Dep	CPU	Thread	Process	Hot Frame	Responsible...	Responsible Caller
0	00:00.779.561	32	6	Main Thread 0...	Safari	tiny_free_list_add_ptr	libsystem_c...	tiny_free_list_add_ptr
1	00:01.135.413	13	2	Main Thread 0...	Safari	objc_msgSend	AppKit	-[NSWindow(NSWind...
2	00:01.138.630	20	6	Main Thread 0...	Safari	CFRelease	AppKit	-[NSWindow sendEv...
3	00:01.139.640	43	6	Main Thread 0...	Safari	argb32_image_mark...	CoreGraphics	argb32_image_mark...
4	00:01.140.648	37	6	Main Thread 0...	Safari	CFRunArrayReplace	AppKit	-[NSWindow sendEv...
5	00:01.141.656	41	6	Main Thread 0...	Safari	magazine_alloc	CoreGraphics	magazine_alloc
6	00:01.143.670	21	6	Main Thread 0...	Safari	objc_msgSend	AppKit	-[NSButtonCell_has...
7	00:01.147.487	36	2	Main Thread 0...	Safari	bzero\$VARIANT\$sse42	libsystem_c...	bzero\$VARIANT\$sse42
8	00:01.148.494	19	2	Main Thread 0...	Safari	mk_timer_arm	libsystem_k...	mk_timer_arm
9	00:01.150.536	25	4	Main Thread 0...	Safari	-[NSArrM_addO...	AppKit	-[NSView(NSInternal...

Detail Pane

Source code

The screenshot displays the Instruments80 application interface. At the top, the target is set to 'Galaxies' and the inspection range is 'Run 1 of 1'. The main window is divided into several panes:

- Instruments:** Shows the 'Time Profiler' instrument active.
- Call Tree:** Displays a call tree for the function 'unsigned int value_texture_for<int>(int const&)' in 'hud.cpp'. The function is selected, and its call tree is visible.
- Source Code:** Shows the source code for the selected function. The code is as follows:

```
125 }  
126  
127 static std::string stringify(unsigned i)  
128 {  
129     char buffer[16];  
130     sprintf(buffer, "%u", i);  
131     return buffer;  
132 }  
133  
134 template <class T>  
135 GLuint value_texture_for(const T &t)  
136 {  
137     std::string text = stringify(t);  
138     GLuint texture = value_textures[text];  
139  
140     if (texture == 0)  
141     {  
142         texture = value_texture(text);  
143         value_textures[text] = texture;  
144     }
```
- Extended Detail:** Shows annotations for the selected code. The annotations are:
 - 75.00% GLuint texture = value_texture...
 - 25.00% sprintf(buffer, "%u", i);

Detail Pane

Source code

The screenshot displays the Xcode Instruments80 interface. At the top, the 'Instruments80' window title is visible. The top bar includes a 'Record' button, a 'Target' dropdown set to 'Galaxies', an 'Inspection Range' showing '00:00:13 Run 1 of 1', and a 'View' button. A search bar on the right contains the text 'Involves Symbol'. Below the top bar, the 'Instruments' section shows a 'Time Profiler' instrument active, with a call tree view selected. The call tree shows the function 'unsigned int value_texture_for<int>(int const&)' selected. The main pane displays the source code for this function in 'hud.cpp'. The code is as follows:

```
125 }
126
127 static std::string stringify(unsigned i)
128 {
129     char buffer[16];
130     sprintf(buffer, "%u", i);
131     return buffer;
132 }
133
134 template <class T>
135 GLuint value_texture_for(const T &t)
136 {
137     std::string text = stringify(t);
138     GLuint texture = value_textures[text];
139
140     if (texture == 0)
141     {
142         texture = value_texture(text);
143         value_textures[text] = texture;
144     }
145 }
```

The 'Extended Detail' pane on the right shows the following annotations:

Annotations
75.00% GLuint texture = value_texture...
25.00% sprintf(buffer, "%u", i);

Detail Pane

Disassembly

The screenshot displays the Instruments application interface. At the top, the target is identified as Safari (2751) and the run time is 00:00:30. The main window is divided into several panes:

- Time Profiler:** Shows a flame chart of CPU usage over time.
- Call Tree:** Displays the call stack. The selected function is `png_write_find_filter` in `libPng.dylib`.
- Disassembly:** Shows the assembly code for the selected function. The instruction at address 757 is highlighted: `+0xa32 testl %eax, %eax` with a time percentage of 37.1%.
- Extended Detail:** Provides a list of instructions and their source code. The selected instruction is `callq DYLD-STUB$$deflate` at address `+0xa2d`, which is the source of the selected assembly instruction.

Address	Instruction	Source	Time %
750	+0xa16 movq -64(%rbp), %rax		
751	+0xa1a jmpq 0xe2		0.2%
752	+0xa1f movl %ebx, 256(%r15)		
753	+0xa26 xorl %ebx, %ebx		
754	+0xa28 movq %r14, %rdi		
755	+0xa2b xorl %esi, %esi		
756	+0xa2d callq DYLD-STUB\$\$deflate		
757	+0xa32 testl %eax, %eax		37.1%
758	+0xa34 je 0x1c0		
759	+0xa3a movq 296(%r15), %rsi		
760	+0xa41 testq %rsi, %rsi		

Detail Pane

System Calls

The screenshot displays the Instruments80 application interface. At the top, the target is set to 'Galaxies' and the recording status is 'Run 1 of 1' with a duration of 00:00:16. The main area shows a threads timeline with several threads from the Galaxies process. The 'System Calls' pane is active, showing a call tree with the following data:

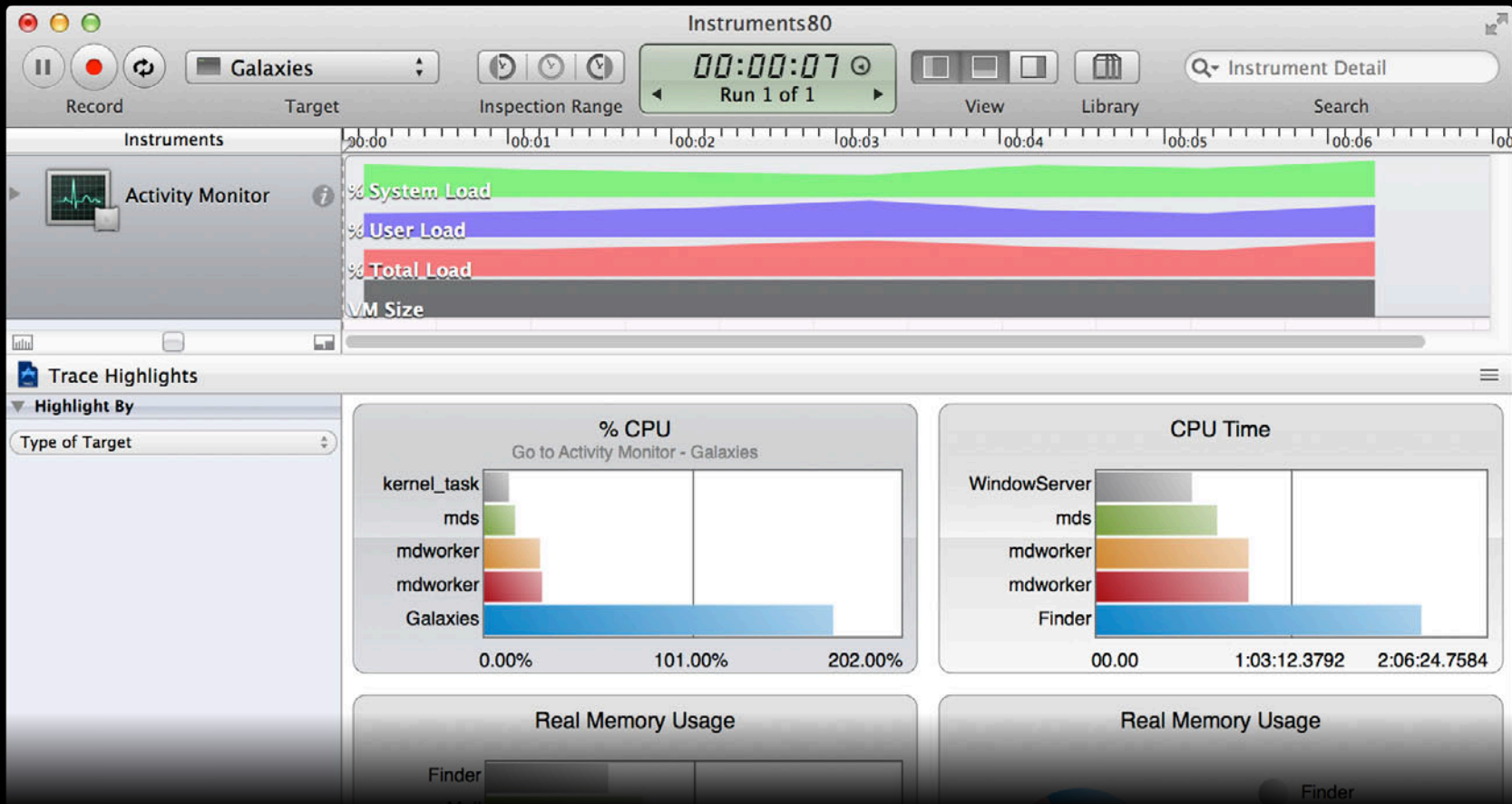
Symbol Name	Calls	Percentage
mk_timer_arm libsystem_kernel.dylib	275808	57.0%
CFRunLoopAddTimer CoreFoundation	91365	18.9%
_CFSetApplyFunction_block_invoke_0 CoreFoundation	90079	18.6%
_CFRunLoopRun CoreFoundation	60670	12.5%
_CFRunLoopDoTimer CoreFoundation	31203	6.4%
CFRunLoopTimerInvalidate CoreFoundation	1269	0.2%
CFRunLoopTimerSetNextFireDate CoreFoundation	1222	0.2%
mach_msg_trap libsystem_kernel.dylib	101917	21.0%
_kernelrpc_mach_port_insert_member_trap libsystem_kernel.dylib	49276	10.1%
_kernelrpc_mach_port_extract_member_trap libsystem_kernel.dylib	49225	10.1%

The 'Extended Detail' pane on the right shows the 'Heaviest Stack Trace' for the selected call:

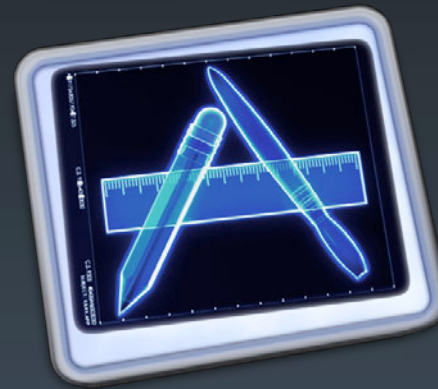
- 275808 mk_timer_arm
- 91365 CFRunLoopAddTimer
- 91360.0 _CFSetApplyFunction_block_invoke_0
- 91360.0 CFBasicHashApply
- 91360.0 CFSetApplyFunction
- 91360.0 CFRunLoopAddTimer
- 90079.0 -[NSObject(NSDelayedPer...)
- 90079.0 +[_NSAutomaticFocusRin...]
- 90079.0 -[NSView setNeedsDispla...]
- 90079.0 -[NSView setNeedsDisplay:]
- 90079.0 _NSFireTimer
- 90079.0 _CFRUNLOOP_IS_CALLIN...
- 90079.0 _CFRunLoopDoTimer
- 90079.0 _CFRunLoopRun
- 90079.0 CFRunLoopRunSpecific
- 90079.0 RunCurrentEventLoopInMode
- 90079.0 ReceiveNextEventCommon
- 90079.0 BlockUntilNextEventMatc...
- 90079.0 _DPSNextEvent
- 90079.0 -[NSApplication nextEven...

Detail Pane

Activity summary



Instruments



Demo

Time Profiler example

Joe Grzywacz

Performance Tools Engineering

Use Call Tree

The screenshot shows the Xcode Instruments interface for a target named 'App Ocean'. The top bar includes controls for recording, target selection, inspection range (00:02:00, Run 2 of 2), and search filters. The main area displays a Time Profiler instrument with a purple waveform graph. Below the graph, the 'Call Tree' view is active, showing a hierarchical list of function calls with columns for Running Time, Self, and Symbol Name.

Running Time	Self	Symbol Name
61670.0ms 92.5%	0.0	CA::Layer::layout_and_display_if_needed(CA::Transaction*) QuartzCore
61620.0ms 92.4%	0.0	CA::Layer::display_if_needed(CA::Transaction*) QuartzCore
61590.0ms 92.4%	10.0	CA::Layer::display_0 QuartzCore
61490.0ms 92.2%	20.0	CABackingStoreUpdate_ QuartzCore
60750.0ms 91.1%	0.0	-[CALayer drawInContext:] QuartzCore
60750.0ms 91.1%	0.0	-[UIView(CALayerDelegate) drawLayer:inContext:] UIKit
36480.0ms 54.7%	0.0	-[OceanView drawIcons] App Ocean
23190.0ms 34.8%	0.0	-[OceanView drawRect:] App Ocean
1040.0ms 1.5%	0.0	UIRectFill UIKit
10.0ms 0.0%	10.0	objc_msgSend libobjc.A.dylib
10.0ms 0.0%	0.0	PushContext UIKit
10.0ms 0.0%	10.0	objc_msgSend\$shim App Ocean
10.0ms 0.0%	10.0	-[AppParent draw] App Ocean
0.0ms 0.0%	0.0	CGContextGetClipBoundingBox CoreGraphics
530.0ms 0.7%	0.0	-[CATextLayer drawInContext:] QuartzCore

Use Call Tree

Running Time	Self	Symbol Name
61670.0ms 92.5%	0.0	▼CA::Layer::layout_and_display_if_needed(CA::Transaction*) QuartzCore
61620.0ms 92.4%	0.0	▼CA::Layer::display_if_needed(CA::Transaction*) QuartzCore
61590.0ms 92.4%	10.0	▼CA::Layer::display_0 QuartzCore
61490.0ms 92.2%	20.0	▼CABackingStoreUpdate_ QuartzCore
60750.0ms 91.1%	0.0	▼-[CALayer drawInContext:] QuartzCore
60750.0ms 91.1%	0.0	▼-[UIView(CALayerDelegate) drawLayer:inContext:] UIKit
36480.0ms 54.7%	0.0	▶-[OceanView drawIcons] App Ocean
23190.0ms 34.8%	0.0	▶-[OceanView drawRect:] App Ocean
1040.0ms 1.5%	0.0	▶UIRectFill UIKit
10.0ms 0.0%	10.0	objc_msgSend libobjc.A.dylib
10.0ms 0.0%	0.0	▶PushContext UIKit
10.0ms 0.0%	10.0	objc_msgSend\$shim App Ocean
10.0ms 0.0%	10.0	-[AppParent draw] App Ocean
0.0ms 0.0%	0.0	CGContextGetClipBoundingBox CoreGraphics
530.0ms 0.7%	0.0	▶-[CATextLayer drawInContext:] QuartzCore

Use Thread Strategy

The screenshot shows the Instruments application window titled "Instruments_demo1_1". The interface includes a top toolbar with "Record", "Target" (App Ocean), "Inspection Range" (00:02:00 Run 2 of 2), "View", "Library", and "Search" (Involves Symbol). Below the toolbar are filters for "All Cores", "All Processes / Threads", and "All Events".

The main area displays a "Threads" list on the left and a timeline on the right. The threads listed are:

- App Ocean (1599) Main Thread 0x3c7ff
- App Ocean (1599) _dispatch_worker_thread2 0x3c806
- App Ocean (1599) _dispatch_mgr_thread\$VARIANT\$mp...
- App Ocean (1599) _pthread_wqthread 0x3c808

A "Sample" window is open, titled "Idle Call Stack Sample", with a timestamp of 00:35.525.763 and CPU ID "Not running on CPU". It shows a call stack with the following entries:

- mach_msg_trap
- __CFRunLoopServiceMachPort
- __CFRunLoopRun
- CFRunLoopRunSpecific
- CFRunLoopRunInMode
- CFURLConnectionSendSynchronousRequest
- +[NSURLConnection sendSynchronousRe...
- [IconDownloader startDownload]
- [OceanView startIconDownload:]
- [OceanView timerFired]
- __NSFireTimer
- __CFRUNLOOP_IS_CALLING_OUT_TO_A_T...

At the bottom, a "Time Profiler" window is open, showing a "Call Tree" view. The table below summarizes the call tree data:

Sample Perspective	Running Time	Self	S
All Sample Counts	65060.0ms	97.6%	0.0
Running Sample Times	340.0ms	0.5%	0.0
Call Tree	340.0ms	0.5%	0.0
Separate by Thread	280.0ms	0.4%	0.0
Invert Call Tree	220.0ms	0.3%	0.0
Hide Missing Symbols	110.0ms	0.1%	0.0
Hide System Libraries	100.0ms	0.1%	0.0
Show Obj-C Only	70.0ms	0.1%	0.0
Flatten Recursion	70.0ms	0.1%	0.0
Top Functions	40.0ms	0.0%	0.0

Use Thread Strategy

The screenshot shows the Xcode Instruments interface for a target named "App Ocean". The top bar includes a timer at 00:02:00 (Run 2 of 2) and a search filter "Involves Symbol". The "Threads" pane on the left lists several threads, with the main thread selected. A "Sample" popup window is open, displaying an "Idle Call Stack Sample" taken at timestamp 00:35.525.763 on a CPU that is not running. The call stack is shown in two panes: "Call Tree" and "Call Tree".

Call Tree (Top):

- +[NSURLConnection sendSynchronousRe...
- [IconDownloader startDownload]

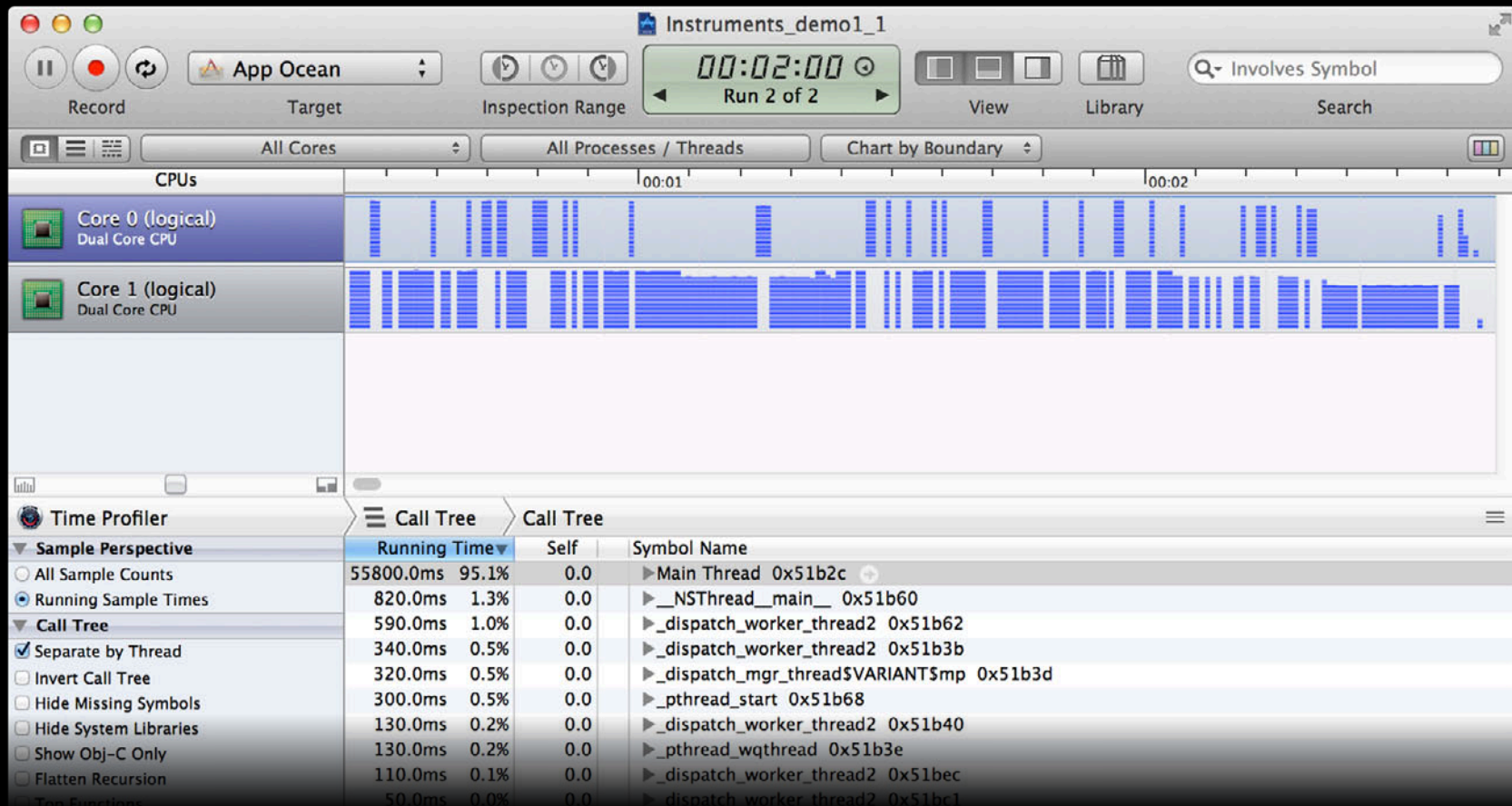
Call Tree (Bottom):

- _CFRunLoopRun
- CFRunLoopRunSpecific
- CFRunLoopRunInMode
- CFURLConnectionSendSynchronousRequest
- +[NSURLConnection sendSynchronousRe...
- [IconDownloader startDownload]
- [OceanView startIconDownload:]
- [OceanView timerFired]
- _NSFireTimer
- _CFRUNLOOP_IS_CALLING_OUT_TO_A.T...

Time Profiler (Bottom Left):

Sample Perspective	Running Time	Self	S
All Sample Counts	65060.0ms	97.6%	0.0
Running Sample Times	340.0ms	0.5%	0.0
Call Tree	340.0ms	0.5%	0.0
Separate by Thread	280.0ms	0.4%	0.0
Invert Call Tree	220.0ms	0.3%	0.0
Hide Missing Symbols	110.0ms	0.1%	0.0
Hide System Libraries	100.0ms	0.1%	0.0
Show Obj-C Only	70.0ms	0.1%	0.0
Flatten Recursion	70.0ms	0.1%	0.0
Top Functions	40.0ms	0.0%	0.0

Use CPU Strategy



Demo

Memory profiling example

Victor Hernandez

Performance Tools Engineering

Use Cycles and Roots

The screenshot displays the Instruments5 application window. At the top, there's a control bar with a 'Record' button, a 'Target' dropdown set to 'App Ocean Mem...', an 'Inspection Range' section with a timer at '00:01:02' and 'Run 1 of 1', and a search field for 'Instrument Detail'. Below this are two graphs: 'Allocations' (a blue area chart showing memory usage over time) and 'Leaks' (a light blue area chart with three red vertical bars indicating leak events). The bottom section is titled 'Leak Cycles' and contains a table with the following data:

#	Type	Details
Cycles (51)		
1	AppParent - 4 nodes	Simple Cycle
2	AppParent - 4 nodes	Simple Cycle
3	AppParent - 4 nodes	Simple Cycle
4	AppParent - 4 nodes	Simple Cycle
5	AppParent - 4 nodes	Simple Cycle
6	AppParent - 4 nodes	Simple Cycle
7	AppParent - 4 nodes	Simple Cycle
8	AppParent - 4 nodes	Simple Cycle

To the right of the table is a 'Graph' section showing a cycle between two objects: 'AppParent' and 'NSMutableArray'. A red arrow points from 'AppParent' to 'NSMutableArray' (labeled 'NSMutableArray* _children'), and another red arrow points from 'NSMutableArray' back to 'AppParent', forming a cycle.

Use Allocations

The screenshot shows the Instruments5 application window. At the top, there are controls for recording, target selection (App Ocean Mem...), inspection range (00:00:58, Run 1 of 1), and search (OceanView). The main area displays a graph of memory usage over time, with a blue area representing memory usage that increases over the 58-second run. Below the graph, there are sections for Allocations and Leaks. At the bottom, the Object Summary table is visible, showing details for OceanViewController and OceanView.

Graph	Category	Live Bytes	# Living	# Transitory	Overall Bytes	# Overall	# Allocations (N...
<input type="checkbox"/>	OceanViewController	208 Bytes	1	2	624 Bytes	3	
<input checked="" type="checkbox"/>	OceanView	576 Bytes	3	0	576 Bytes	3	

Use Allocations

The screenshot shows the Instruments5 application window. The top bar includes a 'Record' button, a target selection dropdown set to 'App Ocean Mem...', an 'Inspection Range' timer at '00:00:58', and a search field containing 'OceanView'. The main area is divided into two panes: 'Allocations' and 'Leaks'. The 'Allocations' pane shows a blue area graph representing memory usage over time. Below the graph, a blue box highlights the number '3'. The bottom pane is titled 'Object Summary' and contains a table with the following data:

Graph	Category	Live Bytes	# Living	# Transitory	Overall Bytes	# Overall	# Allocations (N...
<input type="checkbox"/>	OceanViewController	208 Bytes	1	2	624 Bytes	3	
<input checked="" type="checkbox"/>	OceanView	576 Bytes	3	0	576 Bytes	3	

Closing Thoughts...

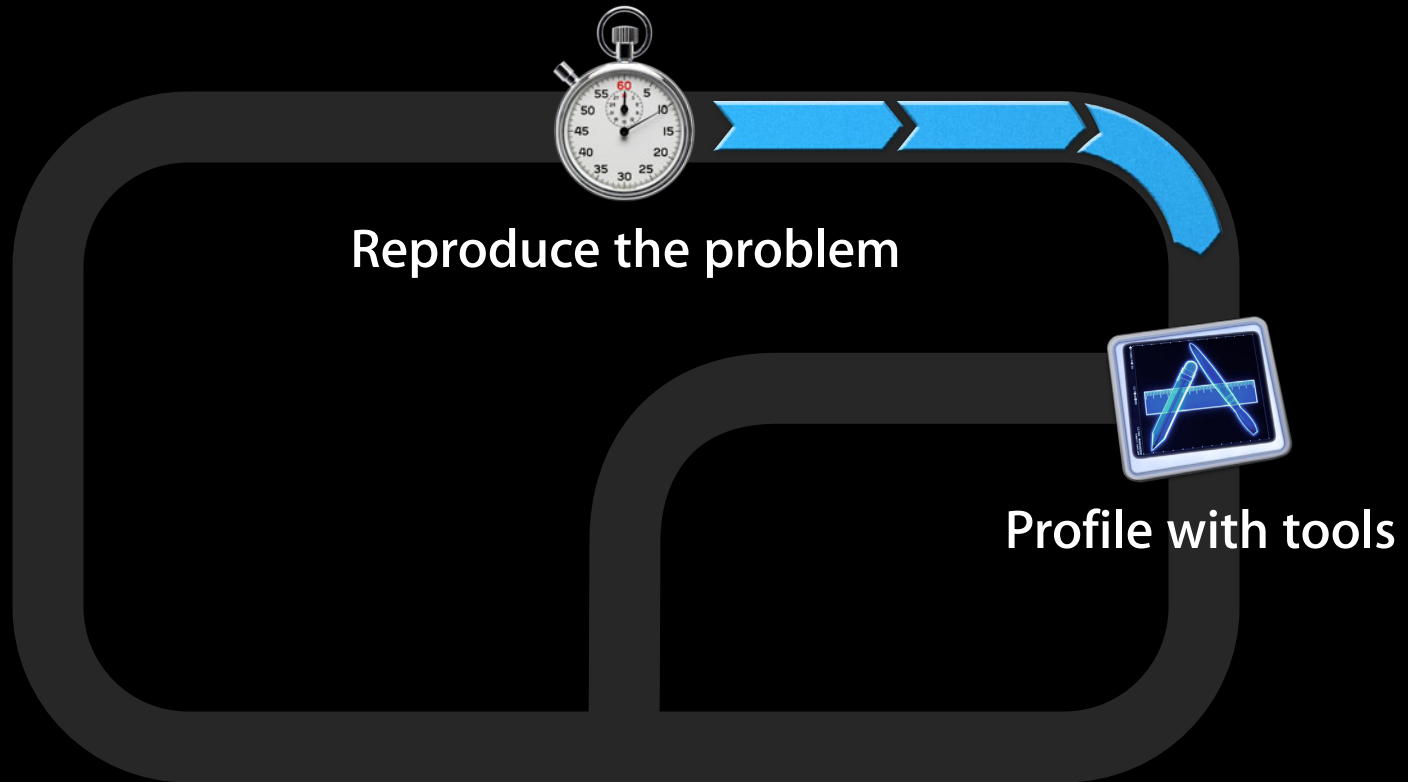
Profile Process

Profile Process

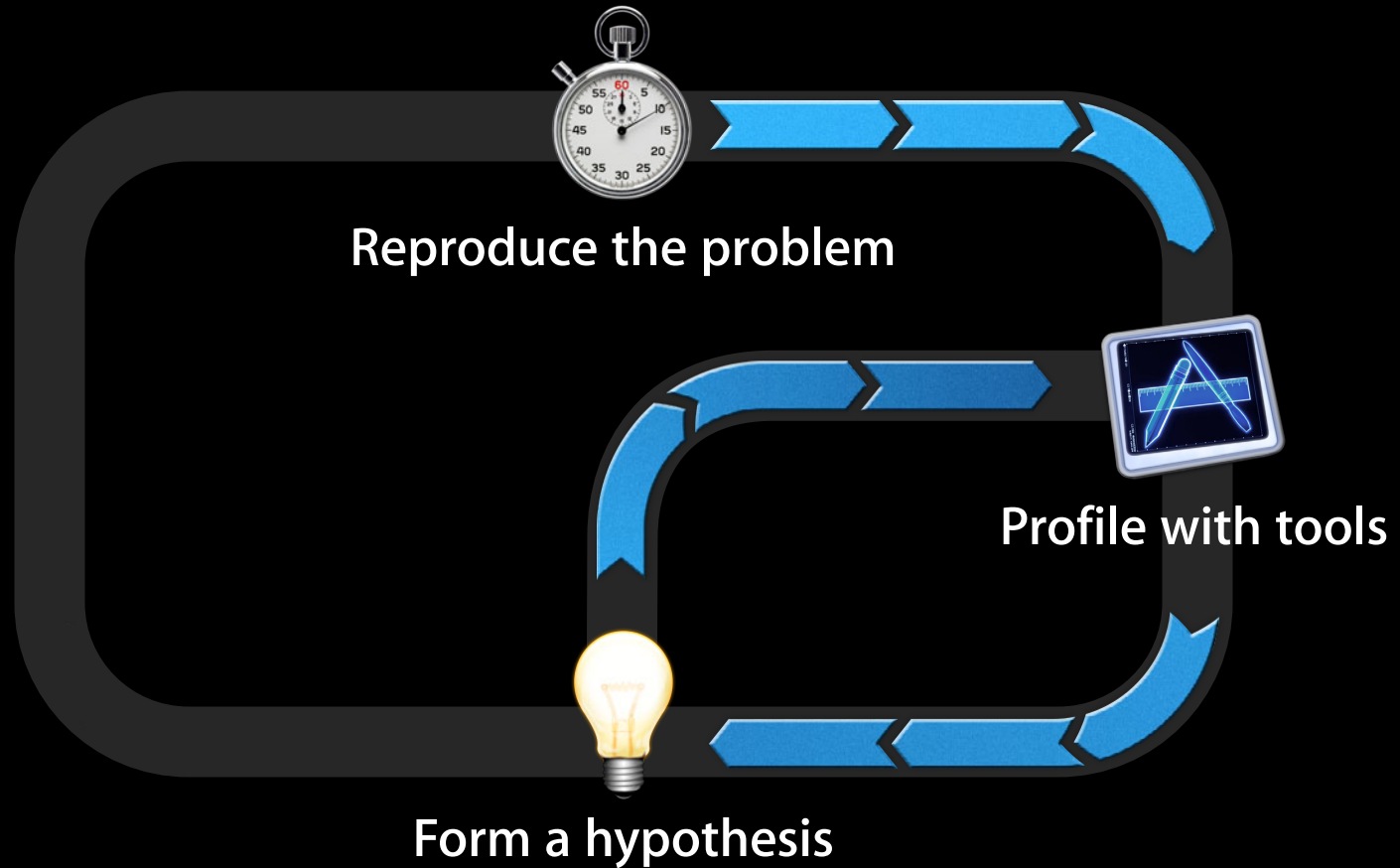


Reproduce the problem

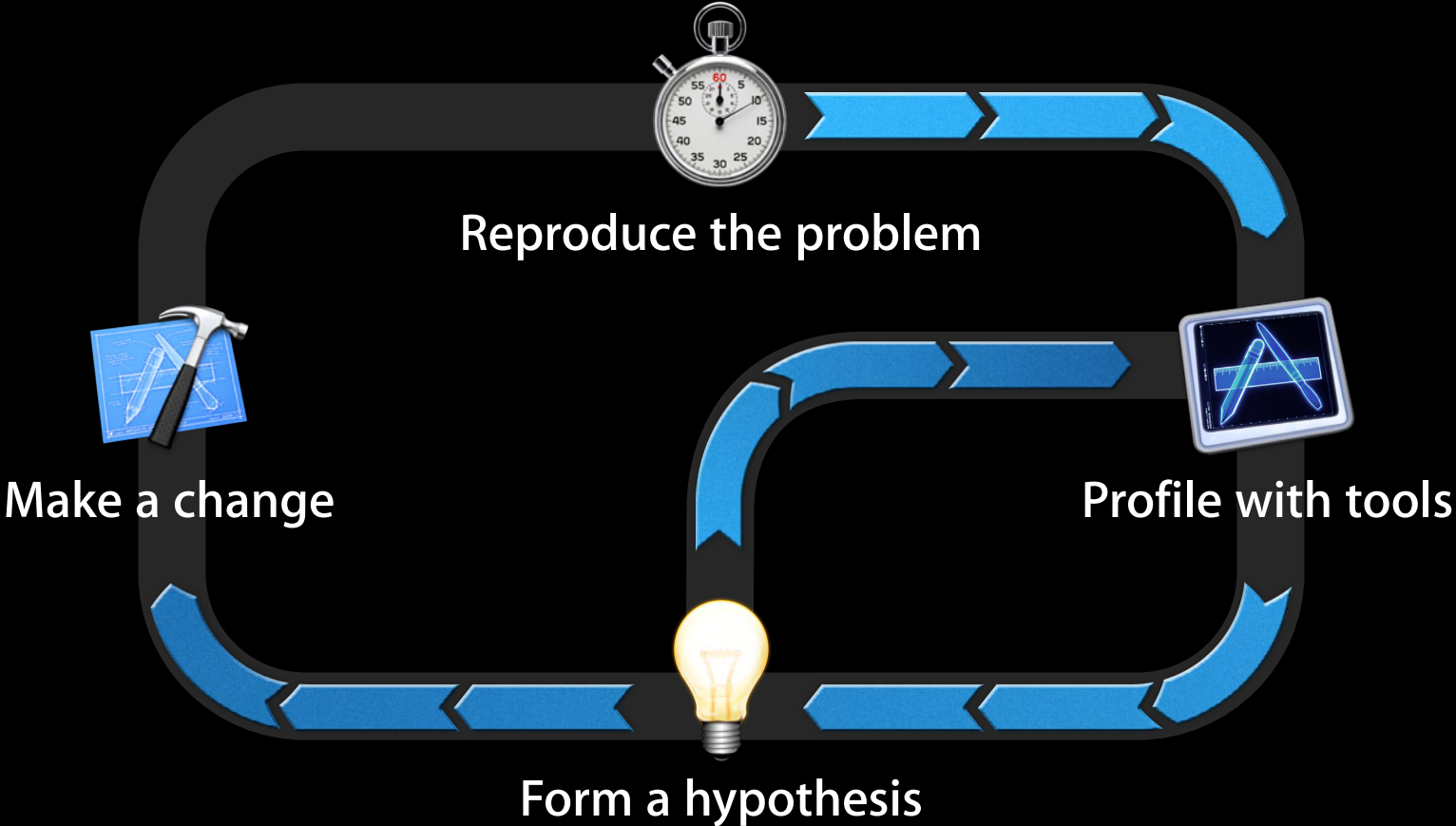
Profile Process



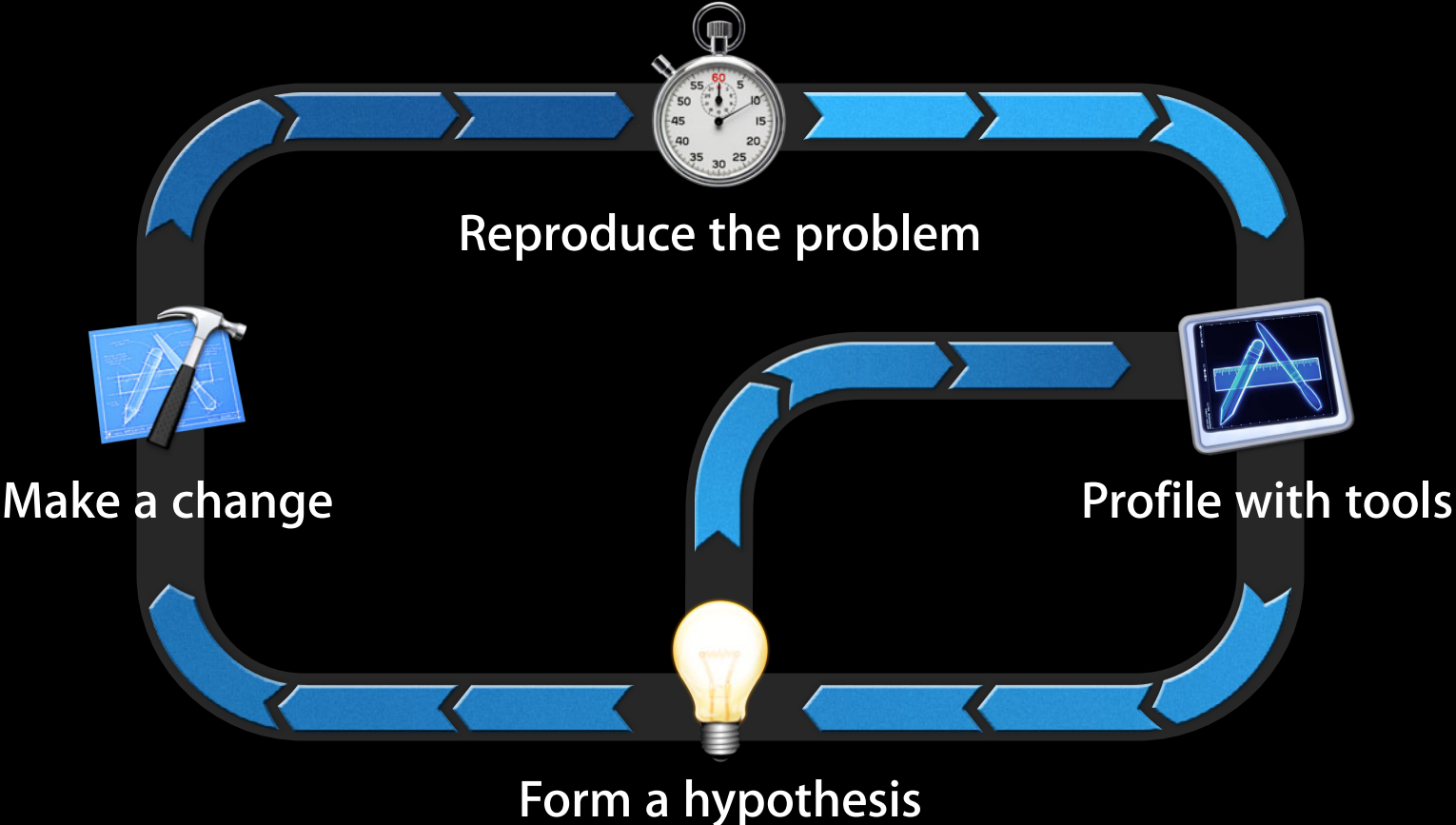
Profile Process



Profile Process



Profile Process



Instruments Templates Used

Instruments Templates Used



3 Improvements



2 Improvements

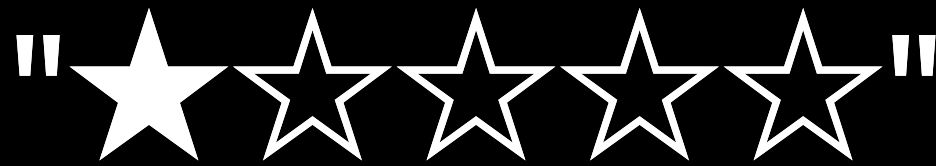
Can Profile Virtually Anything

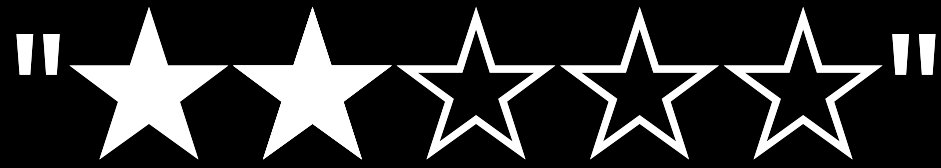
Can Profile Virtually Anything



"Fantastic and Fast"

"Must buy this app"







More Information

Michael Jurewitz

Developer Tools Evangelist

jury@apple.com

Instruments Documentation

[Instruments User Guide \(Xcode Documentation\)](#)

[Instruments New Features User Guide](#)

Apple Developer Forums

<http://devforums.apple.com>

Related Sessions

iOS App Performance: Responsiveness

Presidio
Thursday 11:30AM

iOS App Performance: Graphics and Animations

Presidio
Thursday 3:15PM

iOS App Performance: Memory

Presidio
Thursday 4:30PM

Labs

Xcode Lab

Developer Tools Lab B
Ongoing

 WWDC2012

