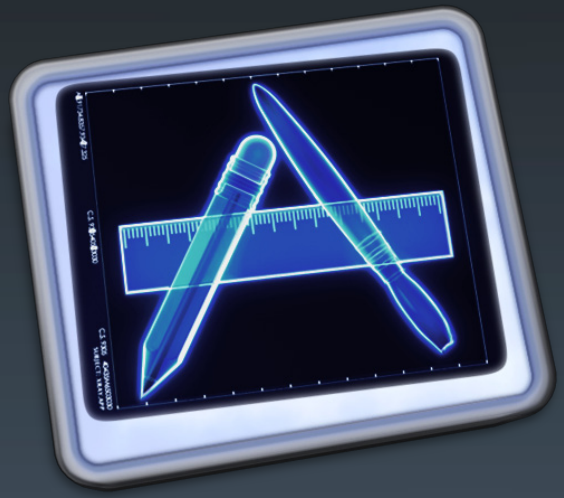
What's New in Instruments



Session 310

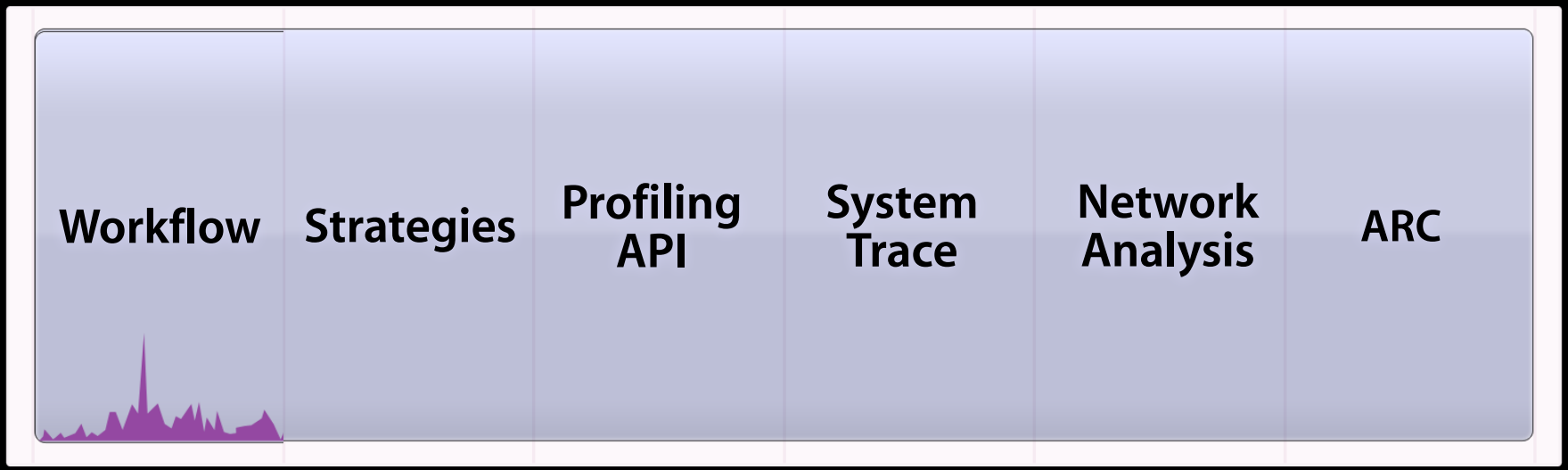
Steve Lewallen
Performance Tools Engineering Manager

Today's Agenda

Workflow	Strategies	Profiling API	System Trace	Network Analysis	ARC

Workflow Improvements

From recording to data mining

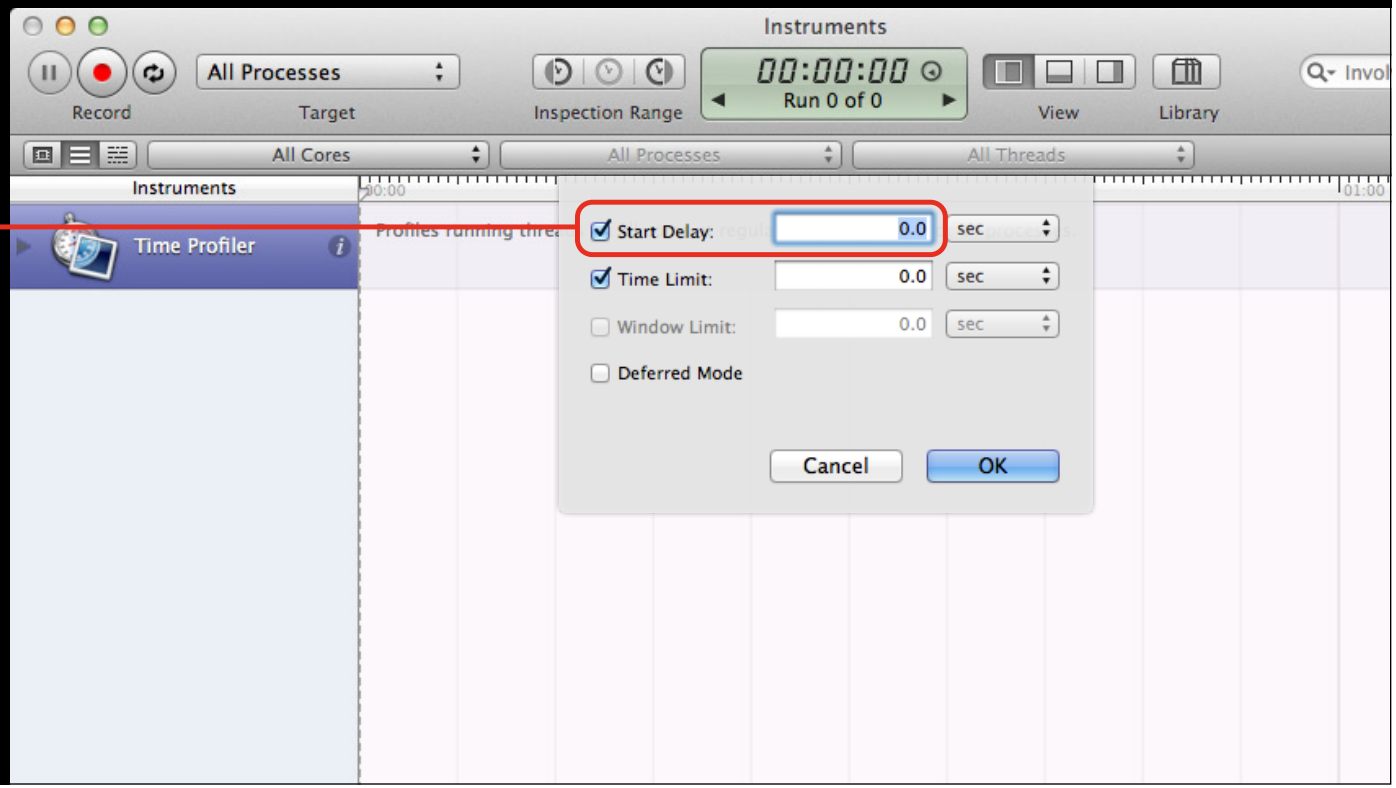


Record Options

Record Options

Refining recording behavior

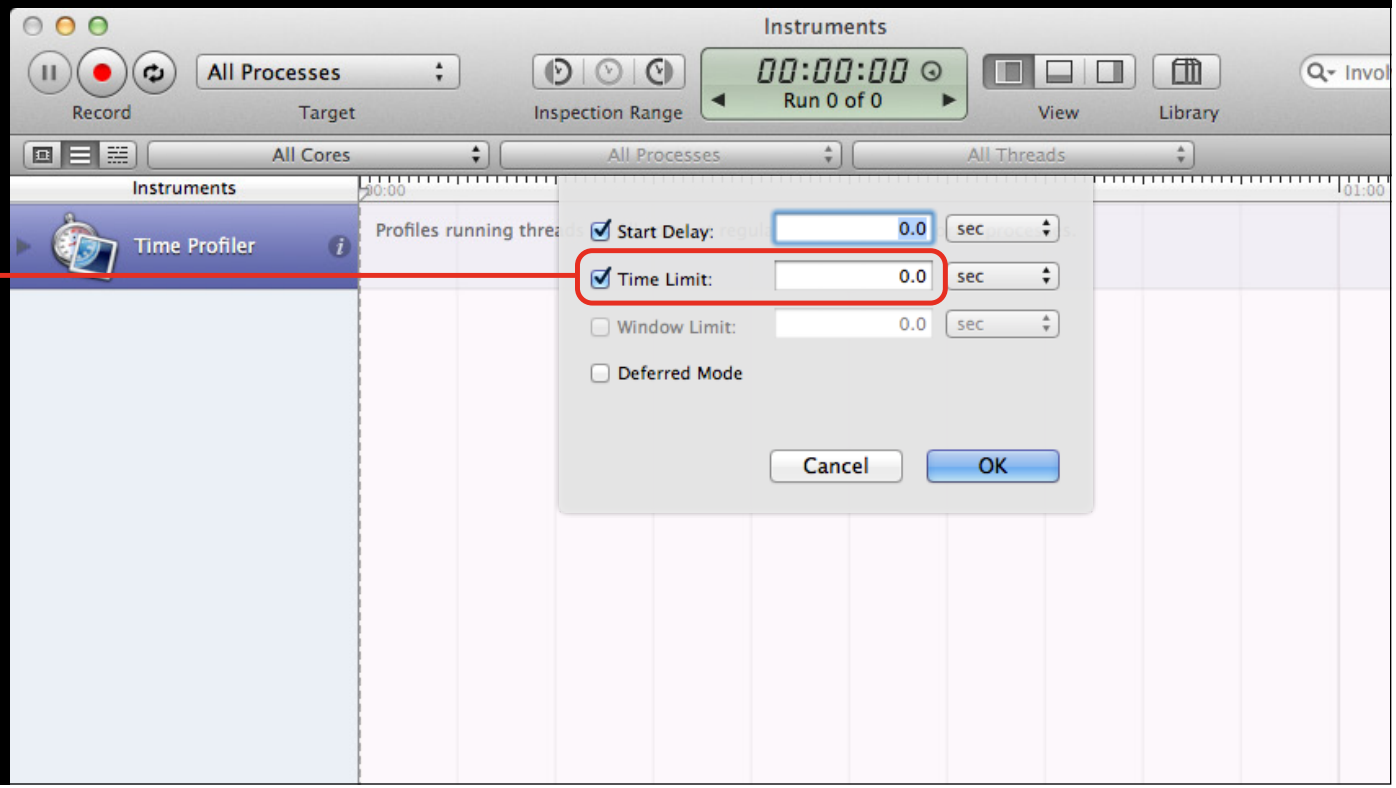
Start Delay
Add delay between hitting record and record starting



Record Options

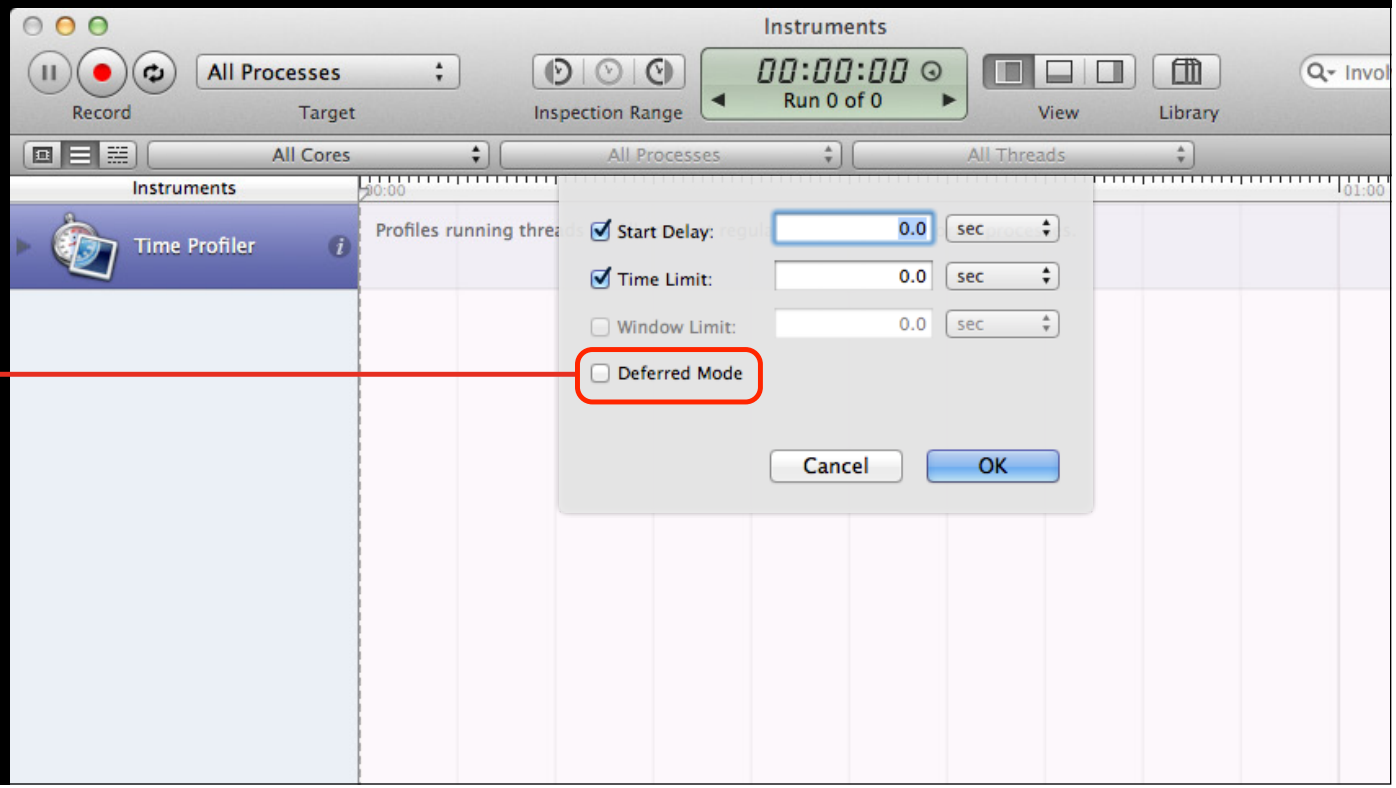
Refining recording behavior

Time Limit
Set a maximum recording length



Record Options

Refining recording behavior

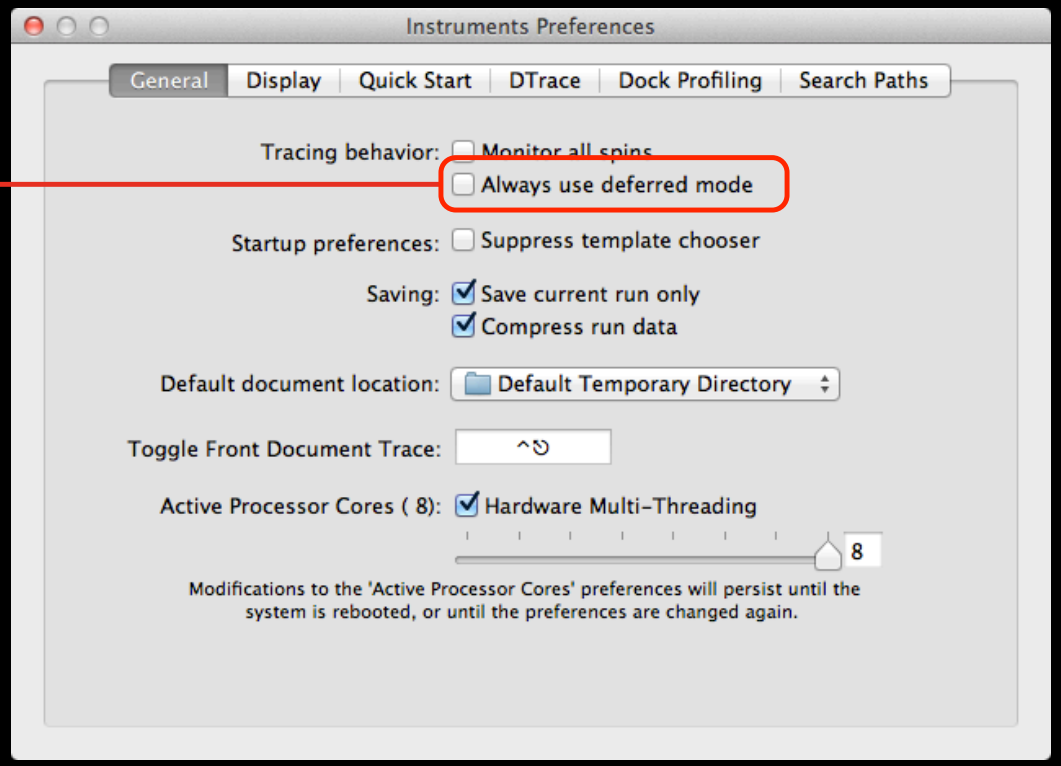


Deferred Mode
Toggle between "immediate" and deferred mode

Record Options

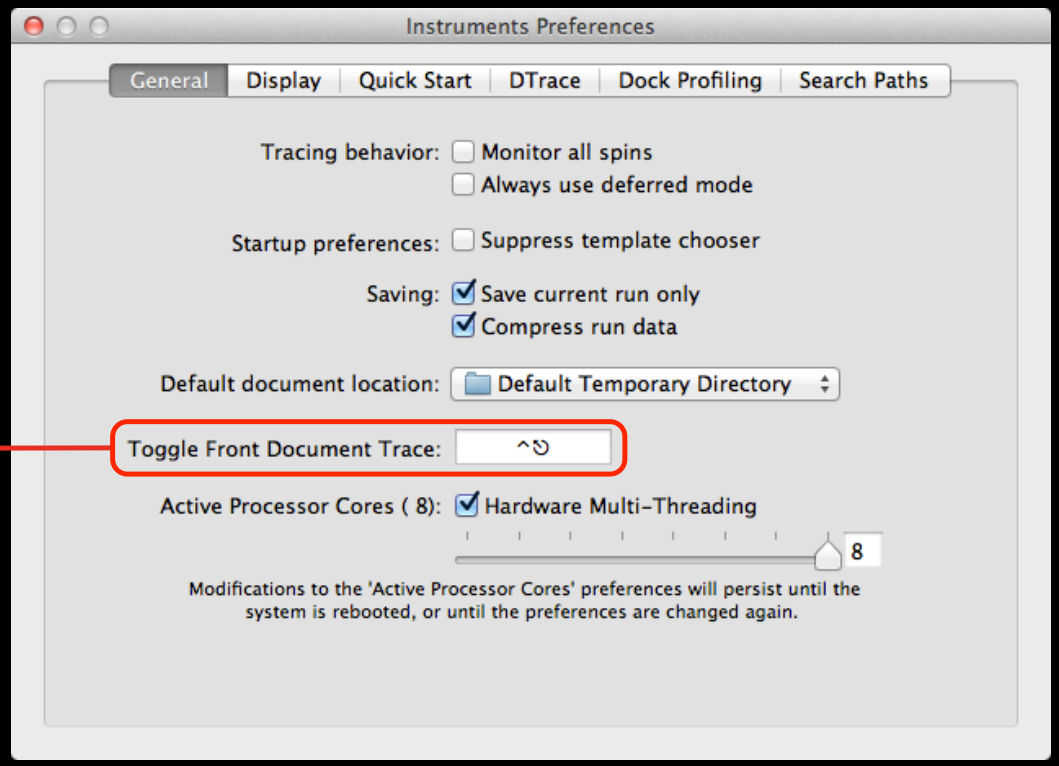
Refining recording behavior

Deferred Mode
Set deferred mode as the defaults for all new recordings



Record Options

Refining recording behavior

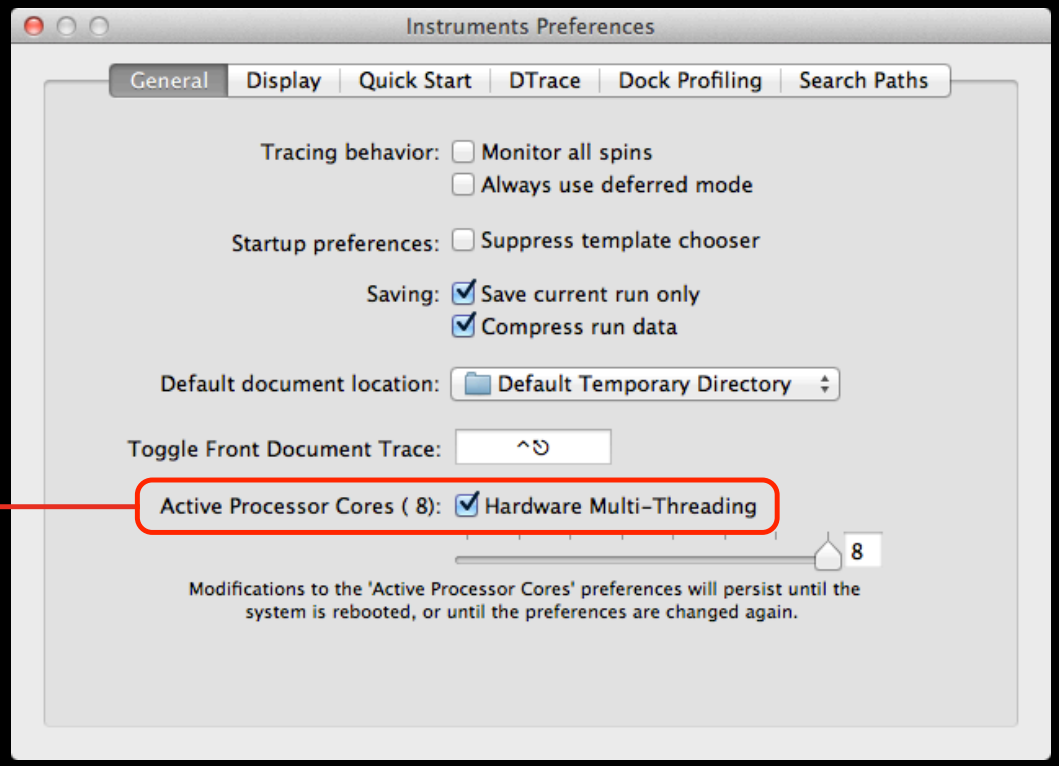


Toggle Recording
Start/Stop recording of front-most trace document

Instruments may be in background

Record Options

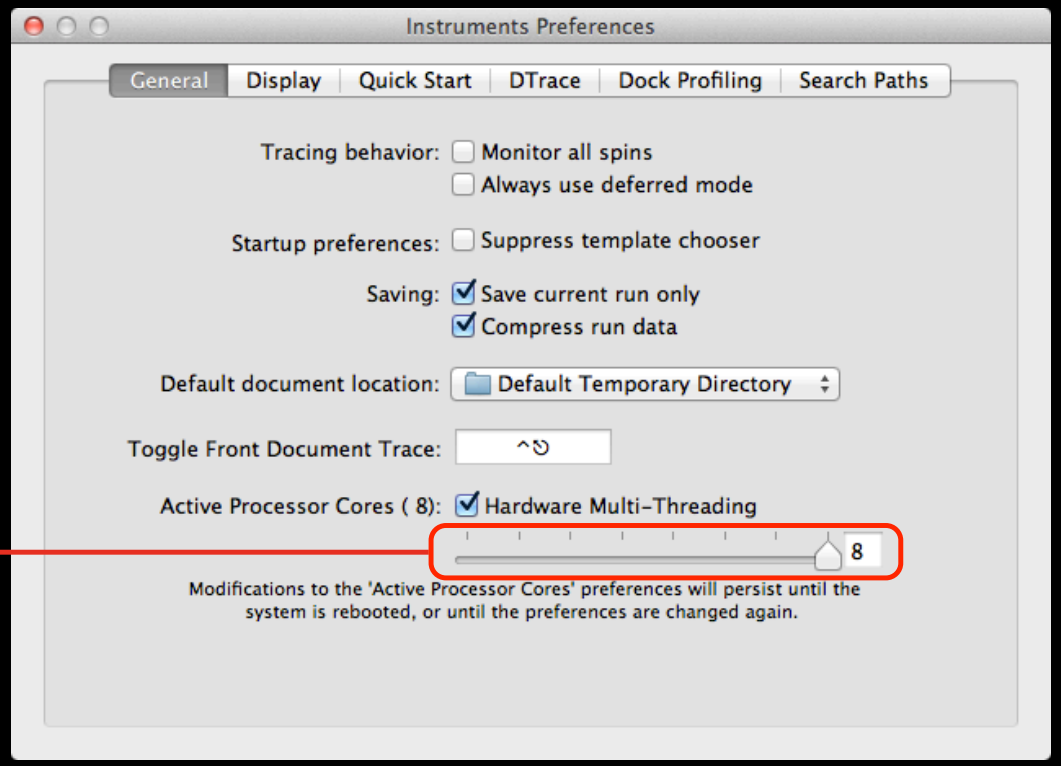
Refining recording behavior



Toggle Hardware Threading
Turns hyper-threading on or off for supported systems

Record Options

Refining recording behavior

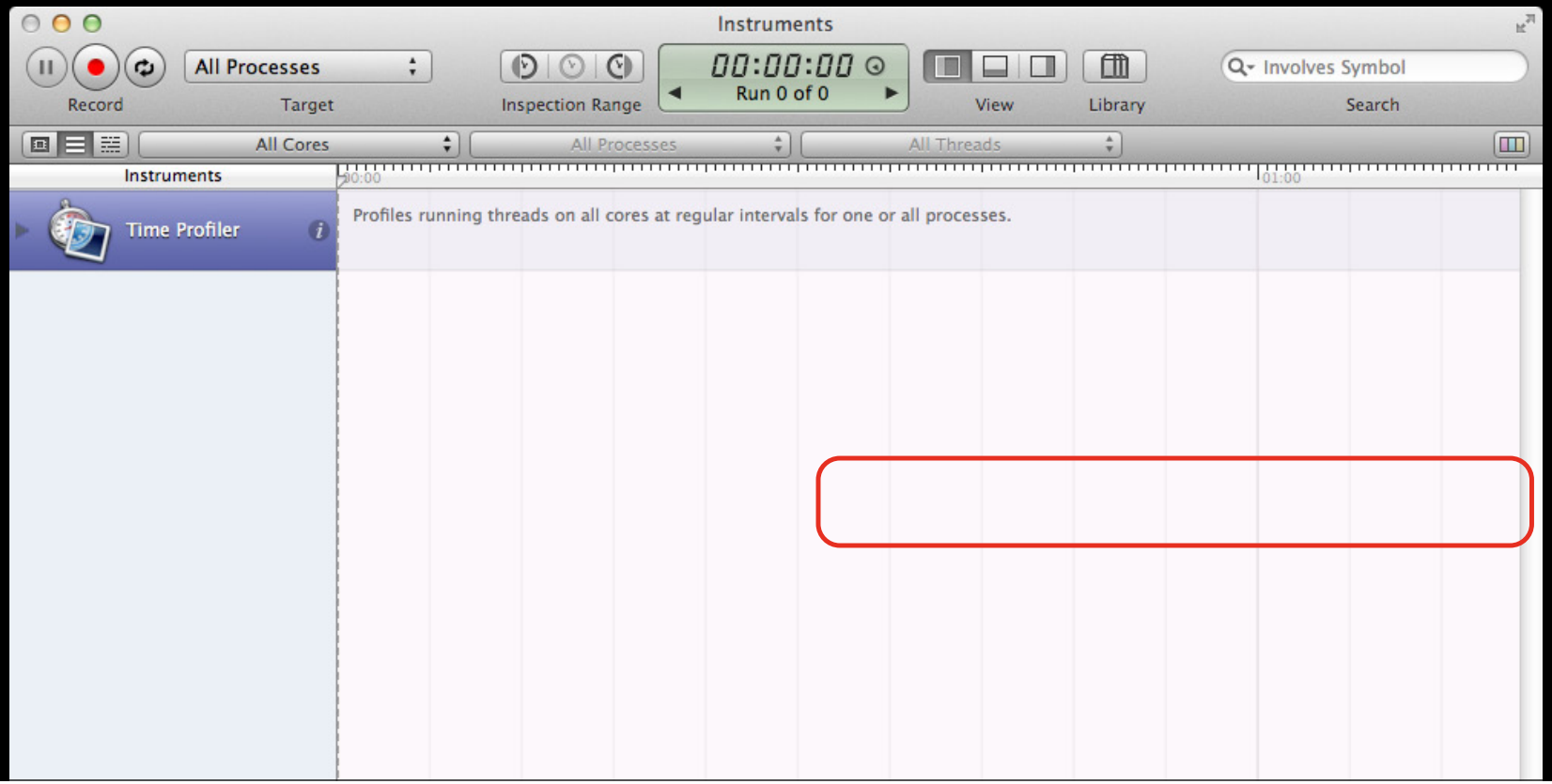


Active Processor Cores
Reduce the number of active cores on the Mac OS X system

Track Gestures

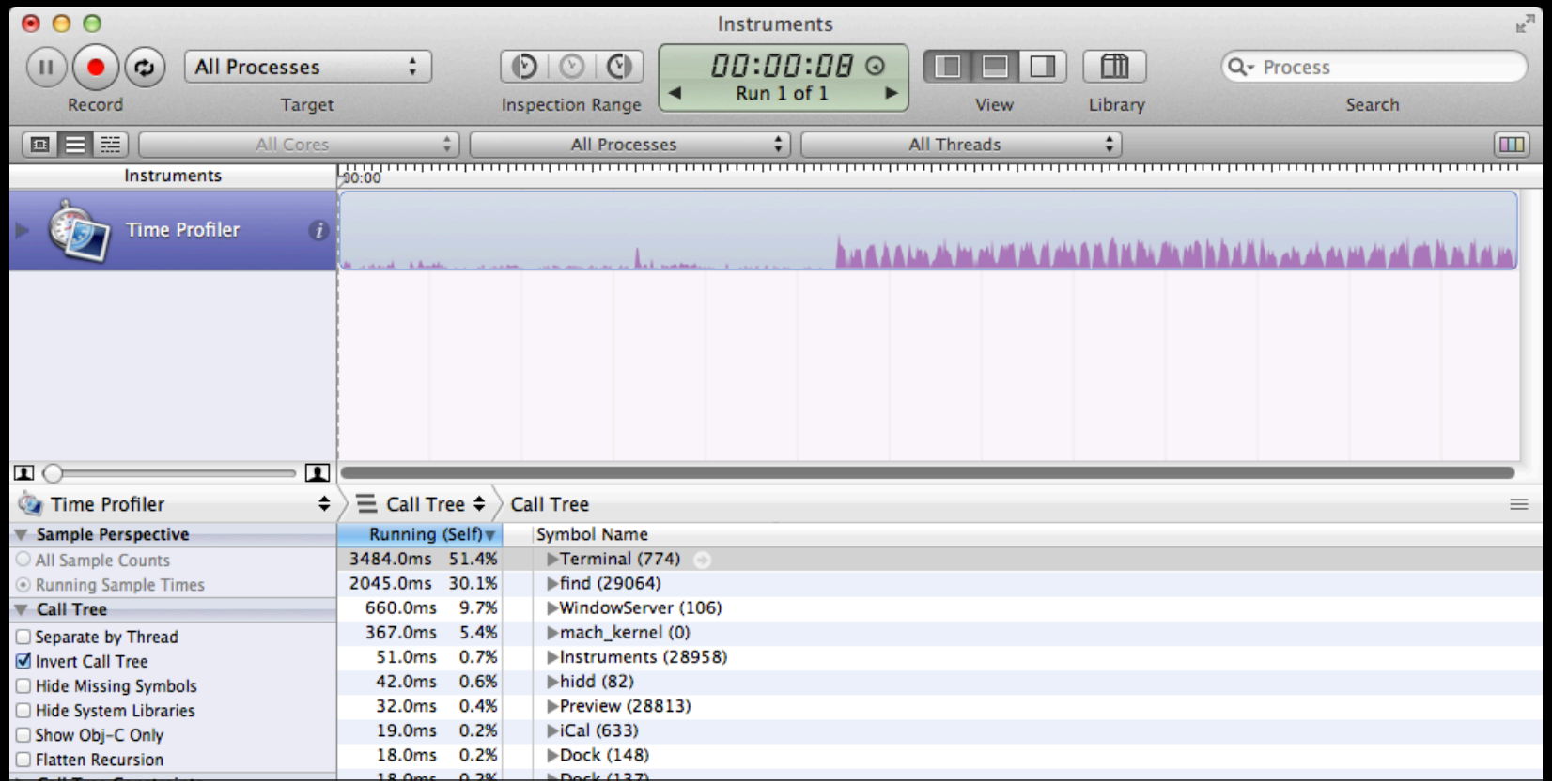
Track Gestures

Working with high-resolution traces



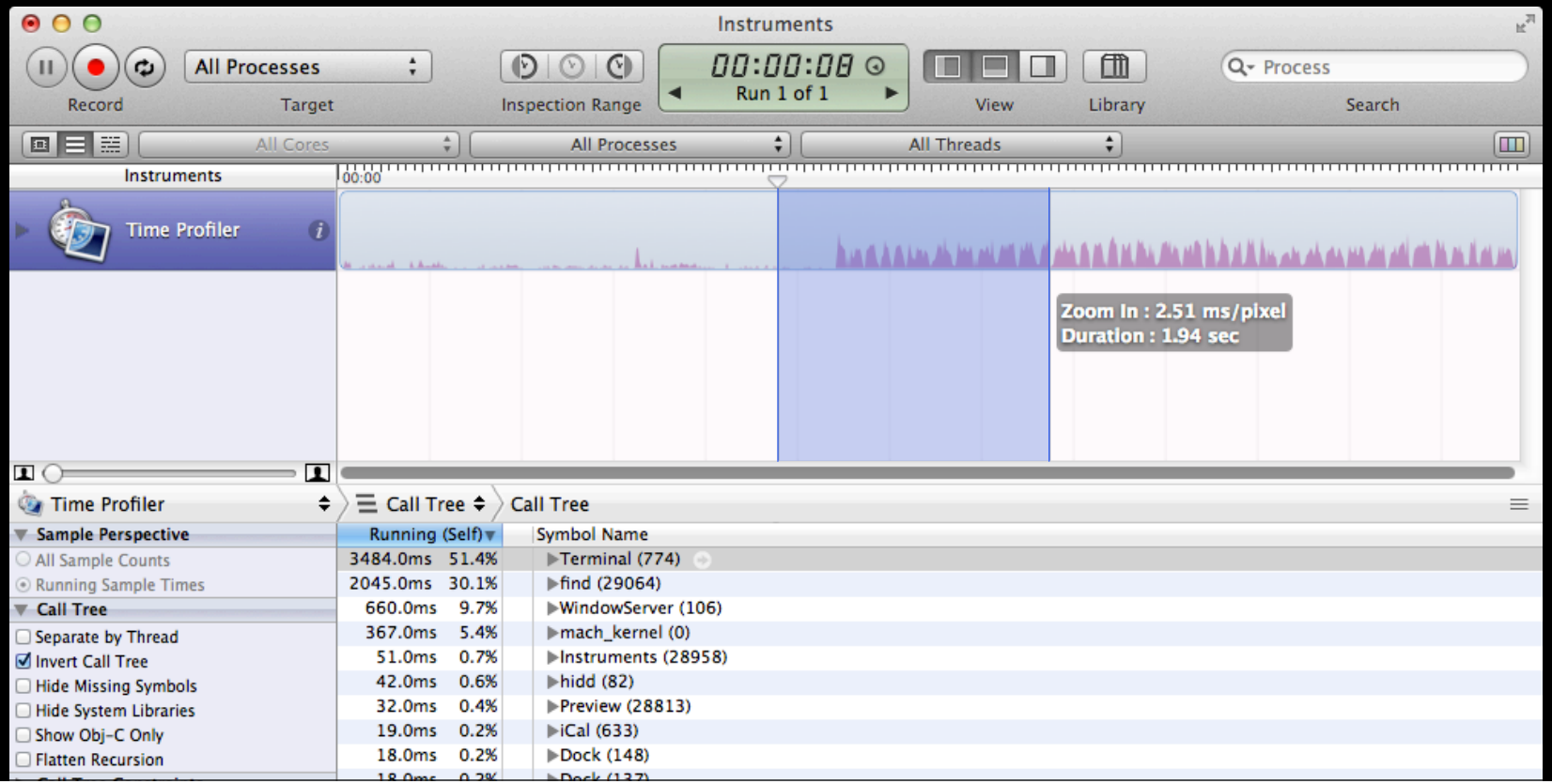
Track Gestures

Working with high-resolution traces



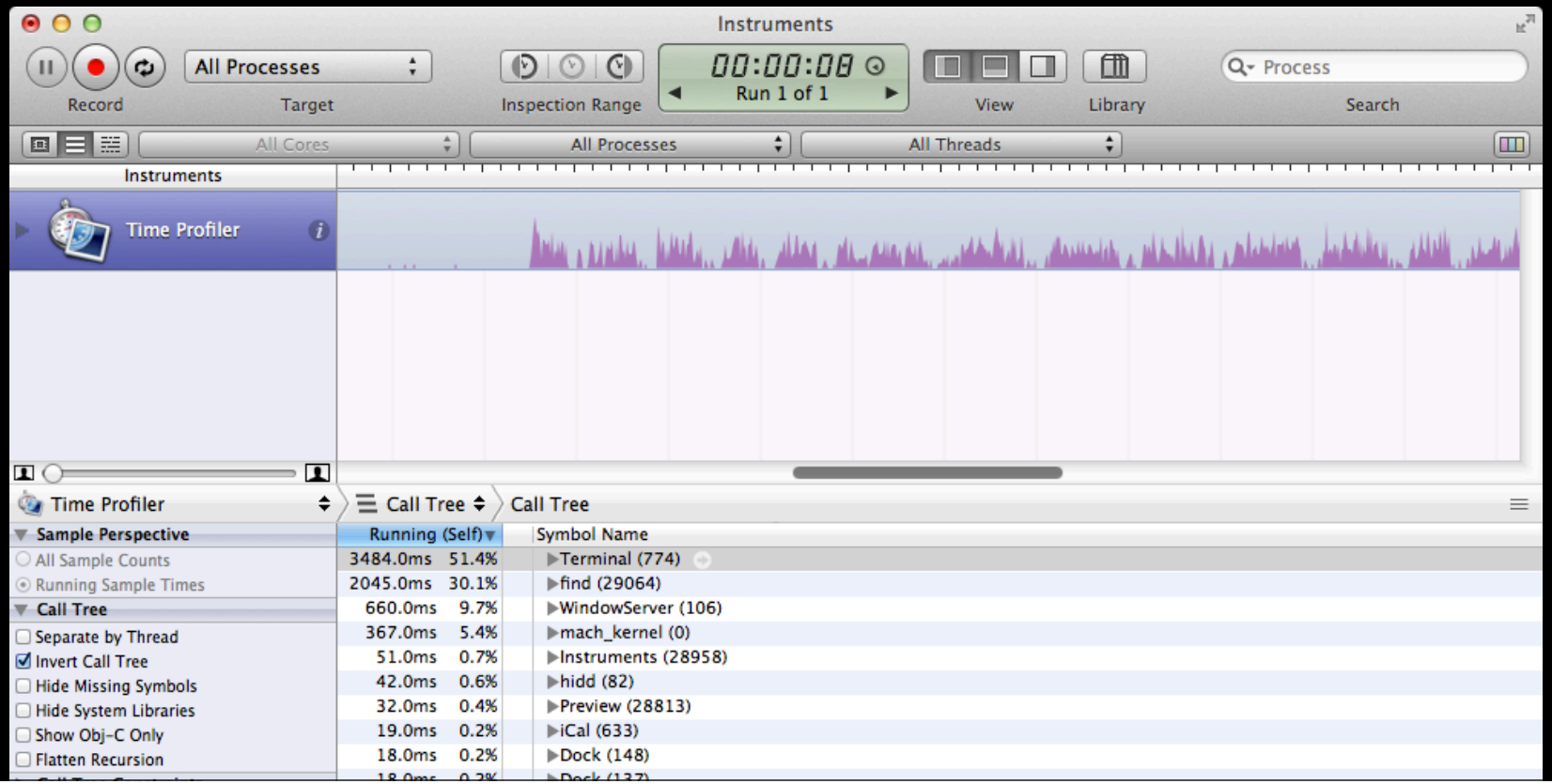
Zoom In

Shift + Mouse Drag



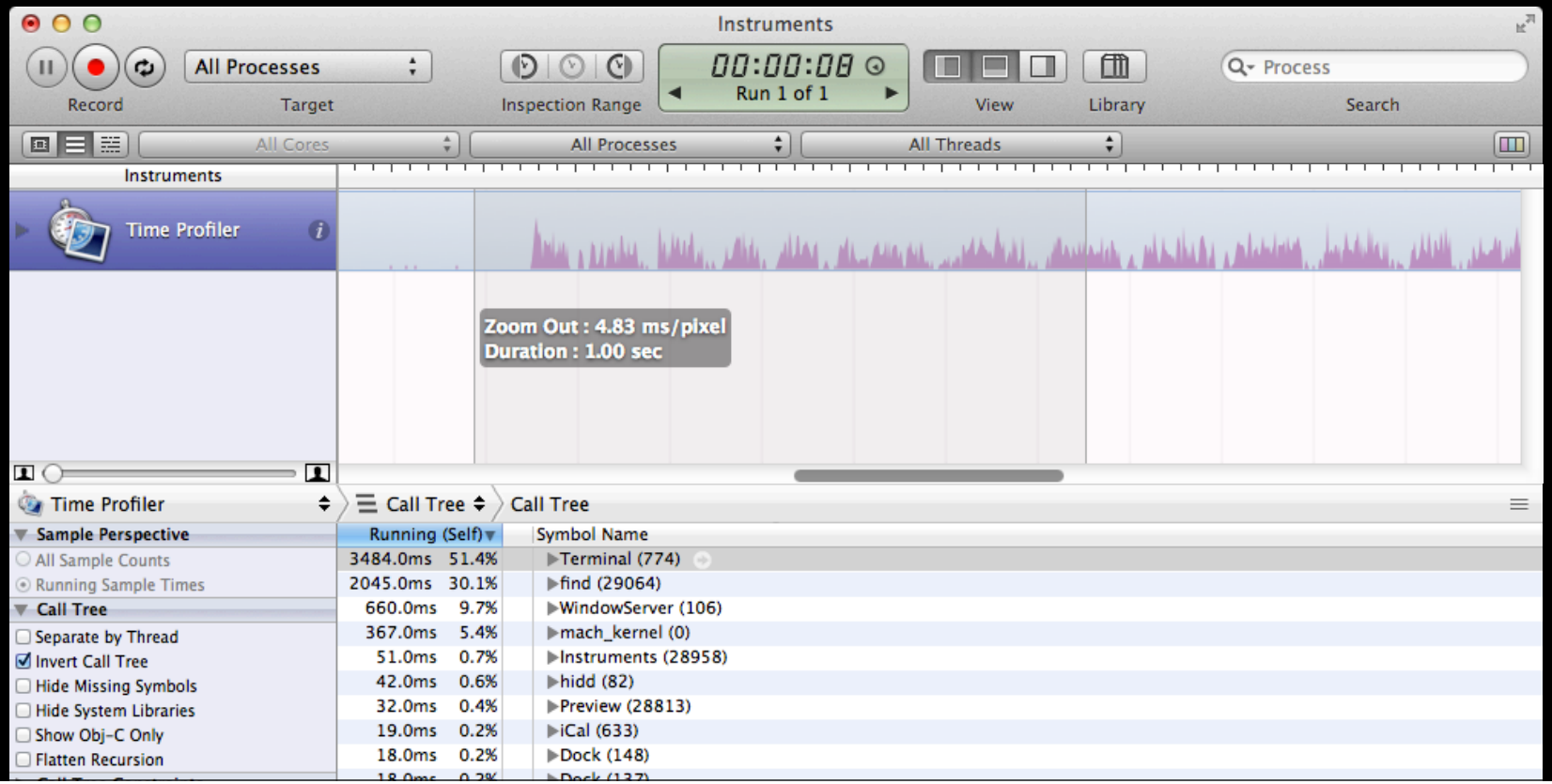
Zoom In

Shift + Mouse Drag



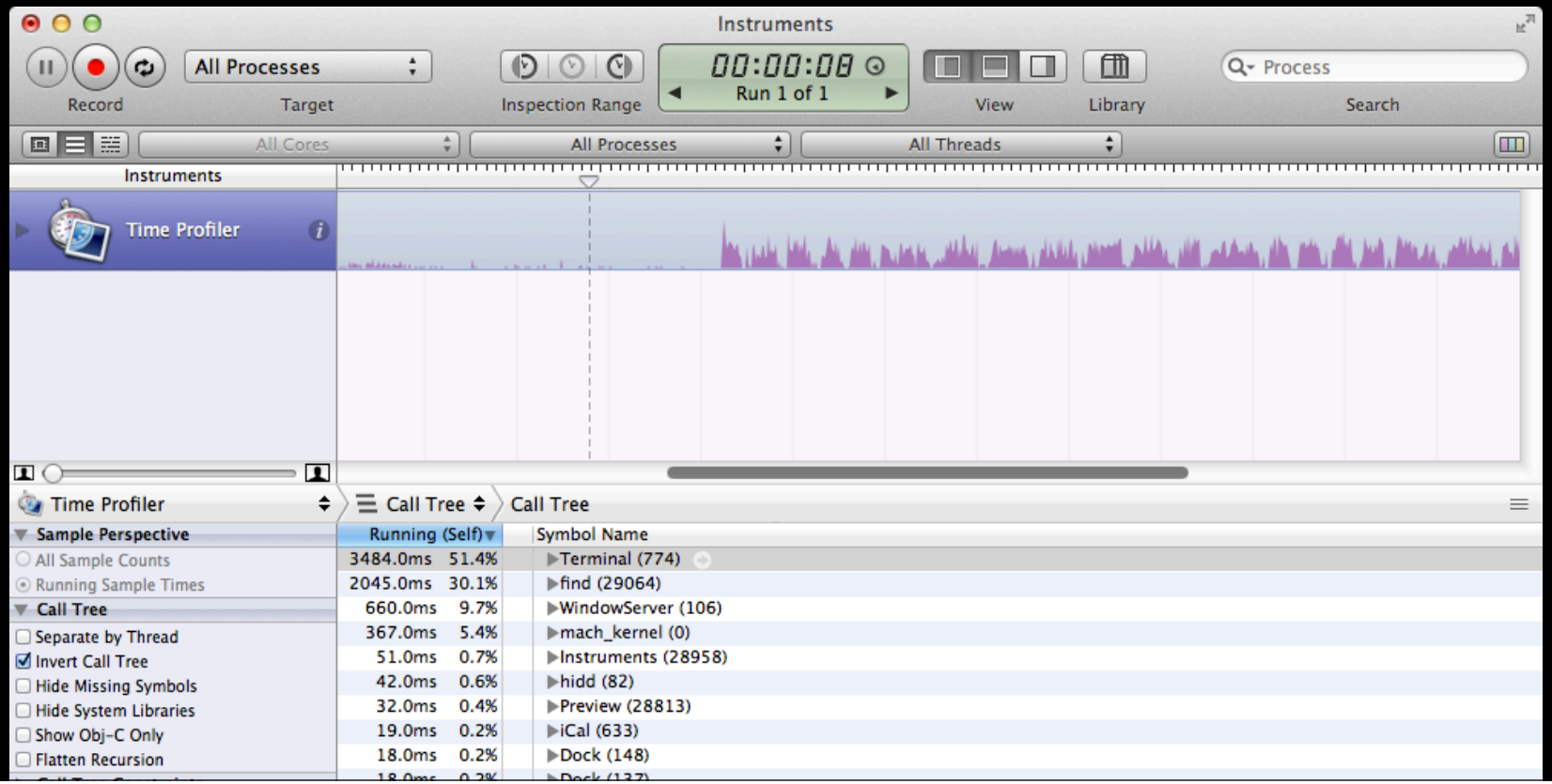
Zoom Out

Control + Mouse Drag



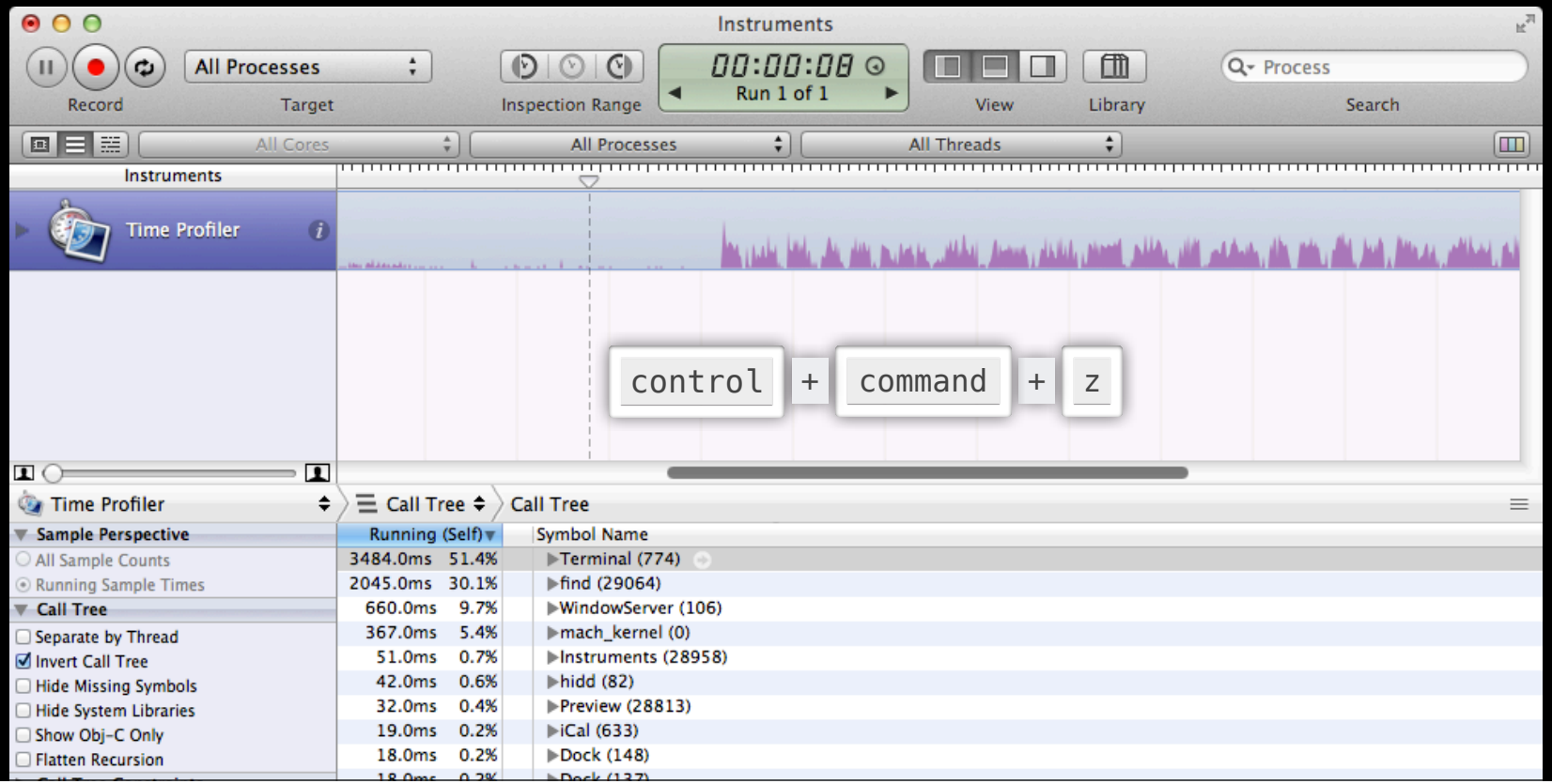
Zoom Out

Control + Mouse Drag



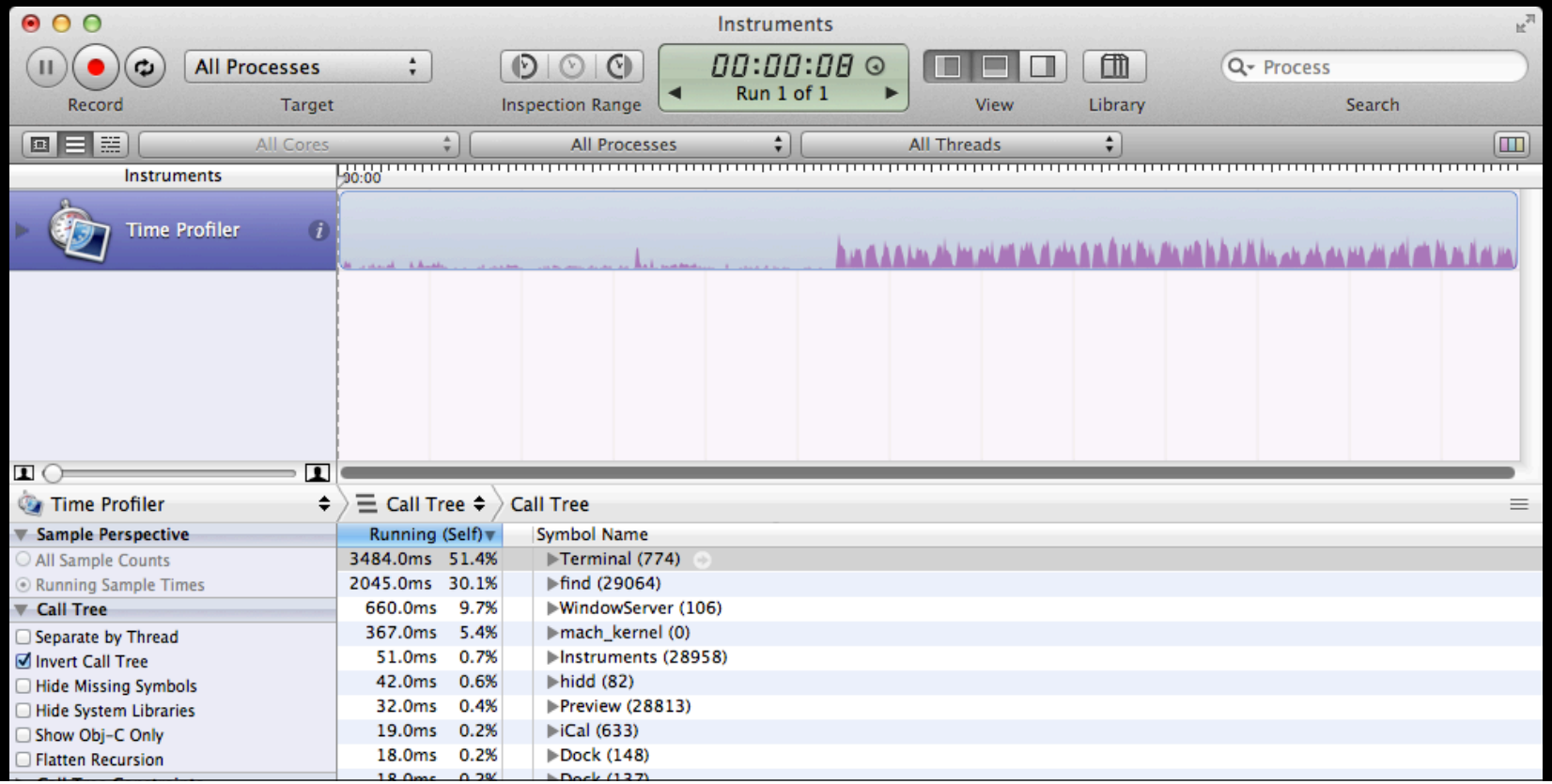
Snap Track to Fit

Control + Command + Z



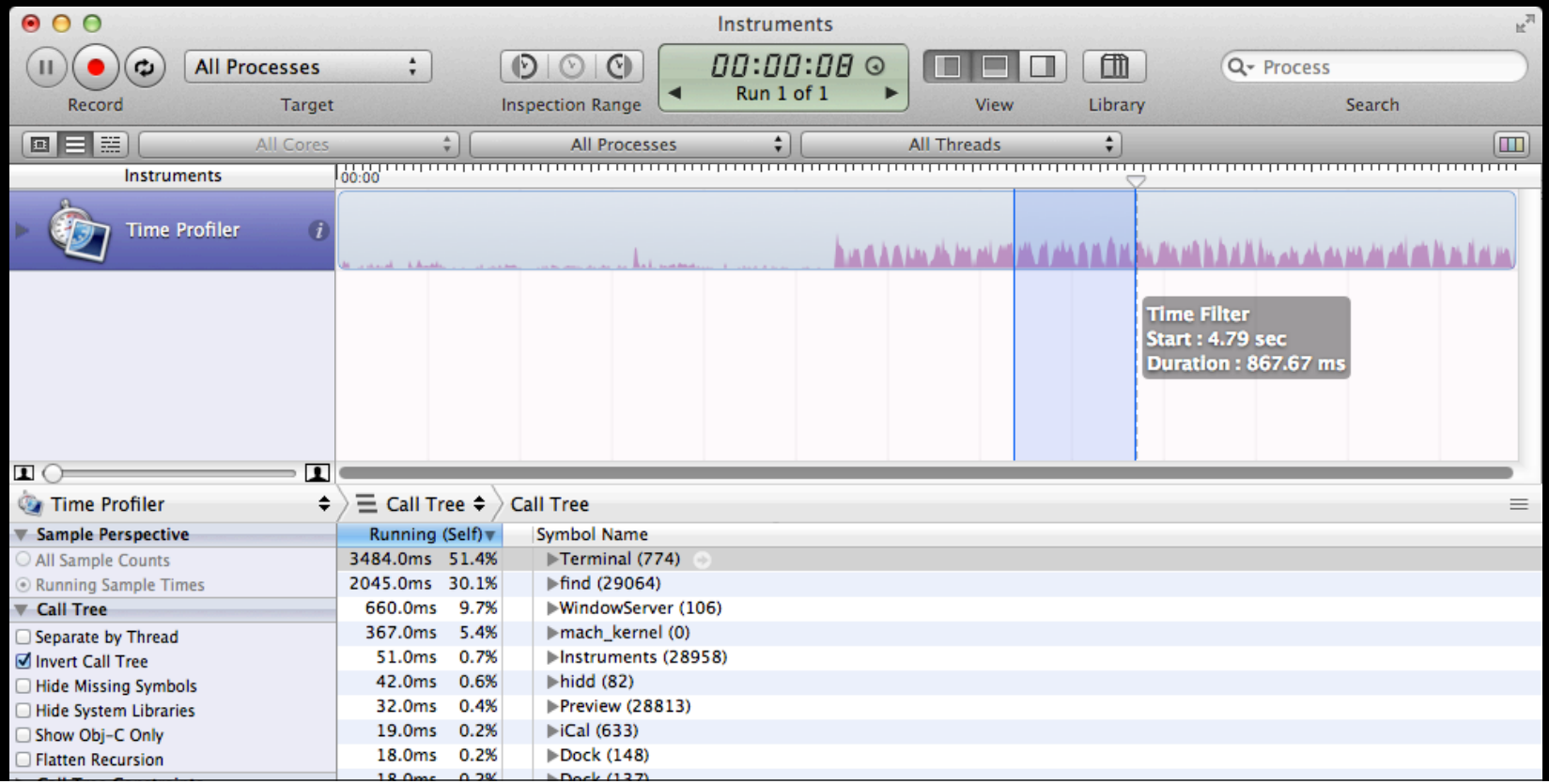
Snap Track to Fit

Control + Command + Z



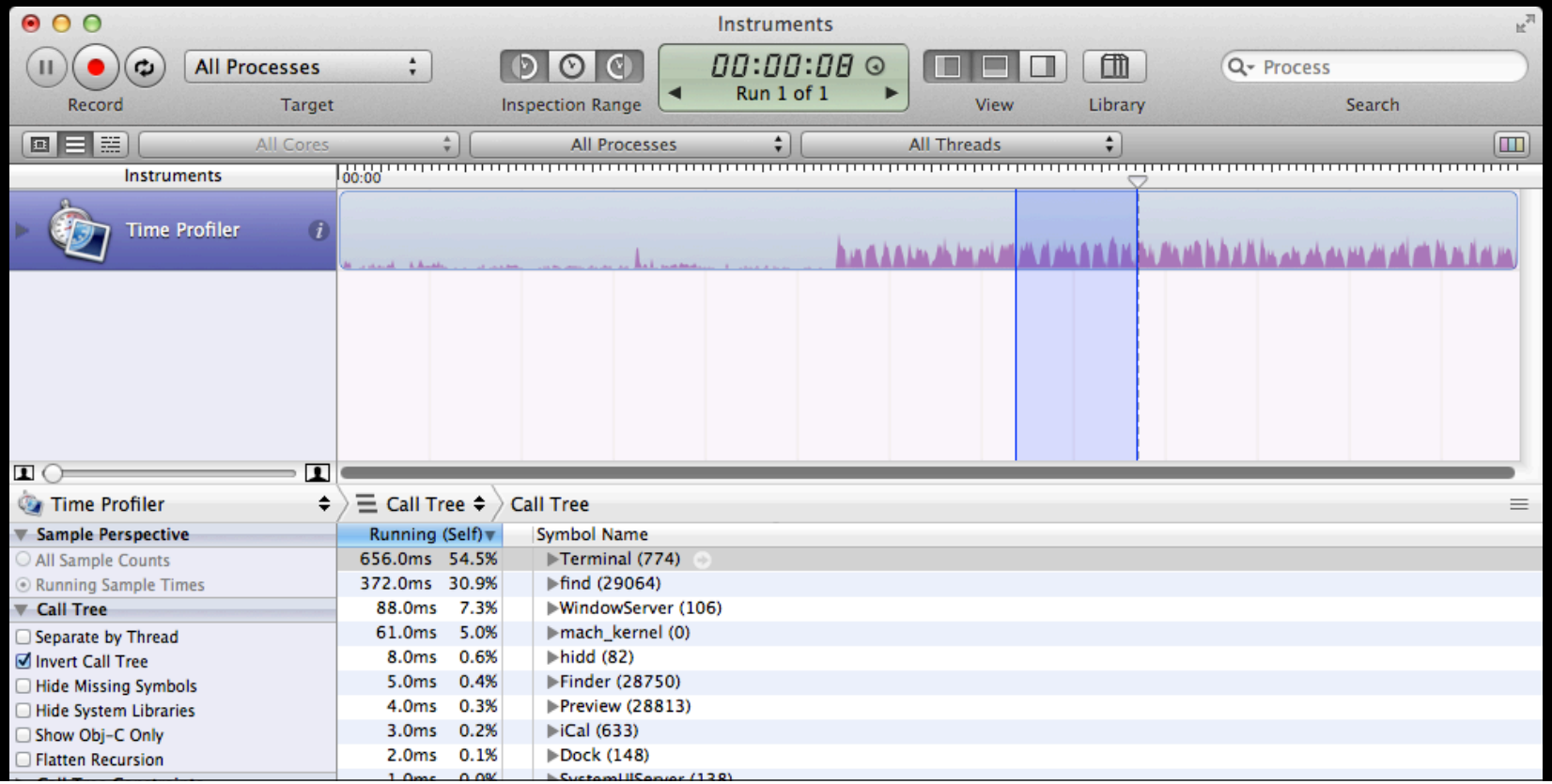
Select Time Range

Option + Mouse Drag



Select Time Range

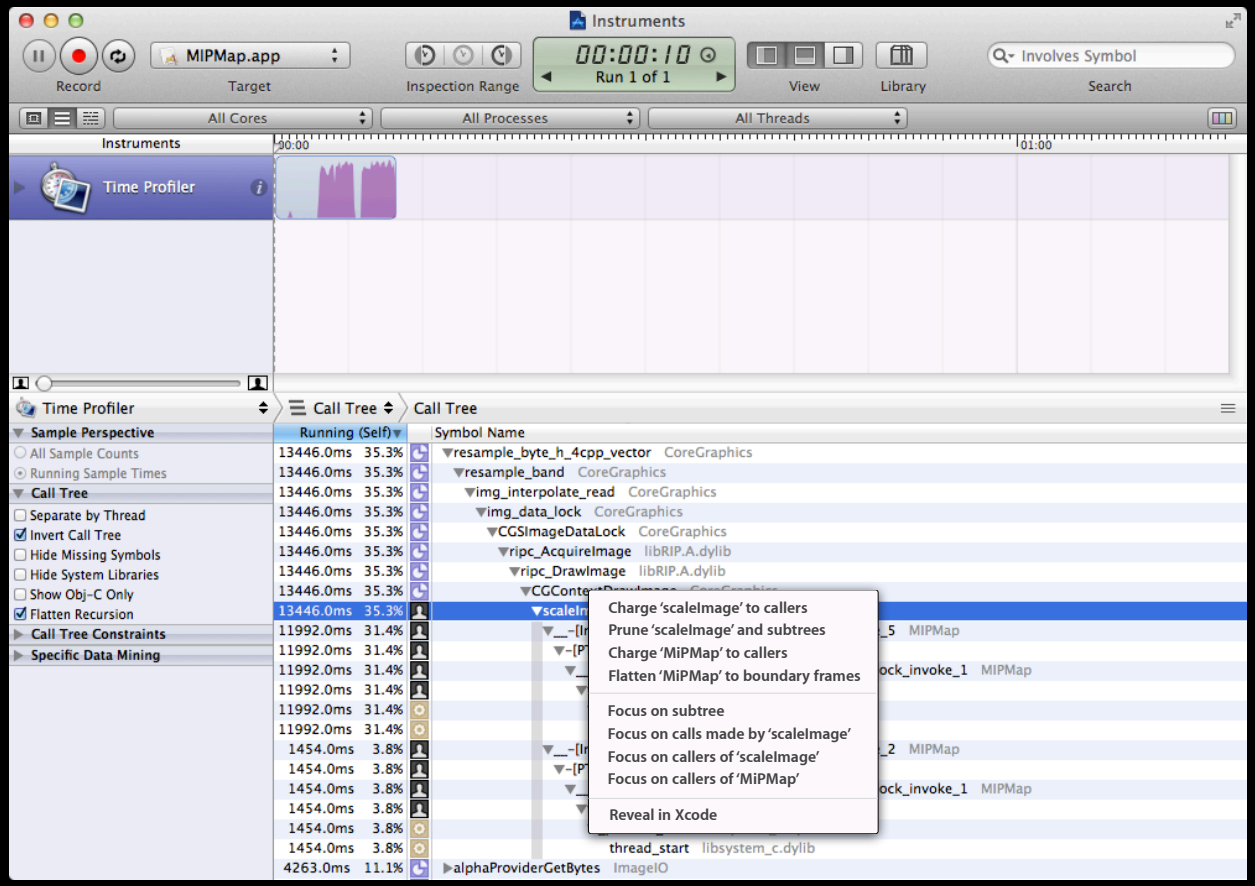
Option + Mouse Drag



Call Tree Data Mining

Data Focusing

Focus call tree to isolate factors



Data Focusing

Focus call tree to isolate relevant subtrees

- Charge 'scaleImage' to callers
- Prune 'scaleImage' and subtrees
- Charge 'MiPMap' to callers
- Flatten 'MiPMap' to boundary frames

- Focus on subtree
- Focus on calls made by 'scaleImage'
- Focus on callers of 'scaleImage'
- Focus on callers of 'MiPMap'

- Reveal in Xcode

Focus on entire tree beneath selected symbol
Used to eliminate noise in call tree

Data Focusing

Focus call tree to isolate relevant subtrees

Charge 'scaleImage' to callers
Prune 'scaleImage' and subtrees
Charge 'MiPMap' to callers
Flatten 'MiPMap' to boundary frames

Focus on subtree
Focus on calls made by 'scaleImage'
Focus on callers of 'scaleImage'
Focus on callers of 'MiPMap'

Reveal in Xcode

Focus on all calls made by symbol
Used to identify all calls made by symbol

Data Focusing

Focus call tree to isolate relevant subtrees

- Charge 'scaleImage' to callers
- Prune 'scaleImage' and subtrees
- Charge 'MiPMap' to callers
- Flatten 'MiPMap' to boundary frames

- Focus on subtree
- Focus on calls made by 'scaleImage'
- Focus on callers of 'scaleImage'
- Focus on callers of 'MiPMap'

- Reveal in Xcode

Focus on all callers of selected symbol
Useful to identify who all calls the selected symbol

Data Focusing

Focus call tree to isolate relevant subtrees

Charge 'scaleImage' to callers
Prune 'scaleImage' and subtrees
Charge 'MiPMap' to callers
Flatten 'MiPMap' to boundary frames

Focus on subtree
Focus on calls made by 'scaleImage'
Focus on callers of 'scaleImage'
Focus on callers of 'MiPMap'

Reveal in Xcode

Focus on all callers into the selected library
Useful to see who all uses a given library

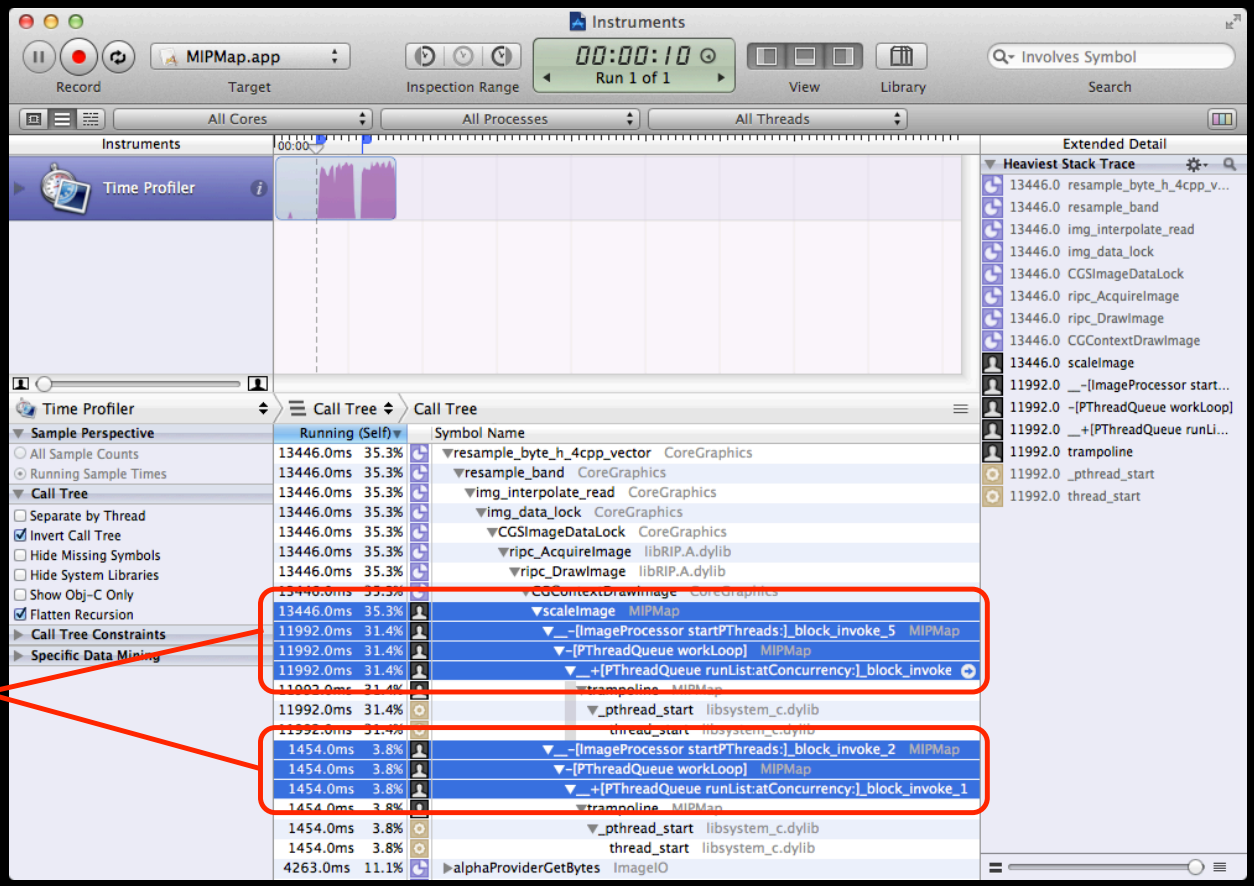
Find in Detail Views

Find data without filter out surrounding context

Find
Perform find in all table and outline detail views

Stack Copy

Discontinuous selection with shallow or deep copy

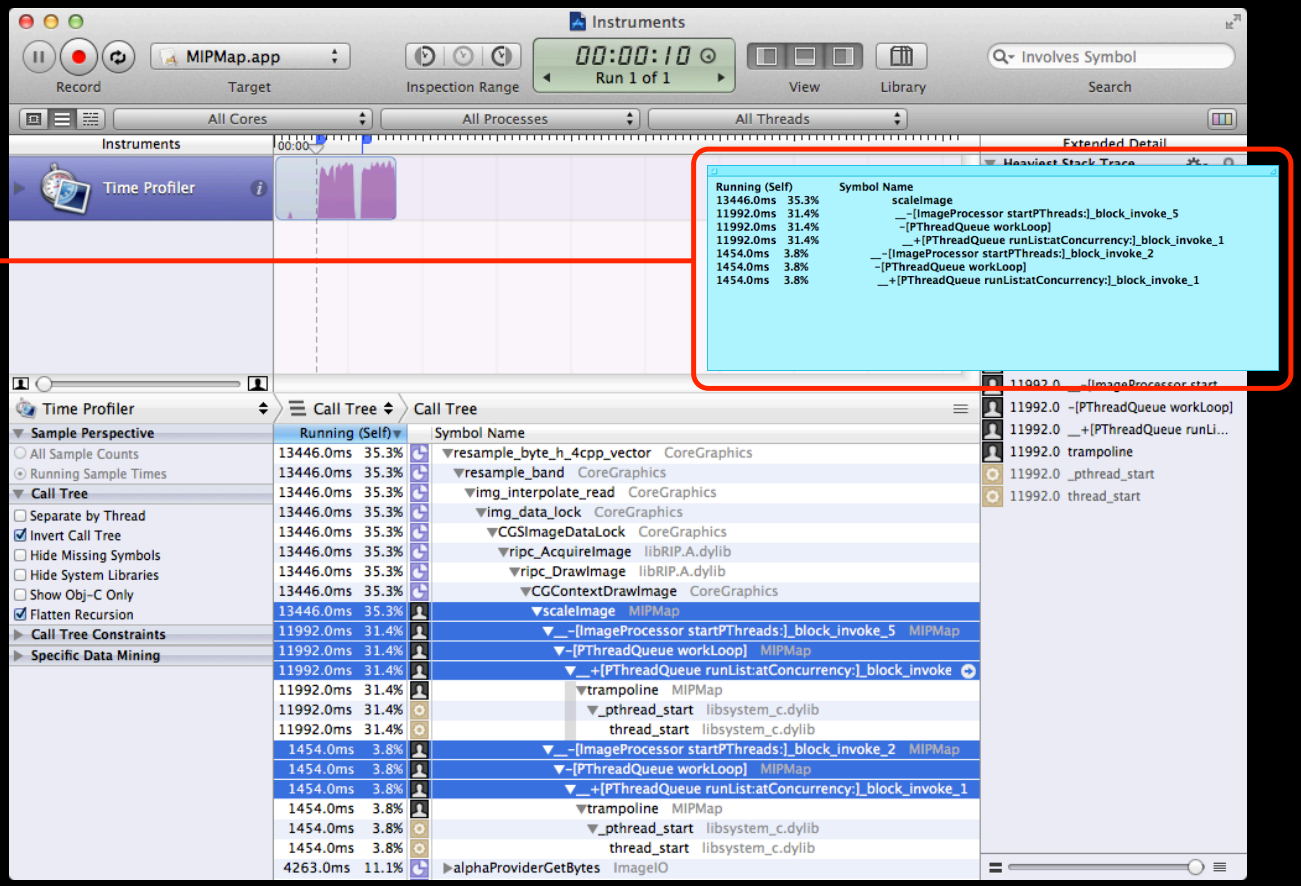


Copy Discontinuous Selection
Select any combination of lines in detail view

Stack Copy User Interface

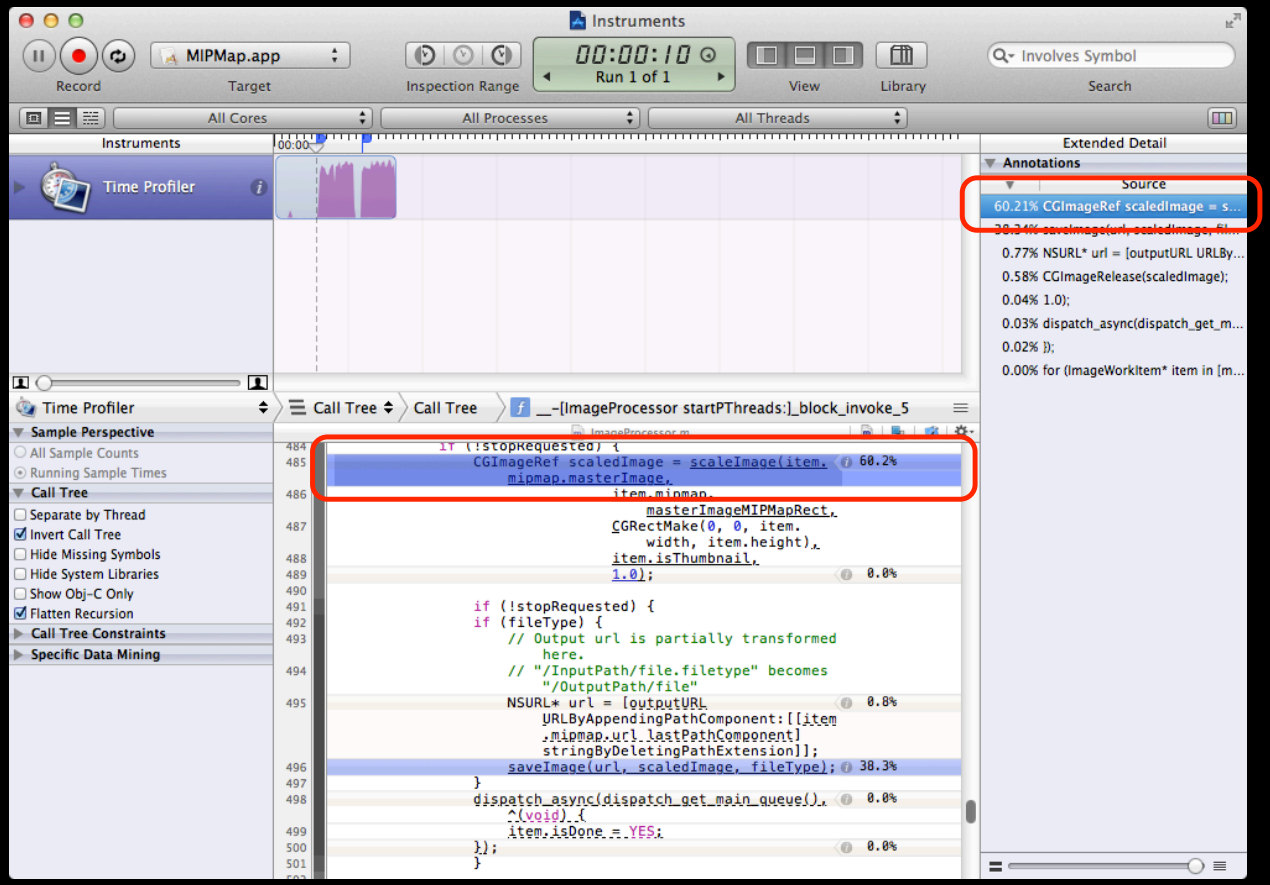
Discontinuous selection with shallow or deep copy

Paste
Text copied includes all columns and outline indentation



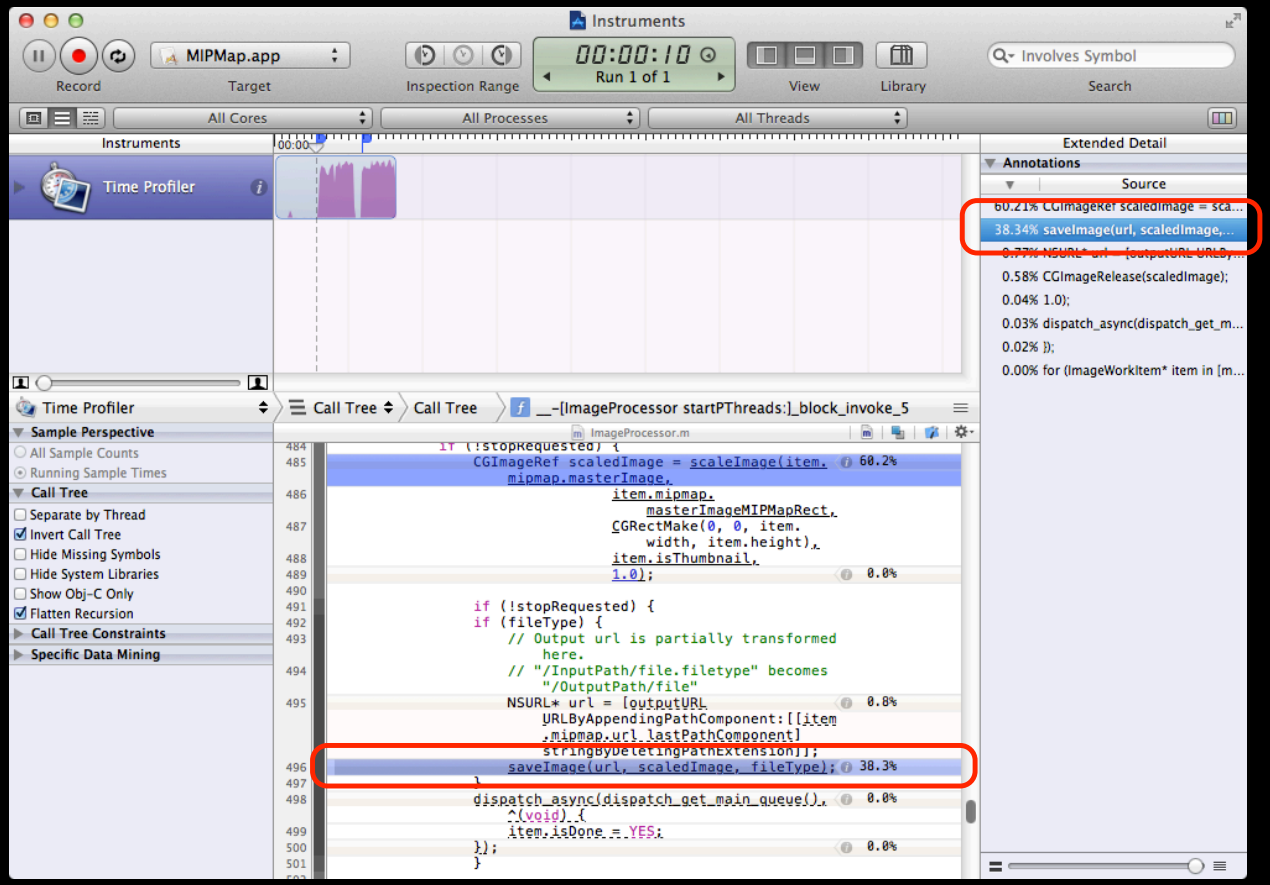
Navigation of Source Annotations

Navigate by hottest to coolest hits



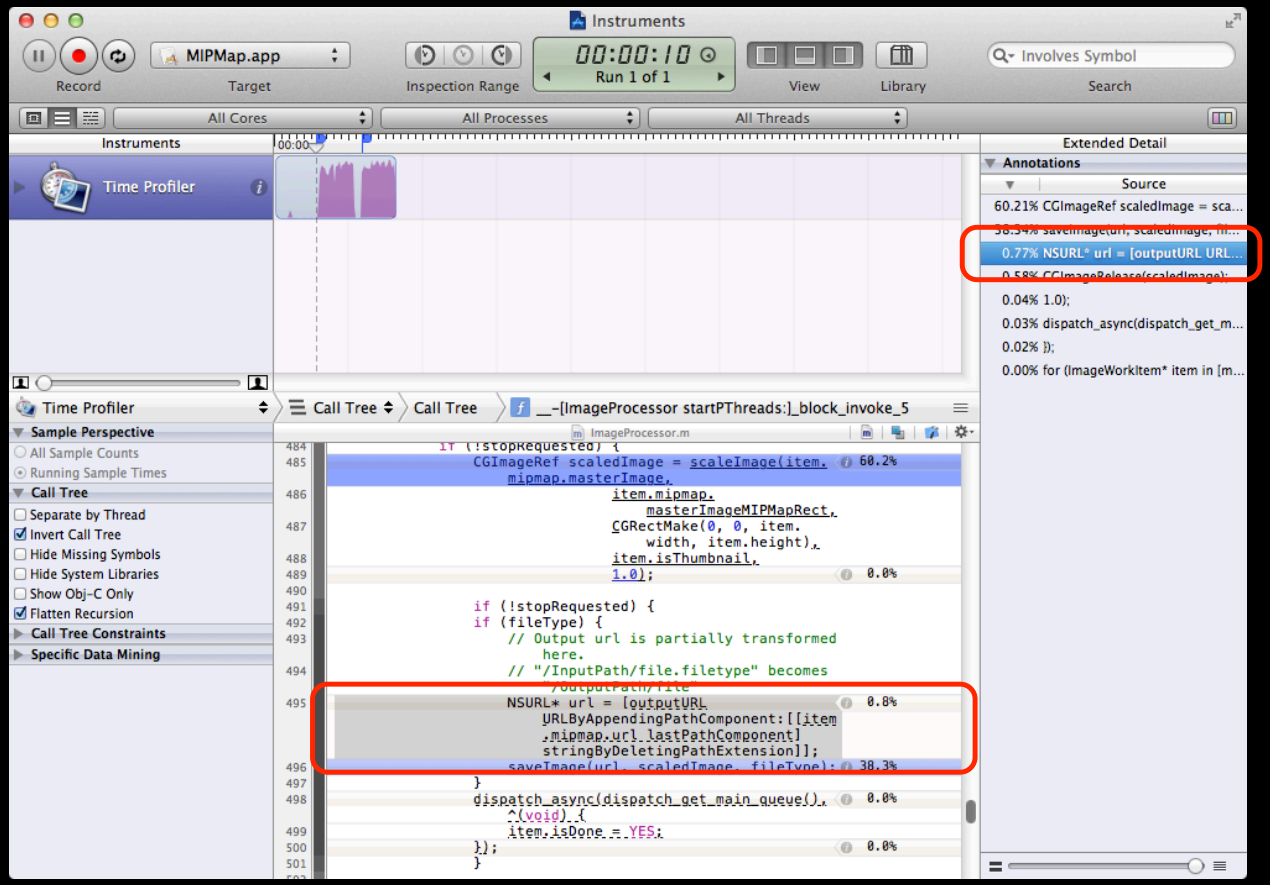
Navigation of Source Annotations

Navigate by hottest to coolest hits

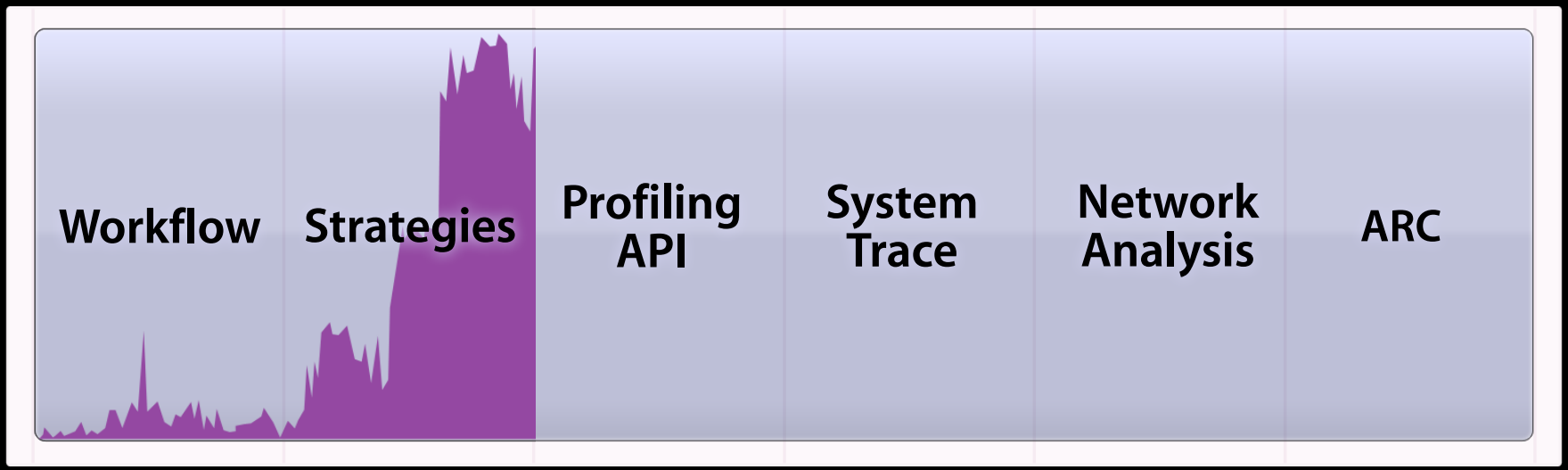


Navigation of Source Annotations

Navigate by hottest to coolest hits



Strategies

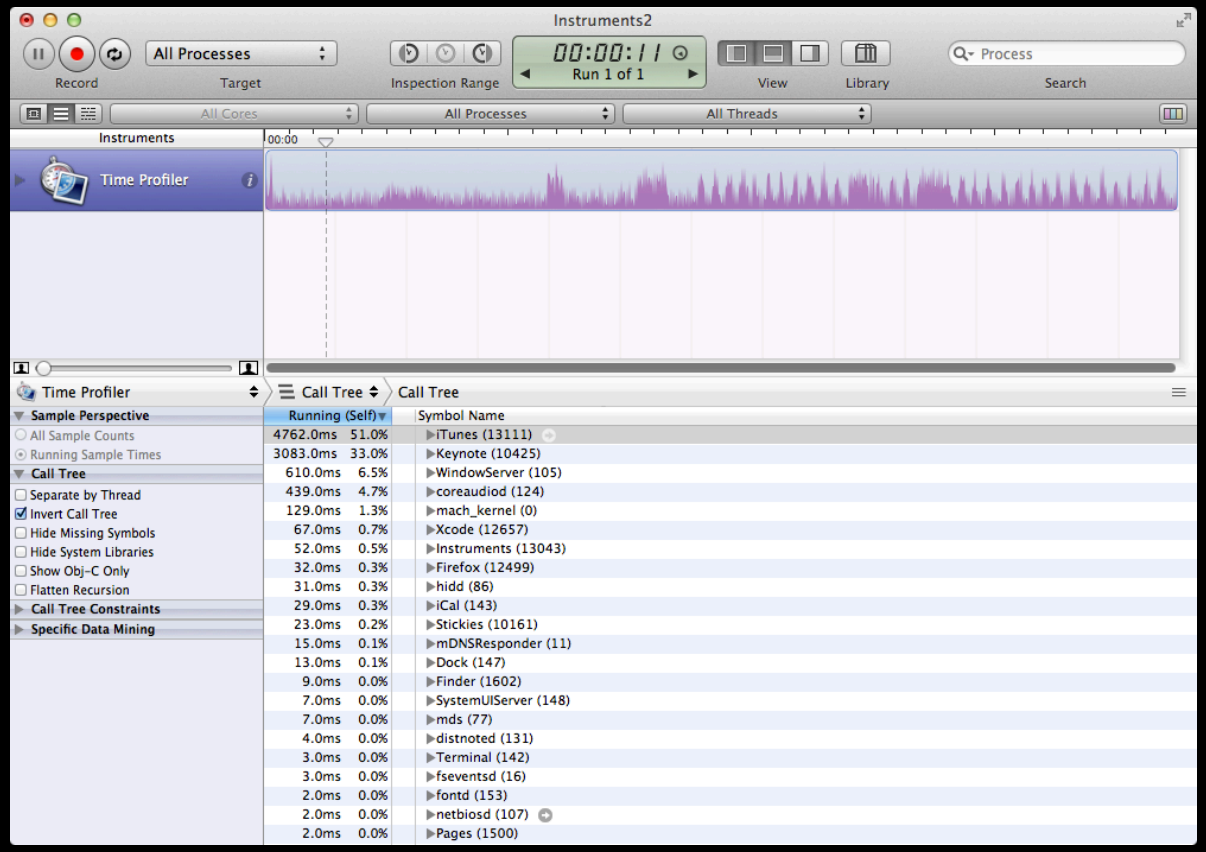

Workflow
Strategies
Profiling API
System Trace
Network Analysis
ARC

What Is a “Strategy?”

A method to categorize, display, and highlight data gathered from multiple instruments along a common axis

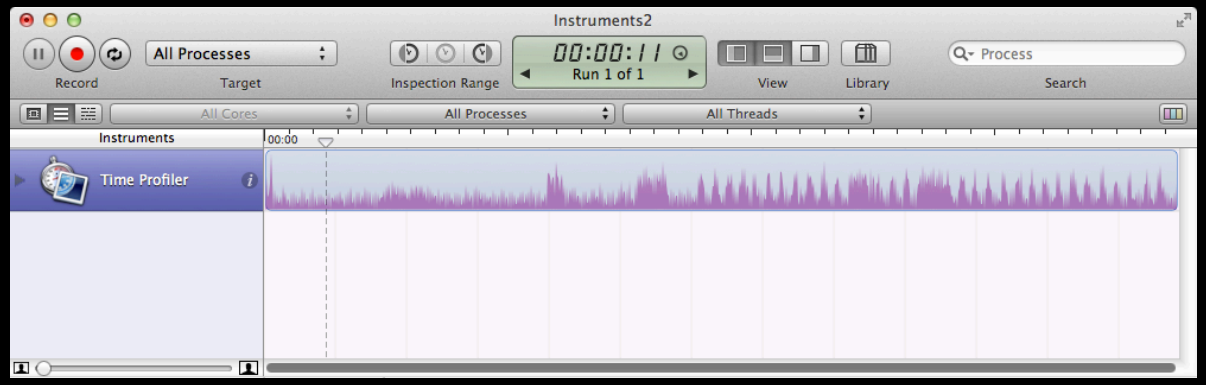
Strategies

The “Instruments” strategy



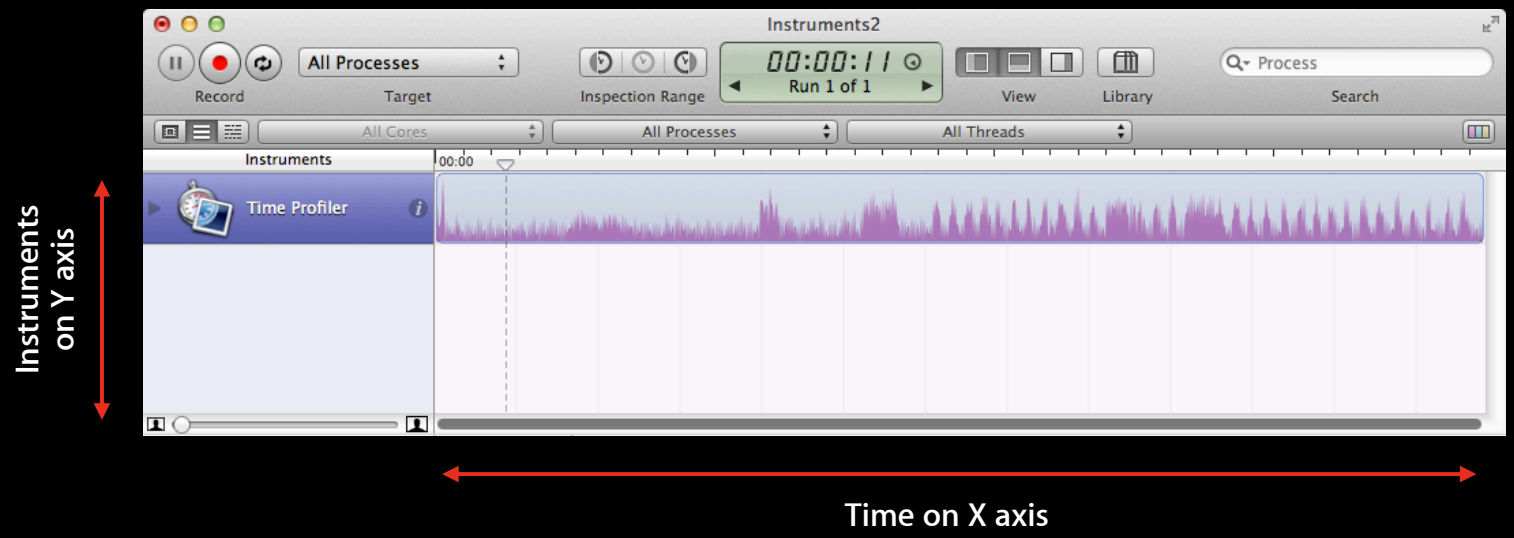
Strategies

The “Instruments” strategy



Strategies

The “Instruments” strategy

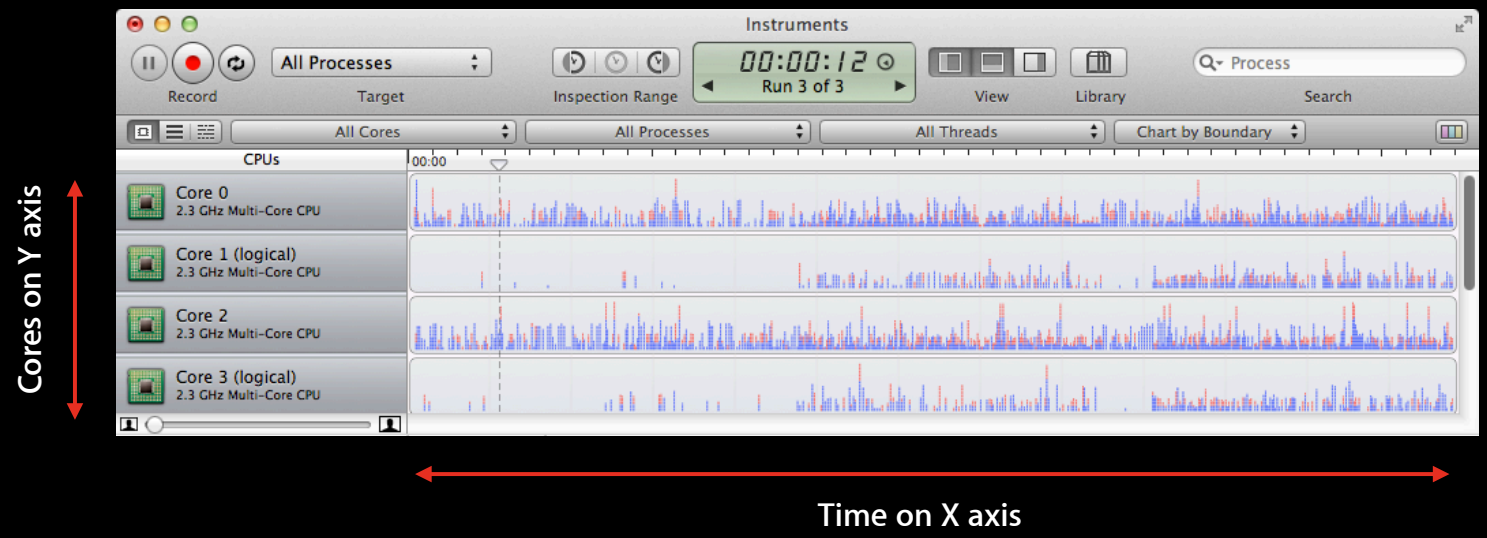


Instruments on Y axis

Time on X axis

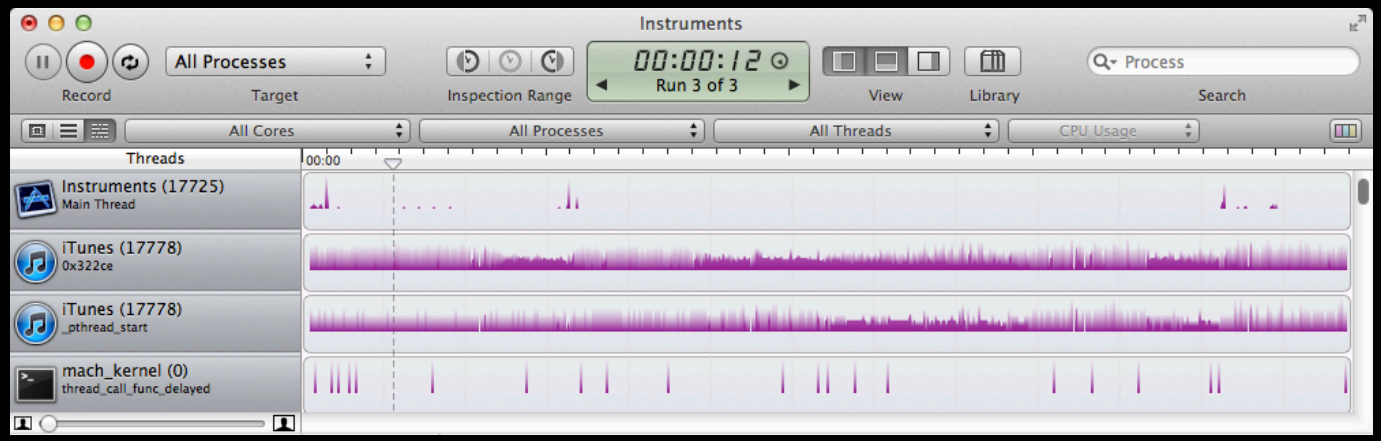
Strategies

The "CPU" strategy



Strategies

The "Threads" strategy



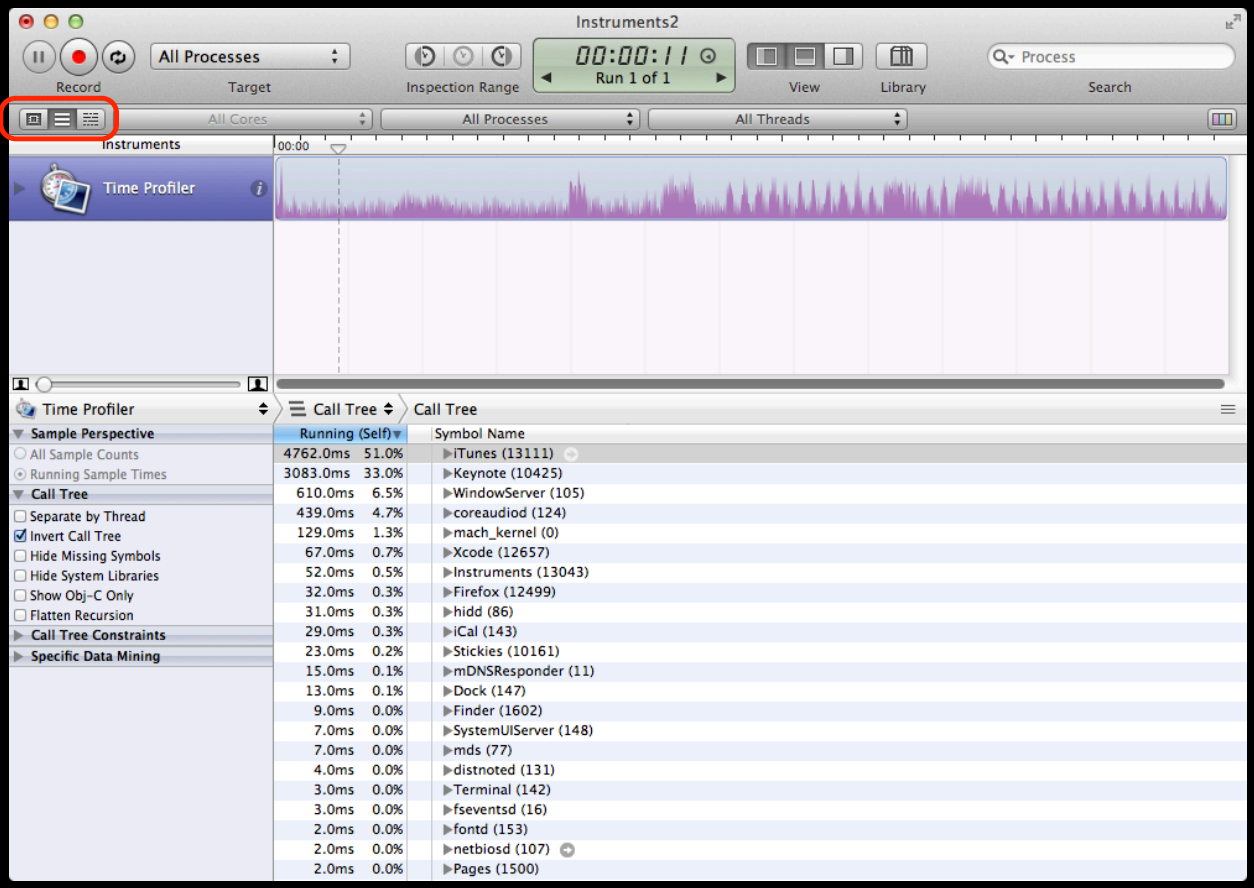
Threads on Y axis

Time on X axis

What Are the Elements of a Strategy?

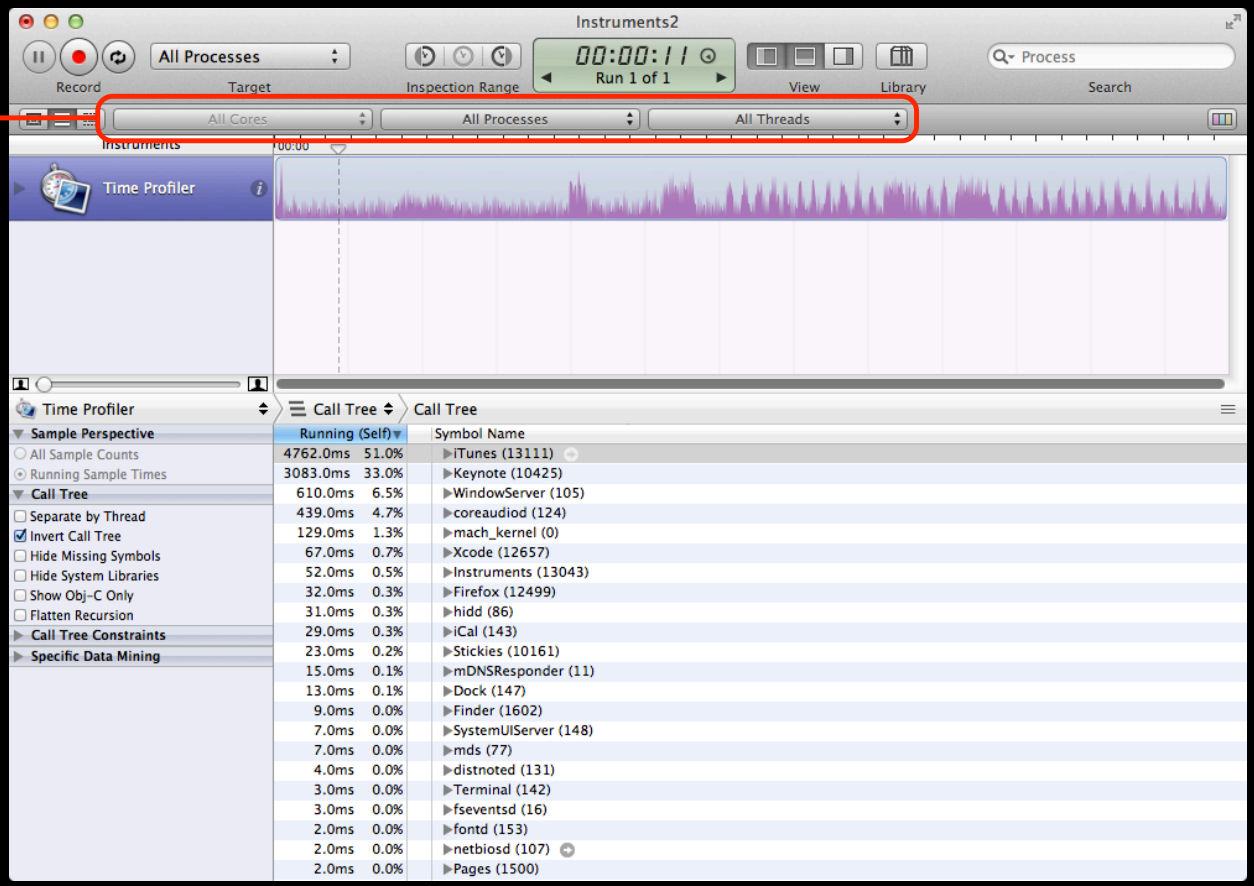
Elements of a Strategy

Strategy Chooser
Controls which strategy to display



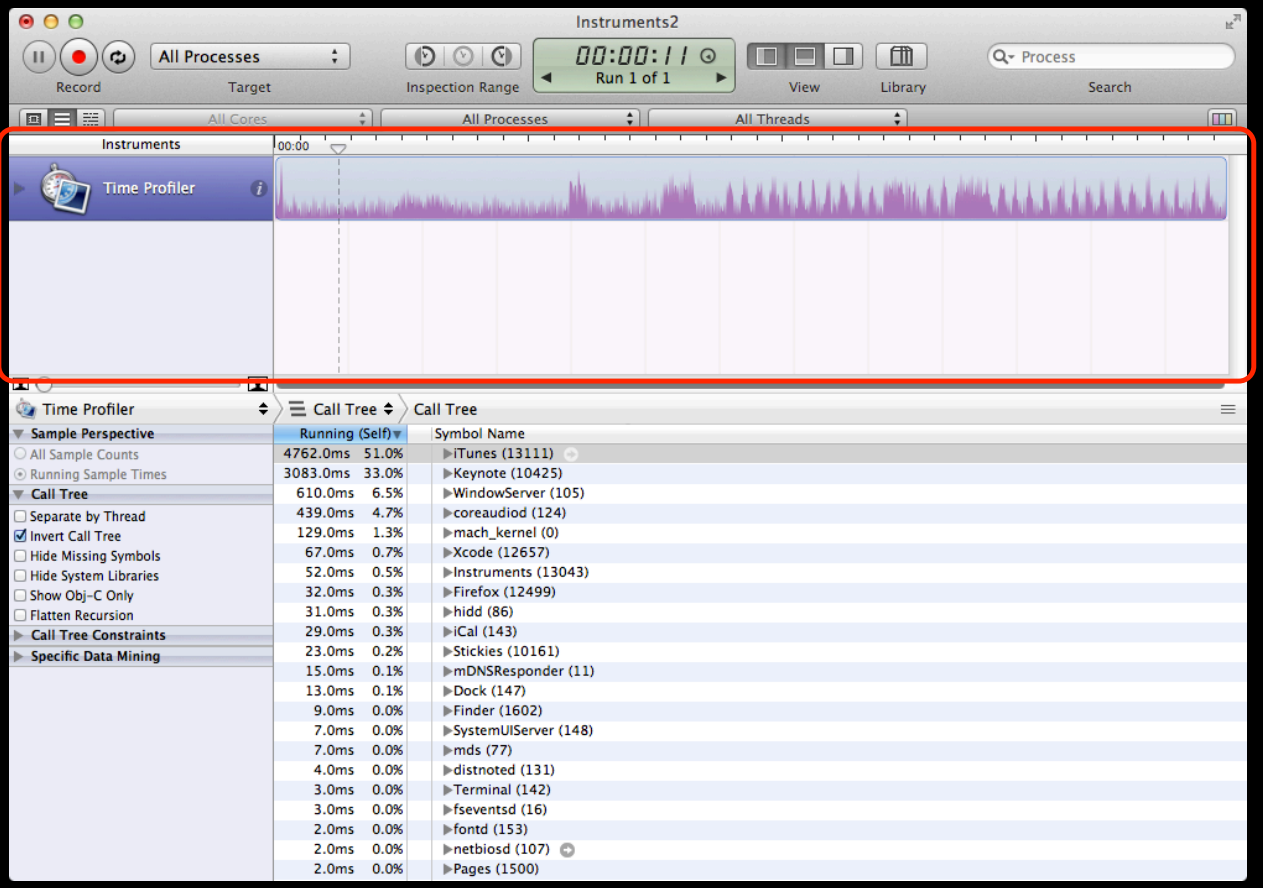
Elements of a Strategy

Highlight Controls
Controls highlighting of data in track and detail views



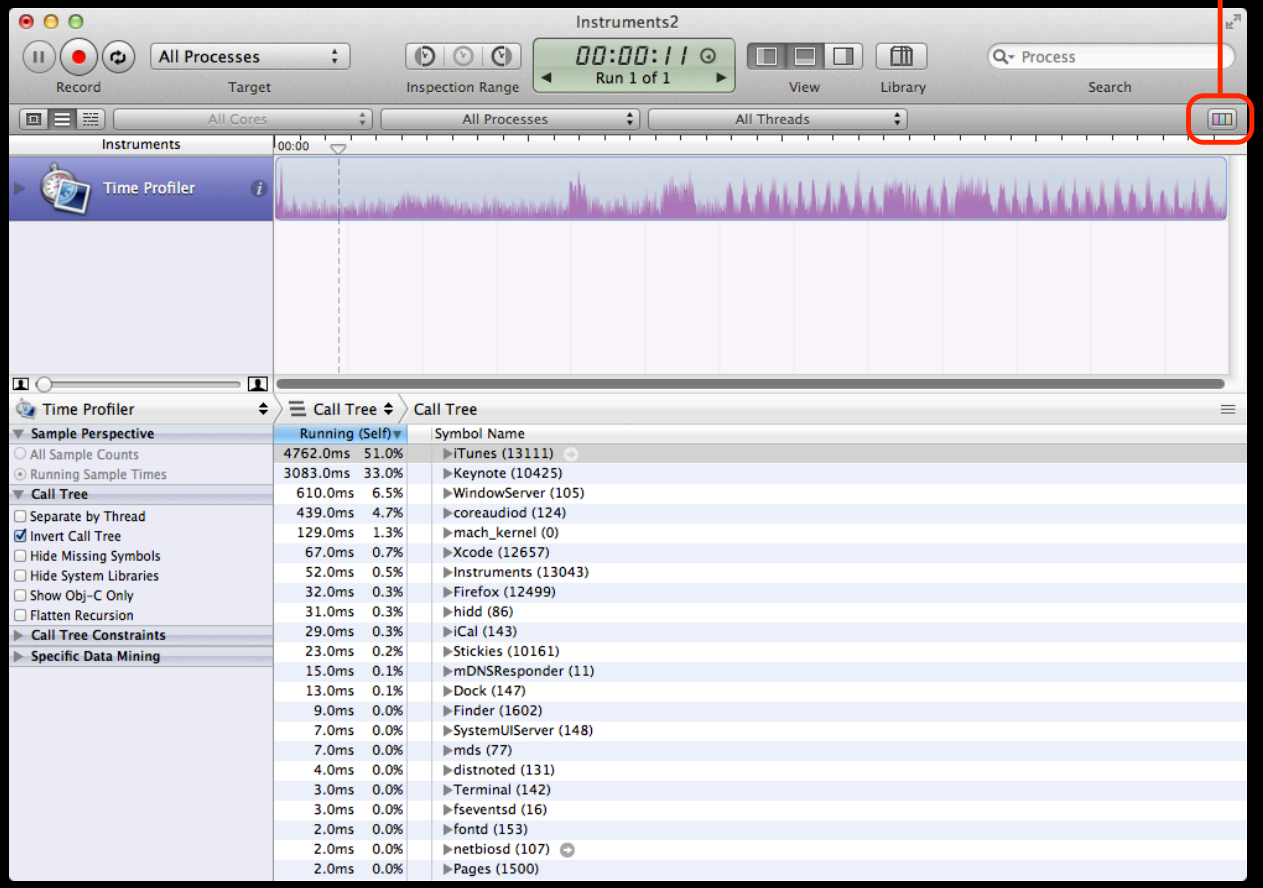
Elements of a Strategy

Track View
Displays data relevant to strategy

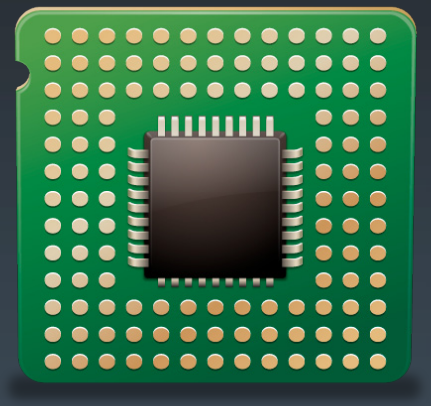


Elements of a Strategy

Legend
Displays a key legend
for the selected strategy



CPU Strategy In-Depth



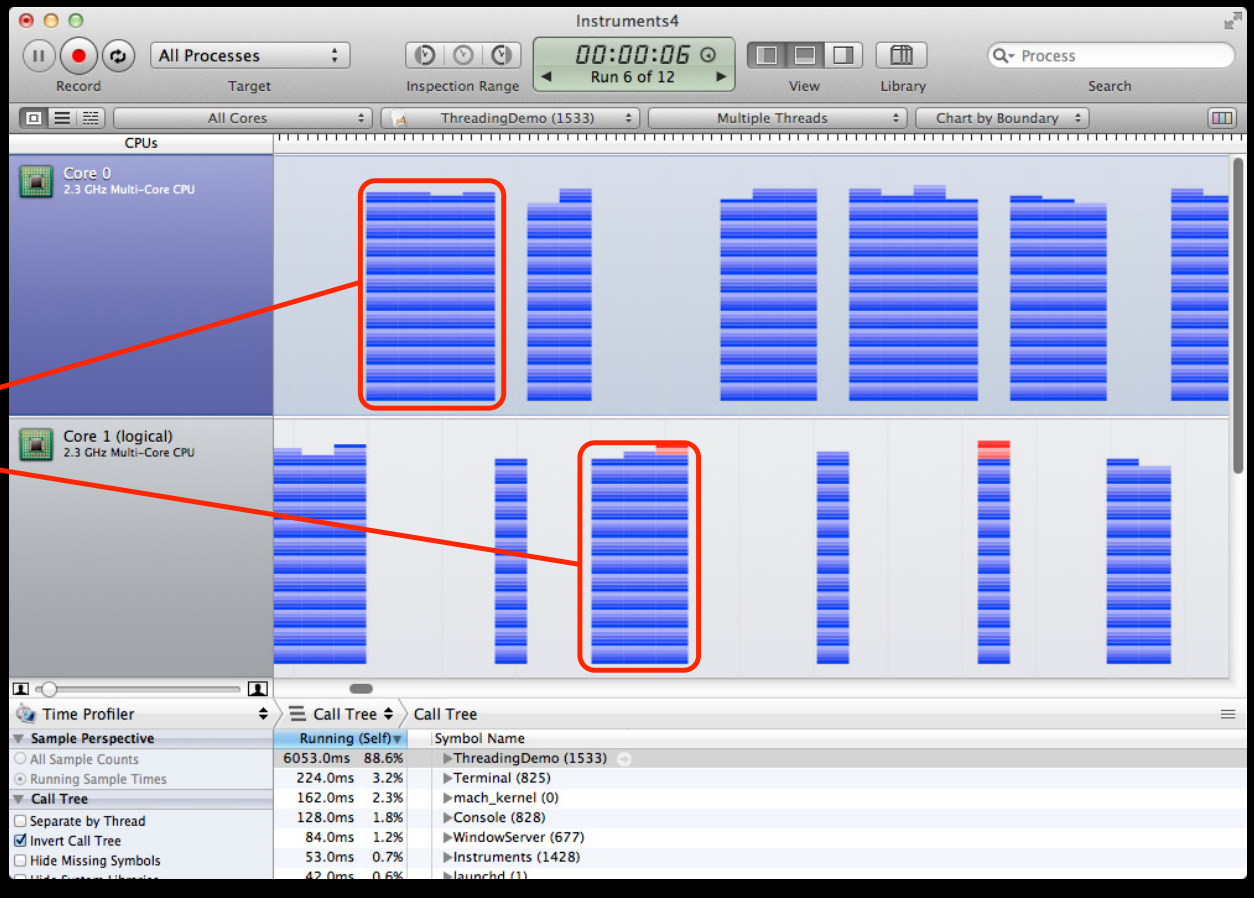
Why Use the CPU Strategy?

- Determine frequency and duration that your code is on CPU
- Identify what keeps your threads off CPU
  - Poor concurrency
    - Lock contention
  - Busy system
    - Multiple processes contending for CPU

Poor Concurrency

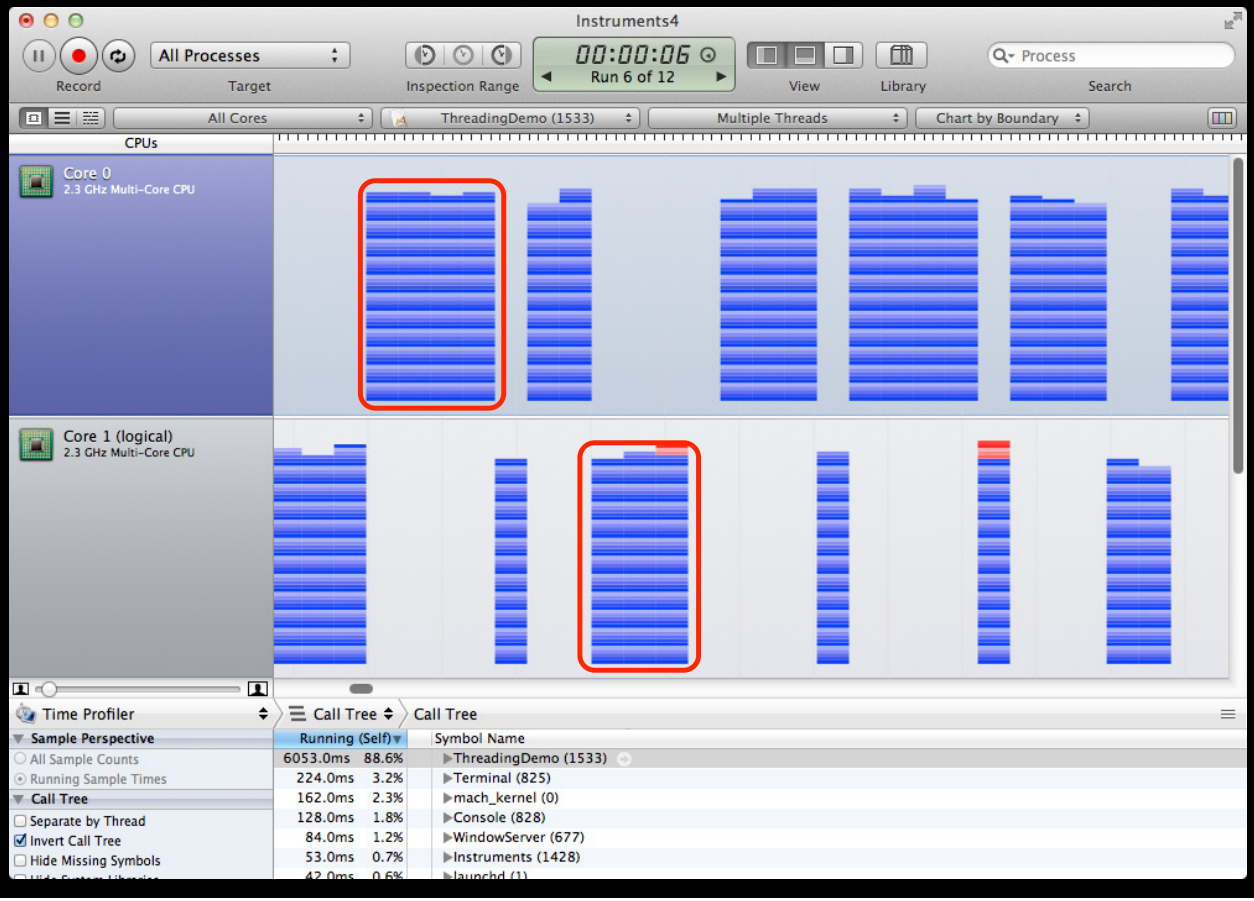
Lock contention

Running Code
Samples taken
while on CPU



Poor Concurrency

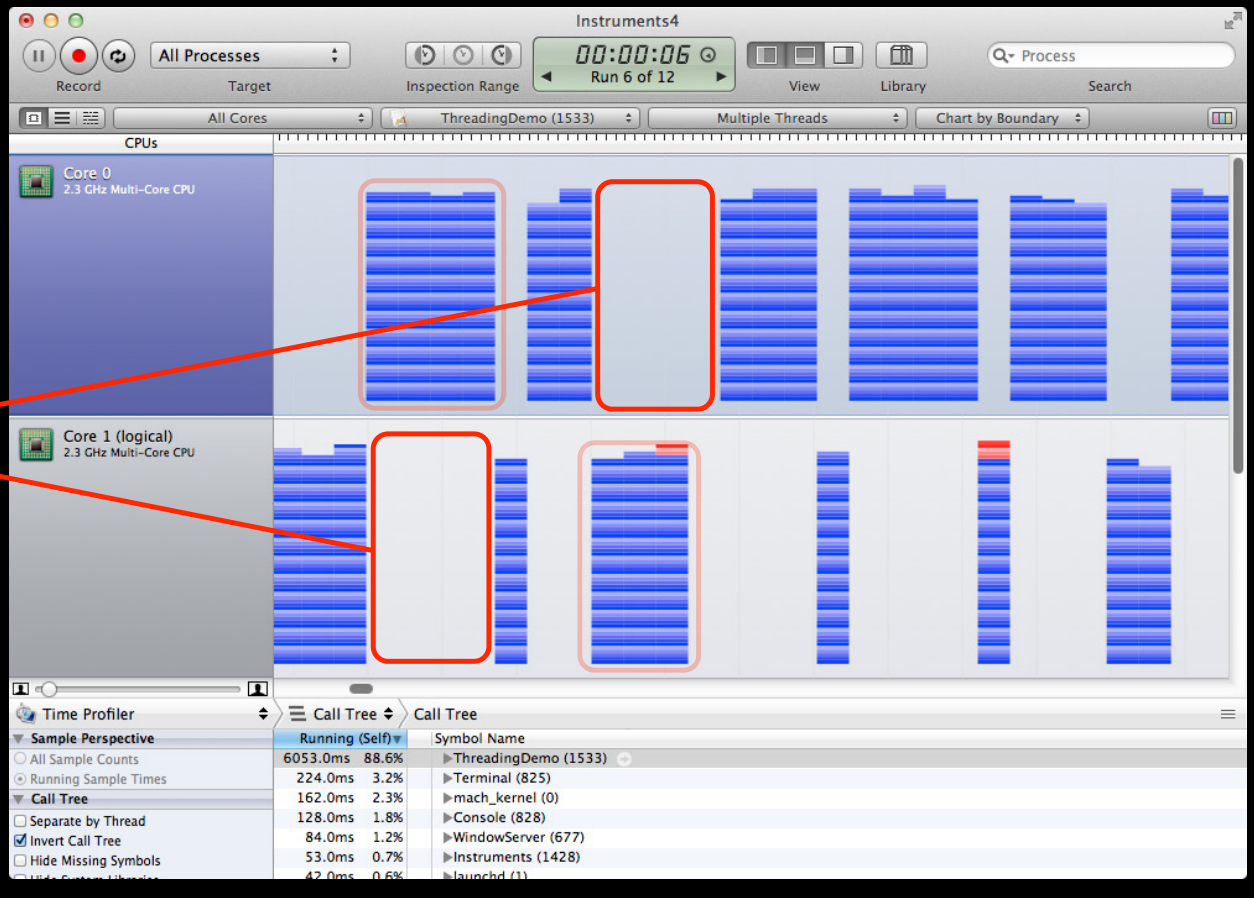
Lock contention



Poor Concurrency

Lock contention

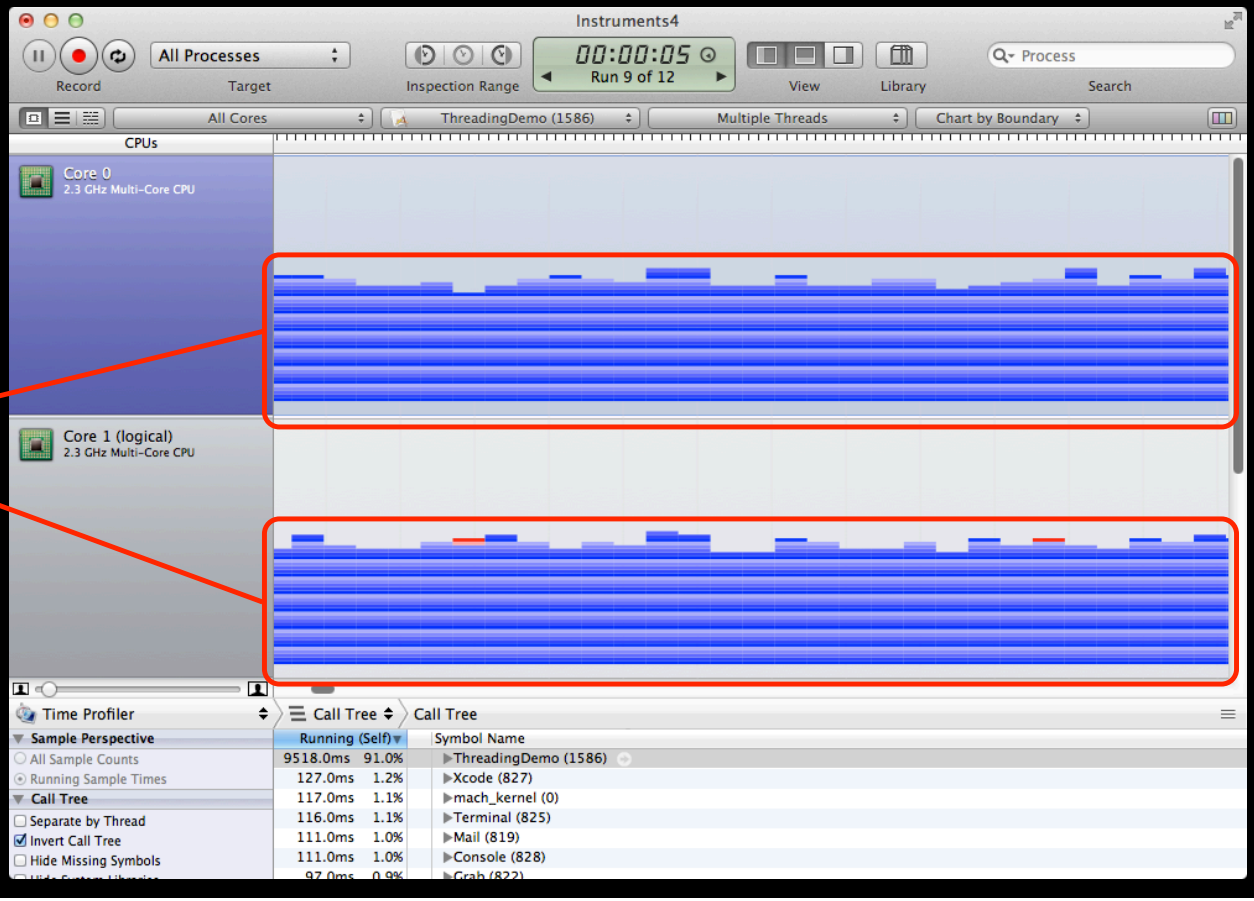
Idle Processor
Gaps indicating no activity on CPU



Idealized Concurrency

Lock contention eliminated with full CPU utilization

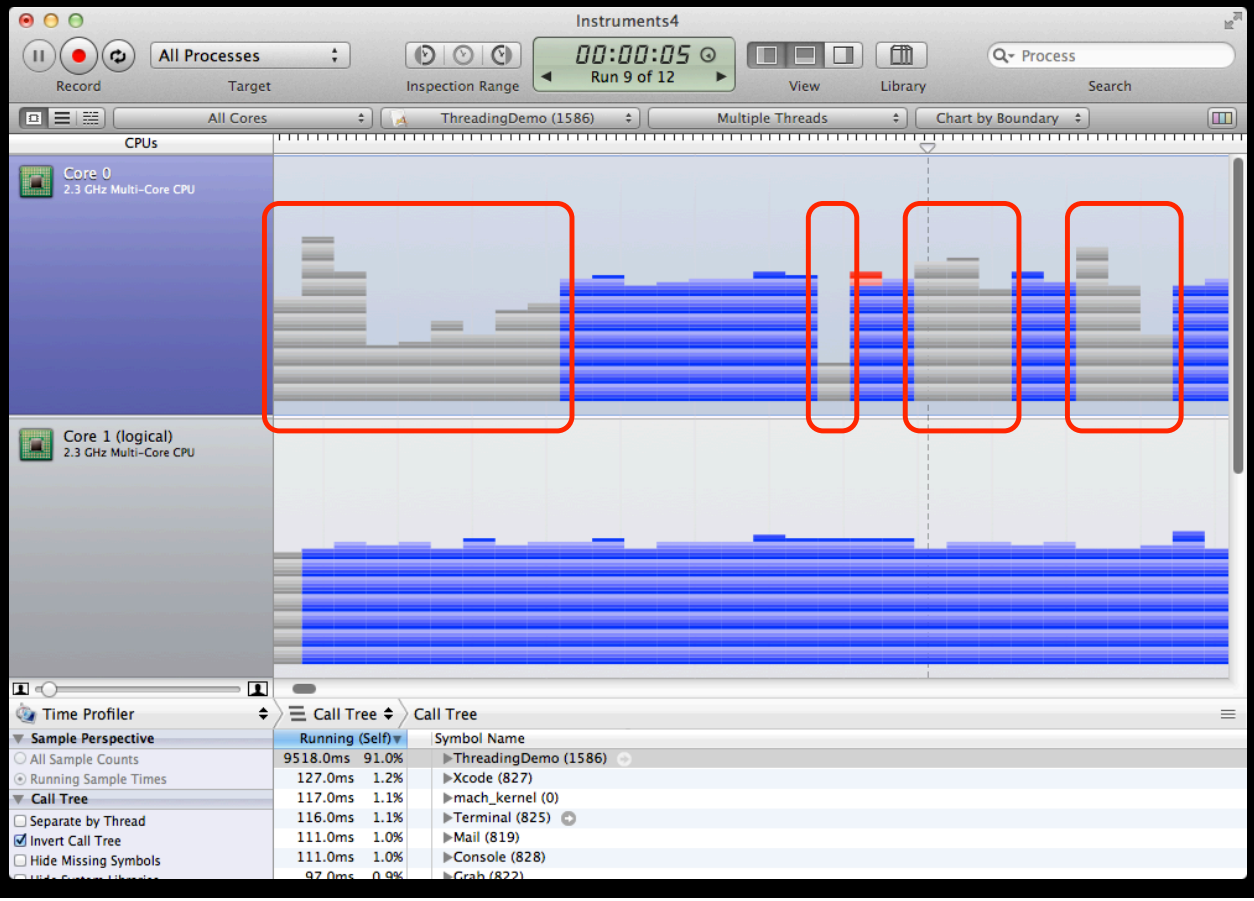
Concurrency
Samples on multiple CPUs at the same time



Concurrency in the Real World

Lock contention eliminated with shared CPU utilization

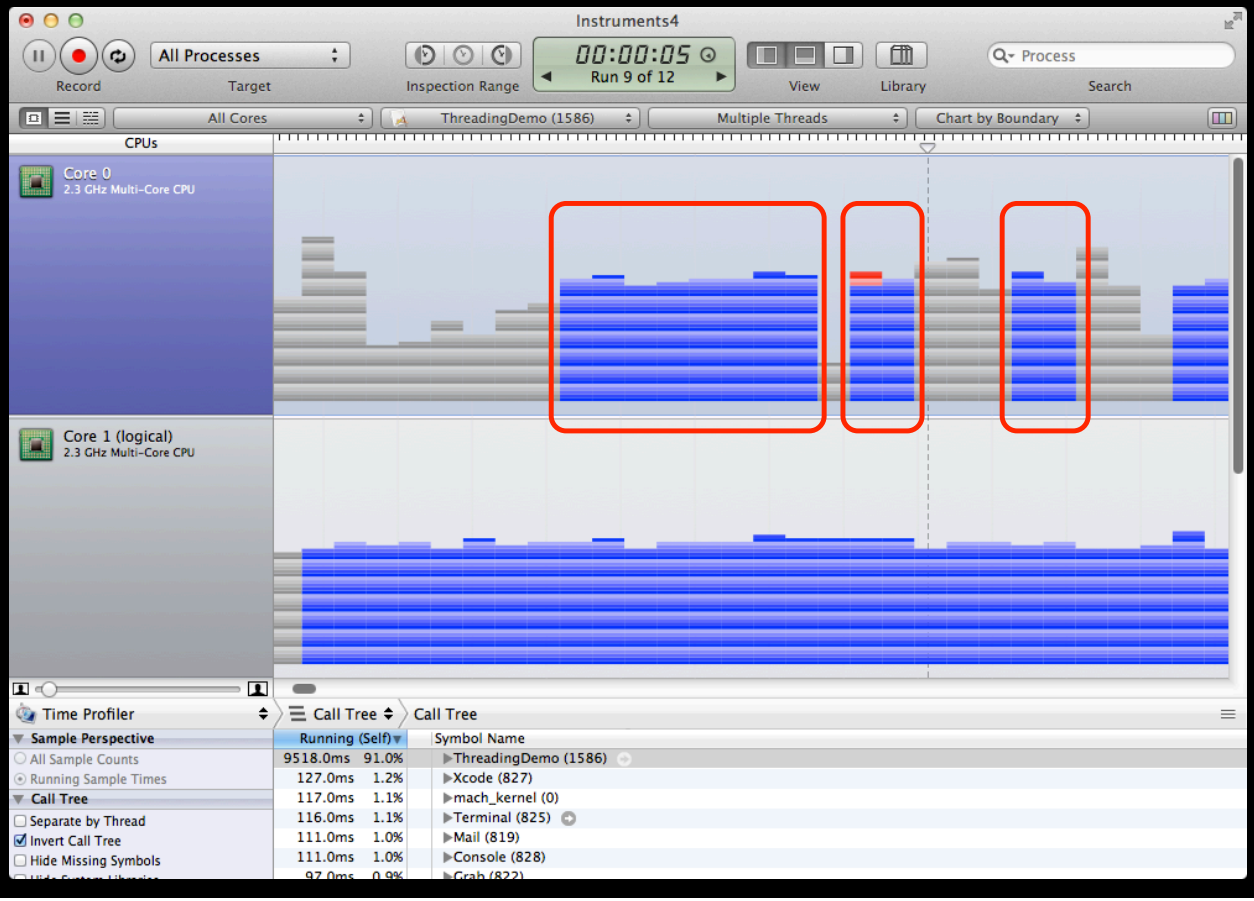
Multi-process Utilization
Samples from multiple processes interleaved



Concurrency in the Real World

Lock contention eliminated with shared CPU utilization

Multiprocess Utilization
Samples from multiple processes interleaved



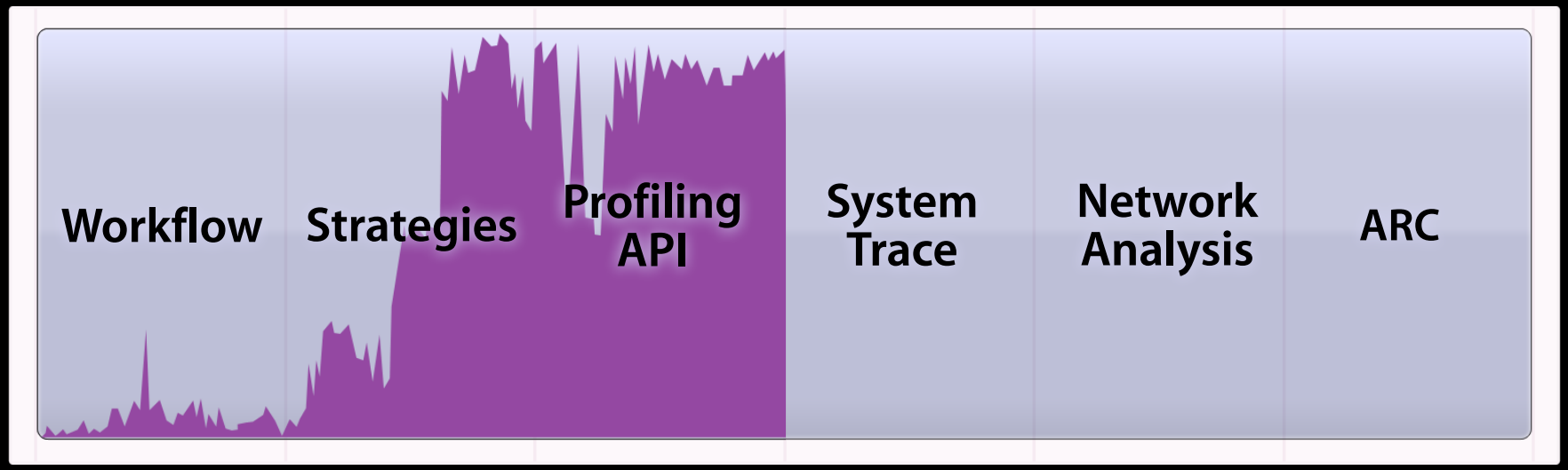
CPU Strategy Demo

Detecting concurrency with time profiler

Daniel Delwood
Performance Tools Engineer

Performance Analysis API

Programmatically control data recording



DTPerformanceSession Framework

Only on Mac OS

- Target self or other processes (existing or launched)
- Time Profiler, System Trace, Leaks, Allocations (with zombies support), Activity Monitor, flags support
- View result in Instruments
  - 'DTPS' file has limited lifespan
    - Symbol data will be out of date once binary is rebuilt
    - Opening in Instruments converts to Trace Document and saves symbol data
- Location

`/Library/Developer/4.0/Instruments/Frameworks/DTPerformanceSession.framework`

Why Use DTPerformanceSession?

- Profile specific operations in your own code
- Automatically flag important events
- Profile your own performance regression tests

Using DTPerformanceSession API

Setup session

```
DTPerformanceSessionCreate(NULL,
      CFStringCreateWithFormat(NULL, NULL, CFSTR("%d"), getpid()),
      NULL, NULL);
DTPerformanceSessionAddInstrument(session,
      CFSTR(DTPerformanceSession_TimeProfiler), NULL, NULL, NULL);
```

Using DTPerformanceSession API

Add calls to start/stop profilers

```
DTPerformanceSessionStart(session, NULL, NULL);
DTPerformanceSessionStop(session, NULL, NULL);
```

Using DTPerformanceSession API

Insert sign posts

```
DTSendSignalFlag("All masterImages updated", DT_POINT_SIGNAL, true);
```

or…

```
DTSendSignalFlag("Master Images Update", DT_START_SIGNAL, true);
DTSendSignalFlag("Master Images Update", DT_END_SIGNAL, true);
```

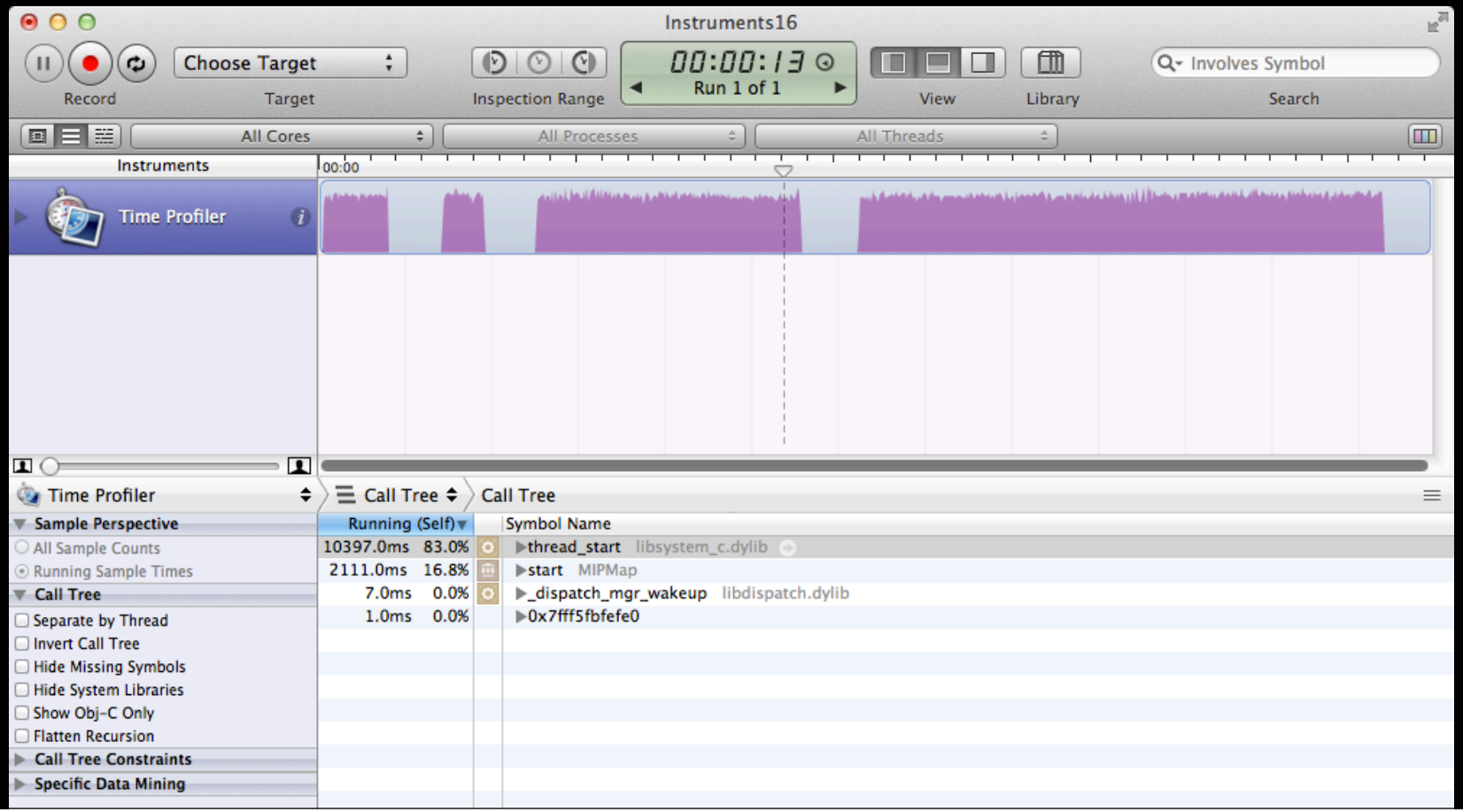
Using DTPerformanceSession API

Save session

```
DTPerformanceSessionSave(session, CFSTR("/tmp/WWDC2011"), NULL);
CFRelease(session);
```

Viewing Results

Open DTPS file in Instruments



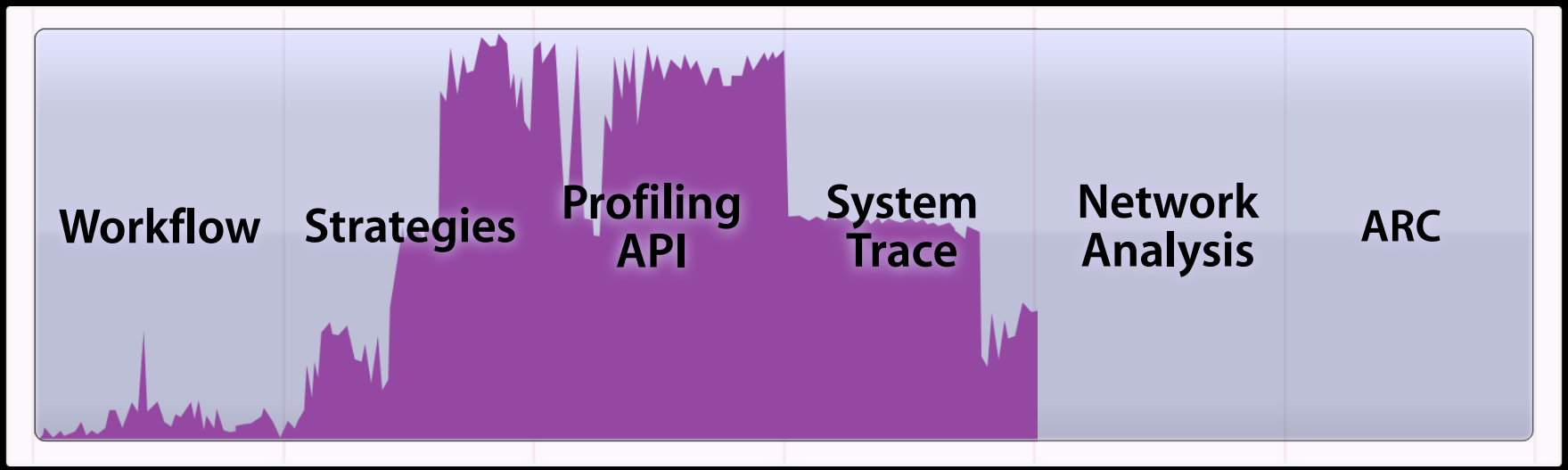
iProfiler

Built with DTPerformanceSession framework

```
iprofiler(1)              BSD General Commands Manual             iprofiler(1)

NAME
     iprofiler, version 1.0

USAGE
     iprofiler [-l] [-L] [-legacy] [-T duration] [-I sampling interval] [-d path] [-o basename]
       [-activitymonitor] [-allocations] [-leaks] [-systemtrace] [-timeprofiler] [-kernelstacks
       | -userandkernelstacks] [-allthreadstates] [-a process/pid | executable [args...]]

DESCRIPTION
    Measure an application's performance without launching Instruments.app and then visualize
    the measurements at a later time in Instruments.app.  The performance data gets saved in a
    .dtps bundle that can be opened in Instruments.app via "Open existing file...".  iprofiler
    supports these instruments: Time Profiler, System Trace, Activity Monitor, Allocations,
    and Leaks.  Any combination of these instruments can be run simultaneously.  iprofiler
    supports attaching to a currently-running process, launching a process that will only run
    during the measurement, or profiling all currently-running processes (by not specifying
    process/pid or executable).

     Options are :

     -l       Lists all supported instruments

     -L       Lists all supported instruments, with a description
```

System Trace

Comprehensive system analysis



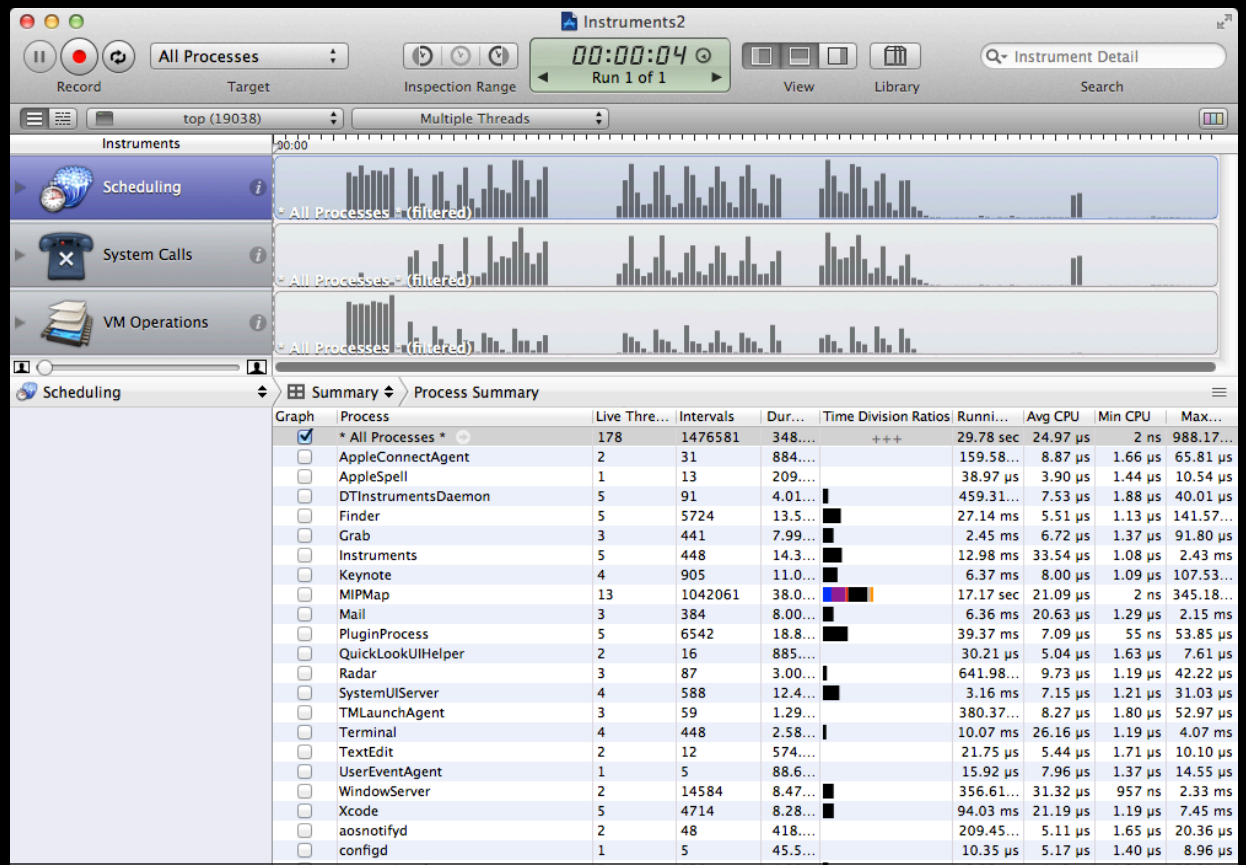
What Is System Trace?

- NEW in iOS 5!
- Comprehensive analysis of system behavior
  - Thread scheduling
  - Virtual memory
  - System calls
- Available in Mac OS X since Instruments 4.0

System Trace

“Instruments” strategy

Scheduling
Records thread context switches and tenures

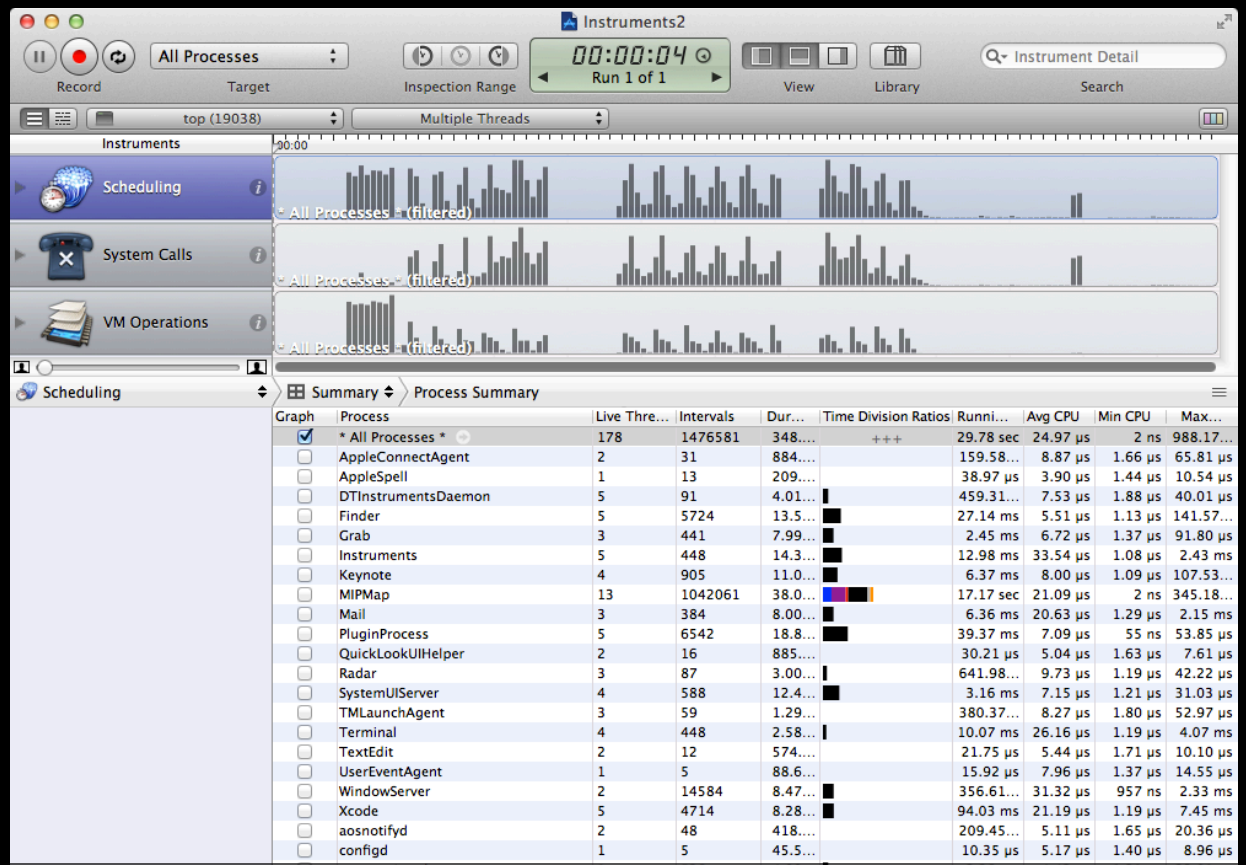


Graph	Process	Live Thre...	Intervals	Dur...	Time Division Ratios	Runni...	Avg CPU	Min CPU	Max...
☑	* All Processes *	178	1476581	348....	+++	29.78 sec	24.97 µs	2 ns	988.17...
☐	AppleConnectAgent	2	31	884....		159.58...	8.87 µs	1.66 µs	65.81 µs
☐	AppleSpell	1	13	209....		38.97 µs	3.90 µs	1.44 µs	10.54 µs
☐	DTInstrumentsDaemon	5	91	4.01...		459.31...	7.53 µs	1.88 µs	40.01 µs
☐	Finder	5	5724	13.5...		27.14 ms	5.51 µs	1.13 µs	141.57...
☐	Grab	3	441	7.99...		2.45 ms	6.72 µs	1.37 µs	91.80 µs
☐	Instruments	5	448	14.3...		12.98 ms	33.54 µs	1.08 µs	2.43 ms
☐	Keynote	4	905	11.0...		6.37 ms	8.00 µs	1.09 µs	107.53...
☐	MIPMap	13	1042061	38.0...		17.17 sec	21.09 µs	2 ns	345.18...
☐	Mail	3	384	8.00...		6.36 ms	20.63 µs	1.29 µs	2.15 ms
☐	PluginProcess	5	6542	18.8...		39.37 ms	7.09 µs	55 ns	53.85 µs
☐	QuickLookUIHelper	2	16	885....		30.21 µs	5.04 µs	1.63 µs	7.61 µs
☐	Radar	3	87	3.00...		641.98...	9.73 µs	1.19 µs	42.22 µs
☐	SystemUIServer	4	588	12.4...		3.16 ms	7.15 µs	1.21 µs	31.03 µs
☐	TMLaunchAgent	3	59	1.29...		380.37...	8.27 µs	1.80 µs	52.97 µs
☐	Terminal	4	448	2.58...		10.07 ms	26.16 µs	1.19 µs	4.07 ms
☐	TextEdit	2	12	574....		21.75 µs	5.44 µs	1.71 µs	10.10 µs
☐	UserEventAgent	1	5	88.6...		15.92 µs	7.96 µs	1.37 µs	14.55 µs
☐	WindowServer	2	14584	8.47...		356.61...	31.32 µs	957 ns	2.33 ms
☐	Xcode	5	4714	8.28...		94.03 ms	21.19 µs	1.19 µs	7.45 ms
☐	aosnotifyd	2	48	418....		209.45...	5.11 µs	1.65 µs	20.36 µs
☐	configd	1	5	45.5...		10.35 µs	5.17 µs	1.40 µs	8.96 µs

System Trace

“Instruments” strategy

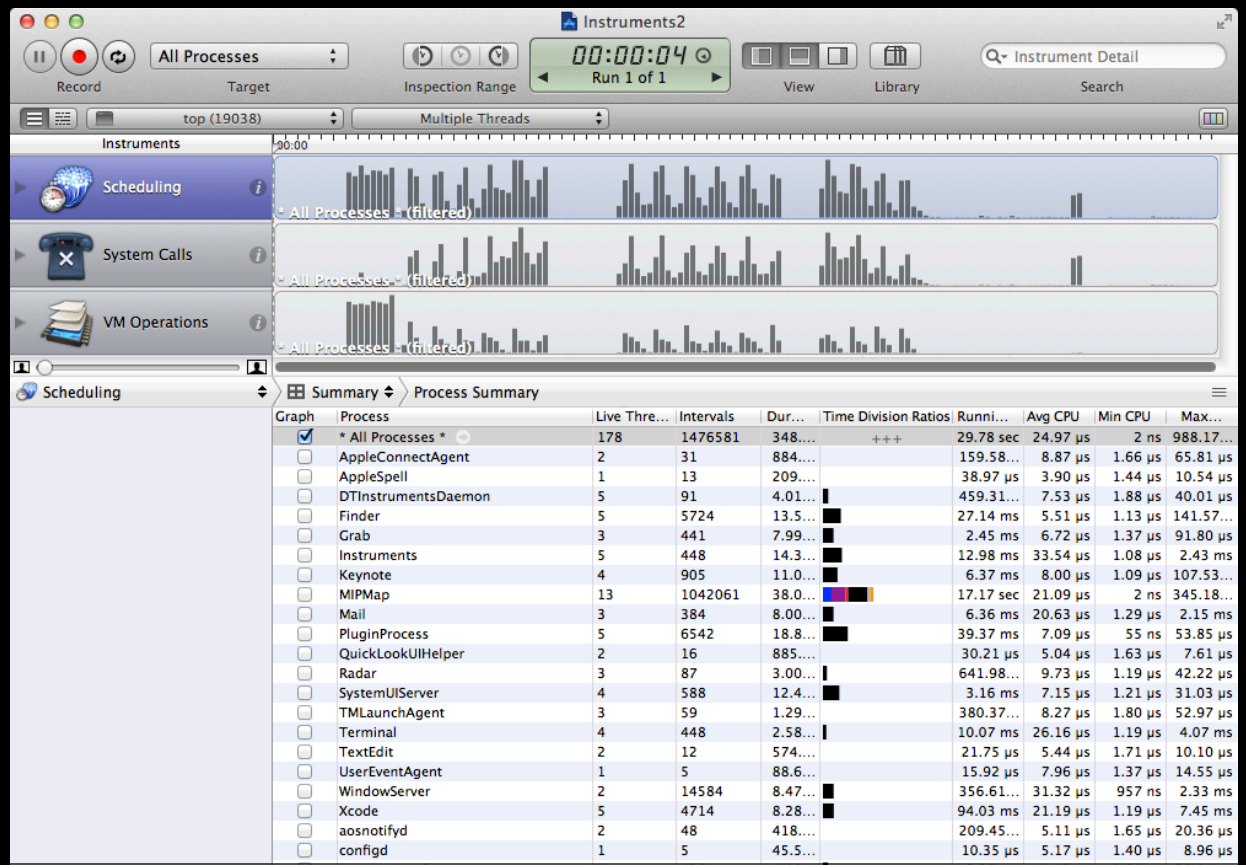
System Calls
Records system calls
and their duration



System Trace

“Instruments” strategy

VM Operations
Records virtual memory fault events

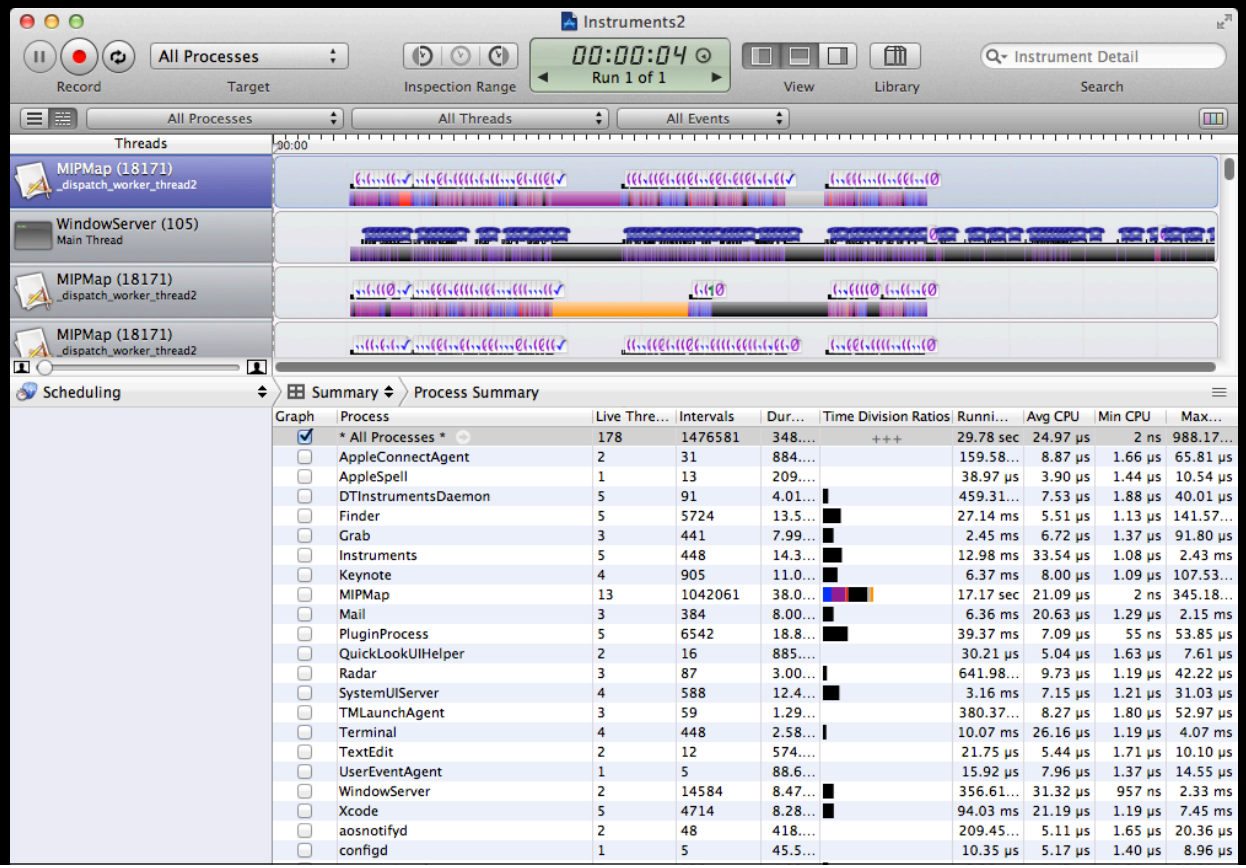


Graph	Process	Live Thre...	Intervals	Dur...	Time Division Ratios	Runni...	Avg CPU	Min CPU	Max...
☑	* All Processes *	178	1476581	348....	+++	29.78 sec	24.97 µs	2 ns	988.17...
☐	AppleConnectAgent	2	31	884....		159.58...	8.87 µs	1.66 µs	65.81 µs
☐	AppleSpell	1	13	209....		38.97 µs	3.90 µs	1.44 µs	10.54 µs
☐	DTInstrumentsDaemon	5	91	4.01...		459.31...	7.53 µs	1.88 µs	40.01 µs
☐	Finder	5	5724	13.5...		27.14 ms	5.51 µs	1.13 µs	141.57...
☐	Grab	3	441	7.99...		2.45 ms	6.72 µs	1.37 µs	91.80 µs
☐	Instruments	5	448	14.3...		12.98 ms	33.54 µs	1.08 µs	2.43 ms
☐	Keynote	4	905	11.0...		6.37 ms	8.00 µs	1.09 µs	107.53...
☐	MIPMap	13	1042061	38.0...		17.17 sec	21.09 µs	2 ns	345.18...
☐	Mail	3	384	8.00...		6.36 ms	20.63 µs	1.29 µs	2.15 ms
☐	PluginProcess	5	6542	18.8...		39.37 ms	7.09 µs	55 ns	53.85 µs
☐	QuickLookUIHelper	2	16	885....		30.21 µs	5.04 µs	1.63 µs	7.61 µs
☐	Radar	3	87	3.00...		641.98...	9.73 µs	1.19 µs	42.22 µs
☐	SystemUIServer	4	588	12.4...		3.16 ms	7.15 µs	1.21 µs	31.03 µs
☐	TMLaunchAgent	3	59	1.29...		380.37...	8.27 µs	1.80 µs	52.97 µs
☐	Terminal	4	448	2.58...		10.07 ms	26.16 µs	1.19 µs	4.07 ms
☐	TextEdit	2	12	574....		21.75 µs	5.44 µs	1.71 µs	10.10 µs
☐	UserEventAgent	1	5	88.6...		15.92 µs	7.96 µs	1.37 µs	14.55 µs
☐	WindowServer	2	14584	8.47...		356.61...	31.32 µs	957 ns	2.33 ms
☐	Xcode	5	4714	8.28...		94.03 ms	21.19 µs	1.19 µs	7.45 ms
☐	aosnotifyd	2	48	418....		209.45...	5.11 µs	1.65 µs	20.36 µs
☐	configd	1	5	45.5...		10.35 µs	5.17 µs	1.40 µs	8.96 µs

System Trace

"Threads" strategy

Threads
System calls, VM faults, and thread states

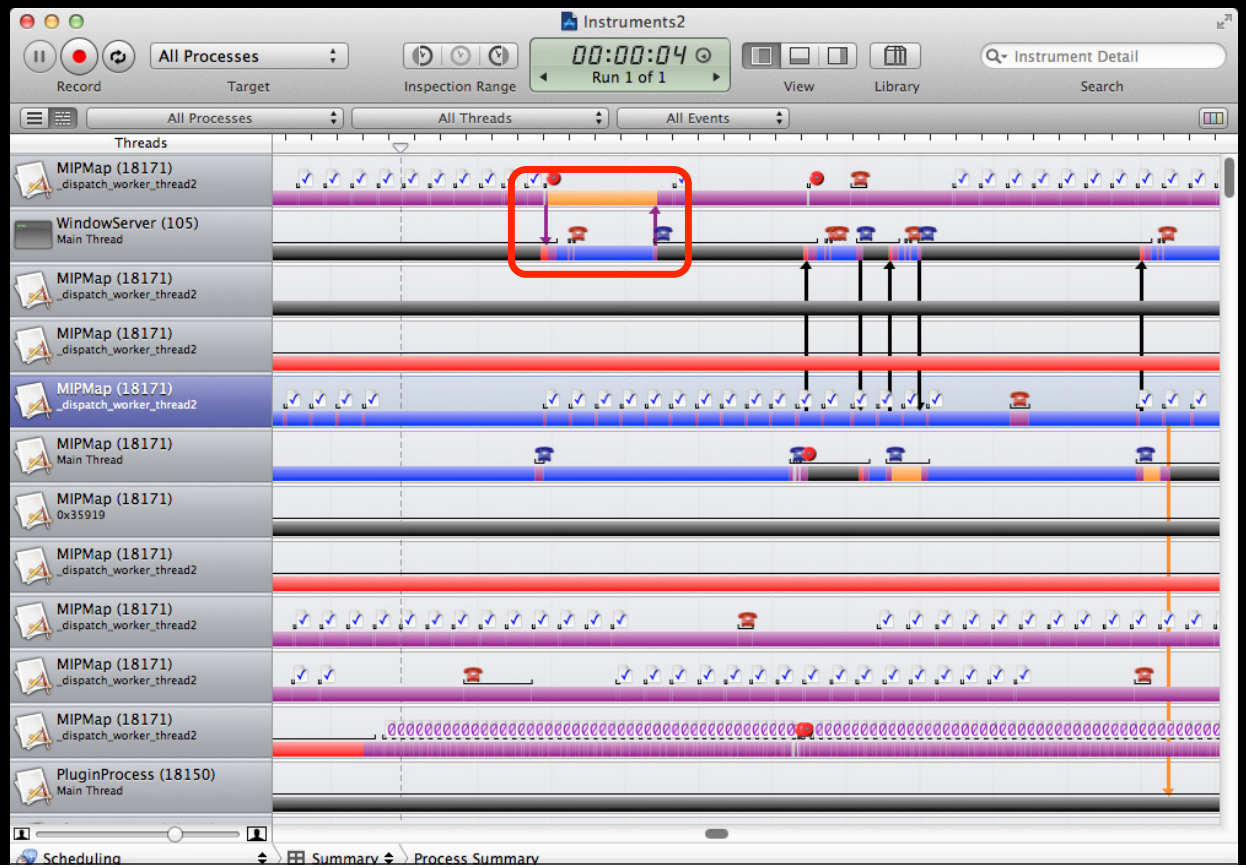


Graph	Process	Live Thre...	Intervals	Dur...	Time Division Ratios	Runni...	Avg CPU	Min CPU	Max...
☑	* All Processes *	178	1476581	348....	+++	29.78 sec	24.97 µs	2 ns	988.17...
☐	AppleConnectAgent	2	31	884....		159.58...	8.87 µs	1.66 µs	65.81 µs
☐	AppleSpell	1	13	209....		38.97 µs	3.90 µs	1.44 µs	10.54 µs
☐	DTInstrumentsDaemon	5	91	4.01...		459.31...	7.53 µs	1.88 µs	40.01 µs
☐	Finder	5	5724	13.5...		27.14 ms	5.51 µs	1.13 µs	141.57...
☐	Grab	3	441	7.99...		2.45 ms	6.72 µs	1.37 µs	91.80 µs
☐	Instruments	5	448	14.3...		12.98 ms	33.54 µs	1.08 µs	2.43 ms
☐	Keynote	4	905	11.0...		6.37 ms	8.00 µs	1.09 µs	107.53...
☐	MIPMap	13	1042061	38.0...		17.17 sec	21.09 µs	2 ns	345.18...
☐	Mail	3	384	8.00...		6.36 ms	20.63 µs	1.29 µs	2.15 ms
☐	PluginProcess	5	6542	18.8...		39.37 ms	7.09 µs	55 ns	53.85 µs
☐	QuickLookUIHelper	2	16	885....		30.21 µs	5.04 µs	1.63 µs	7.61 µs
☐	Radar	3	87	3.00...		641.98...	9.73 µs	1.19 µs	42.22 µs
☐	SystemUIServer	4	588	12.4...		3.16 ms	7.15 µs	1.21 µs	31.03 µs
☐	TMLaunchAgent	3	59	1.29...		380.37...	8.27 µs	1.80 µs	52.97 µs
☐	Terminal	4	448	2.58...		10.07 ms	26.16 µs	1.19 µs	4.07 ms
☐	TextEdit	2	12	574....		21.75 µs	5.44 µs	1.71 µs	10.10 µs
☐	UserEventAgent	1	5	88.6...		15.92 µs	7.96 µs	1.37 µs	14.55 µs
☐	WindowServer	2	14584	8.47...		356.61...	31.32 µs	957 ns	2.33 ms
☐	Xcode	5	4714	8.28...		94.03 ms	21.19 µs	1.19 µs	7.45 ms
☐	aosnotifyd	2	48	418....		209.45...	5.11 µs	1.65 µs	20.36 µs
☐	configd	1	5	45.5...		10.35 µs	5.17 µs	1.40 µs	8.96 µs

System Trace

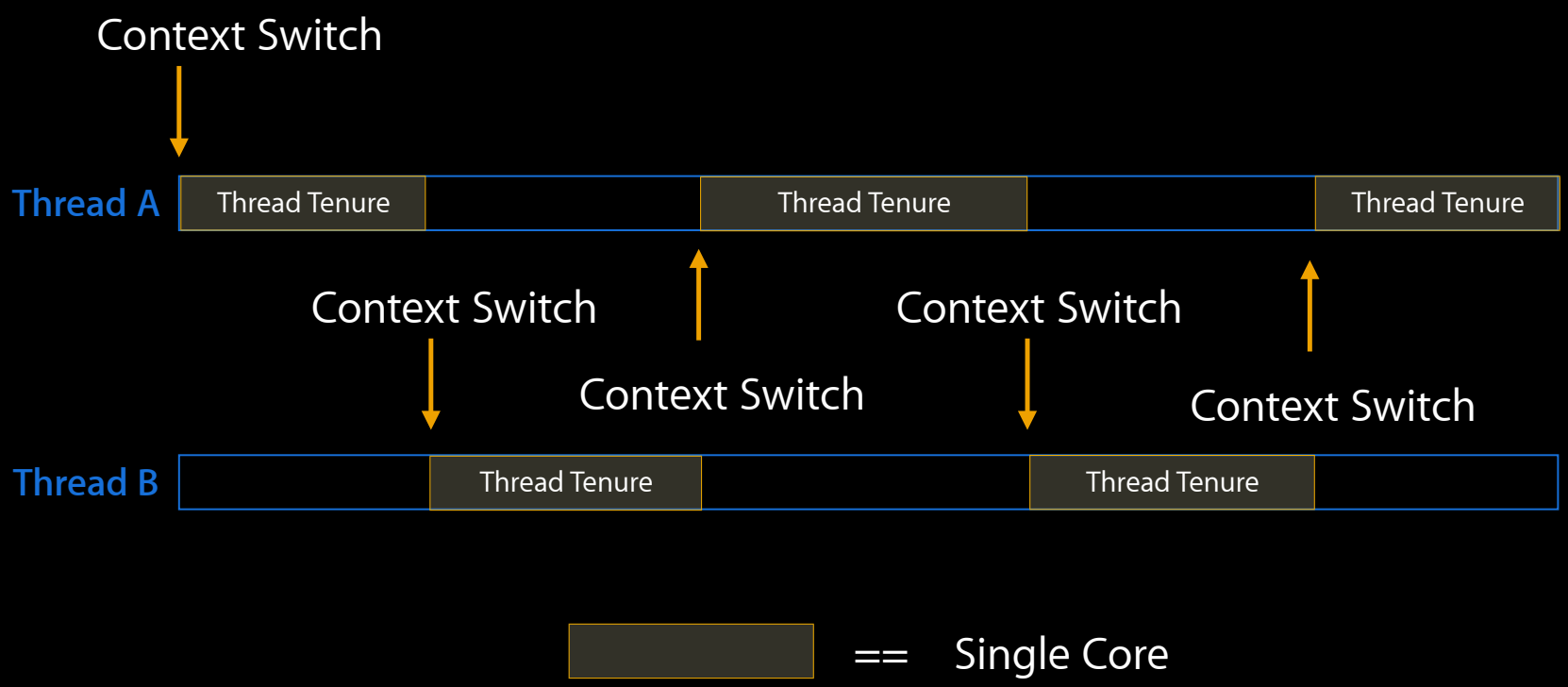
"Threads" strategy

Thread Context Switches
Arrows indicate threads scheduled onto cores



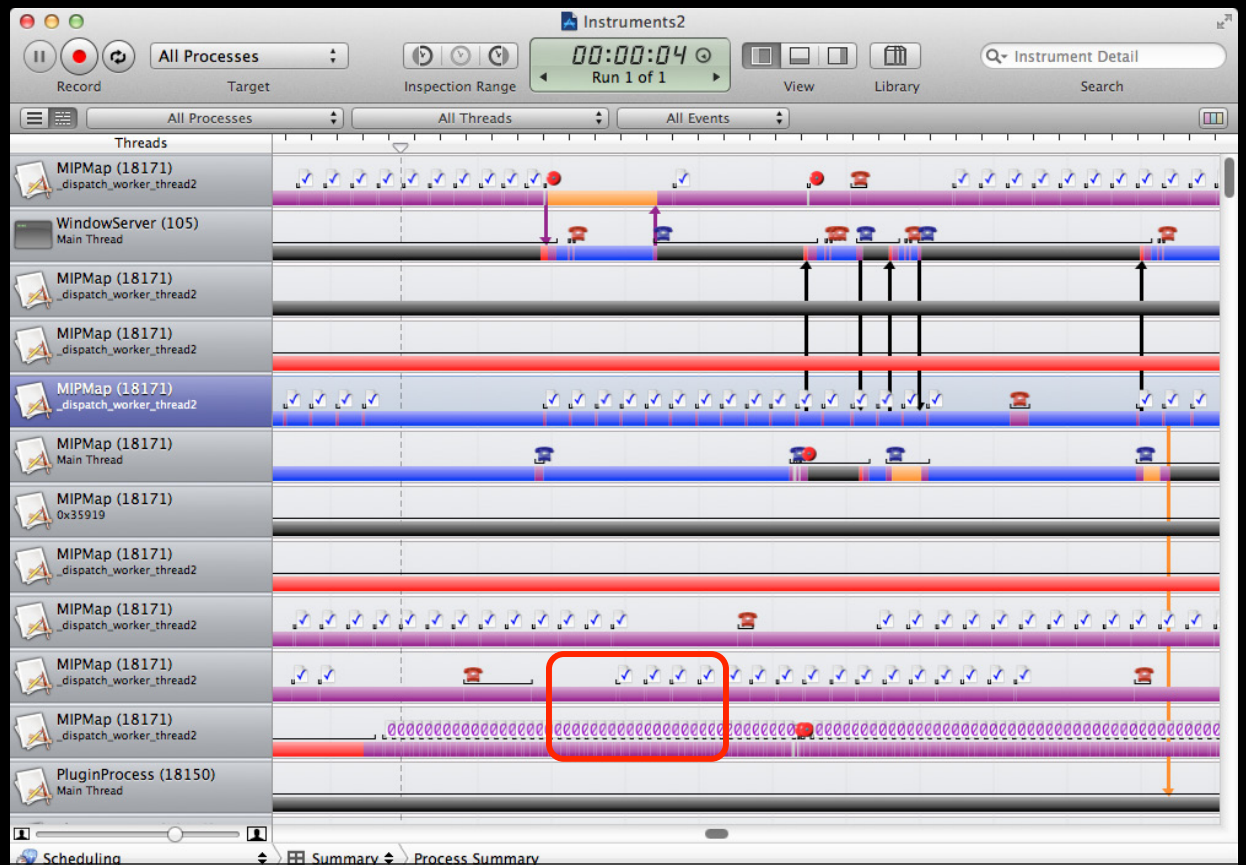
System Trace

Thread context switches



System Trace

"Threads" strategy



Virtual Memory Events
Icons indicate cache hits, zero fills, page-ins, etc.

System Trace

Virtual memory events

- Copy On Write
- Zero Fill
- Page-in
- Page Cache Hit
- Non-Zero Fill
- Guard Page
- File Backed Page-in
- Anonymous Memory Page-in
- Page-out

Virtual Memory Events Explained

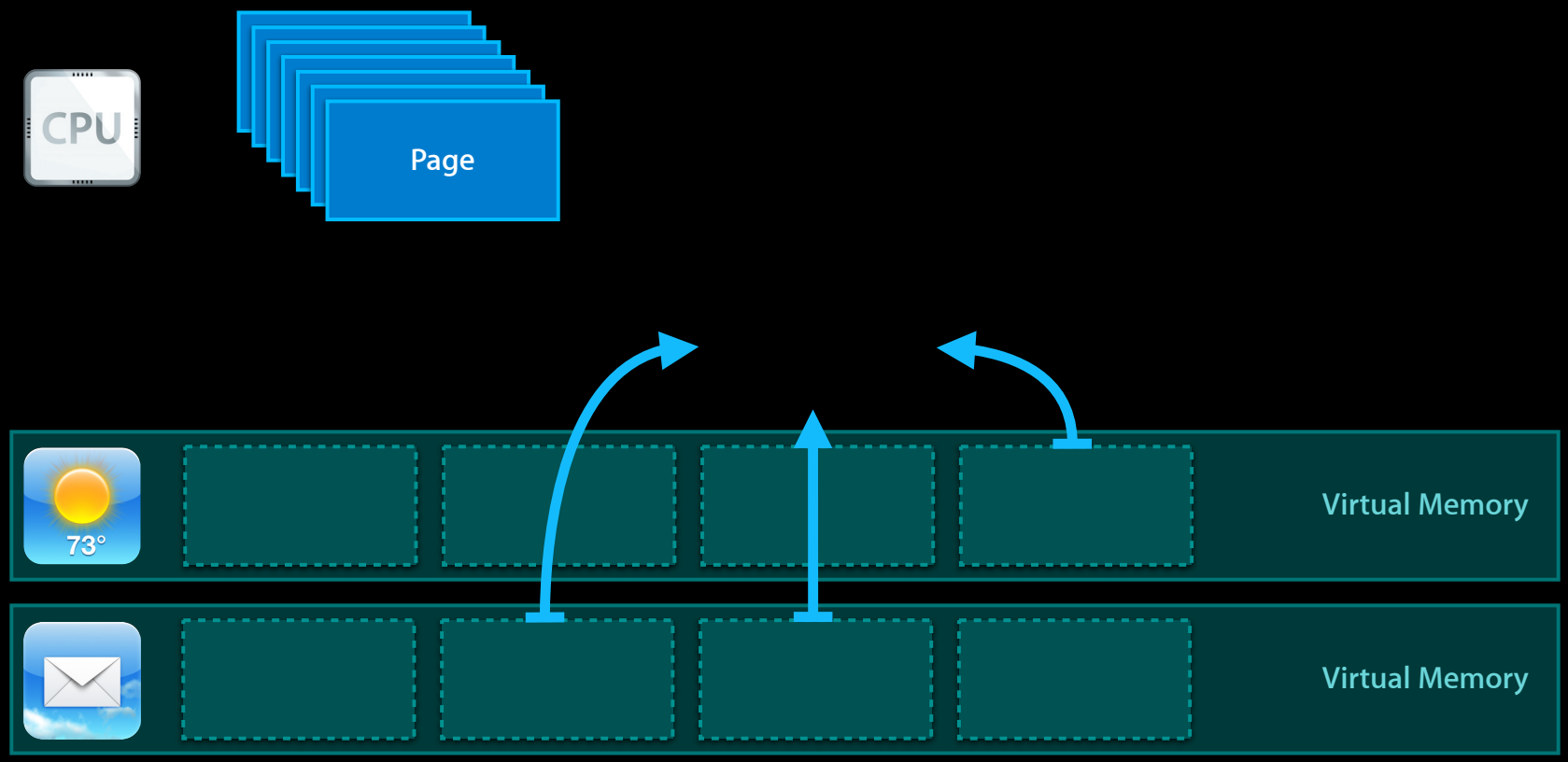
Memory is managed in segments called "Pages"

Virtual Memory Events Explained

Virtual Pages become real Pages by "Page faults"

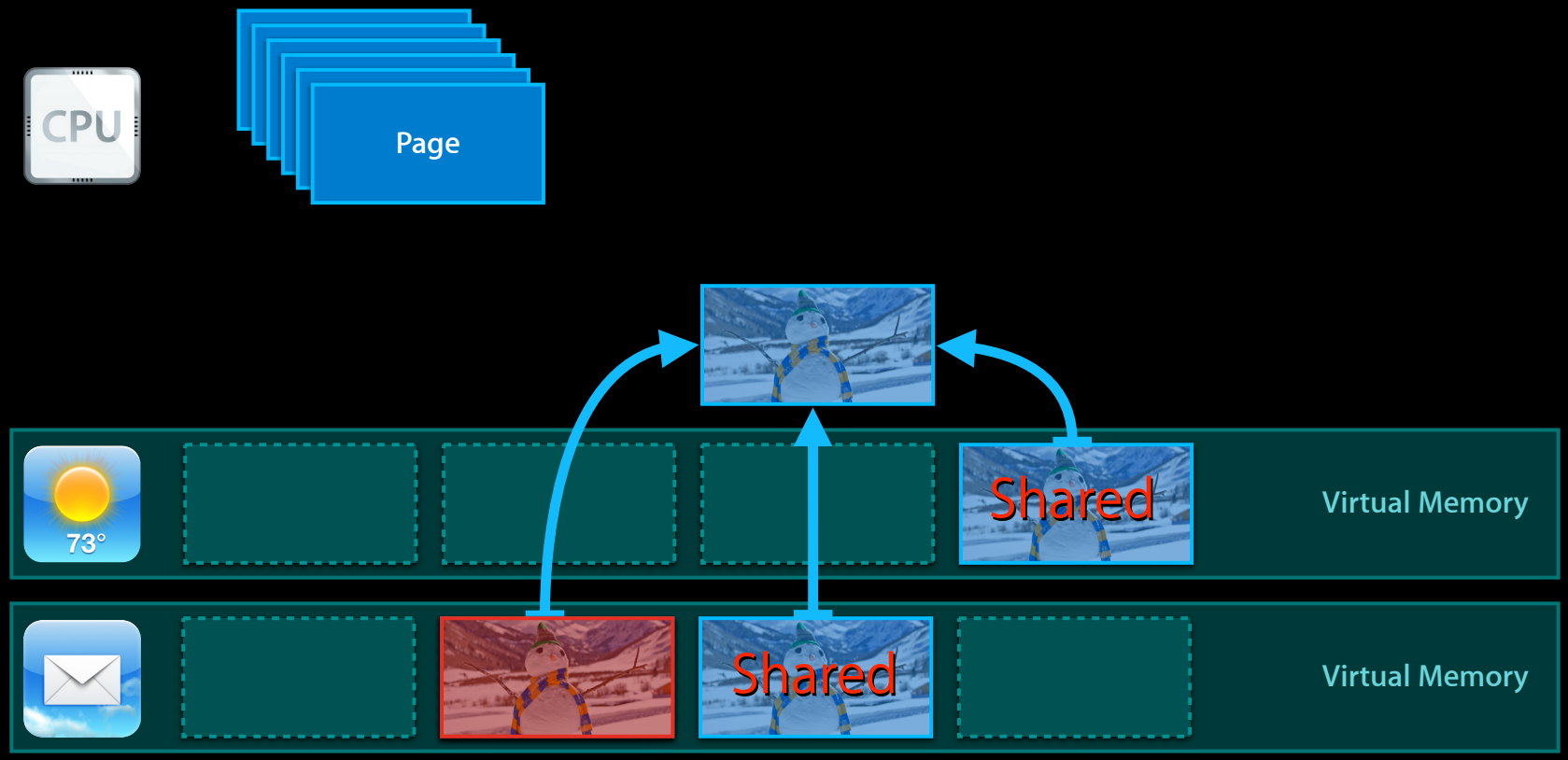
Virtual Memory Events Explained

Pages can be shared



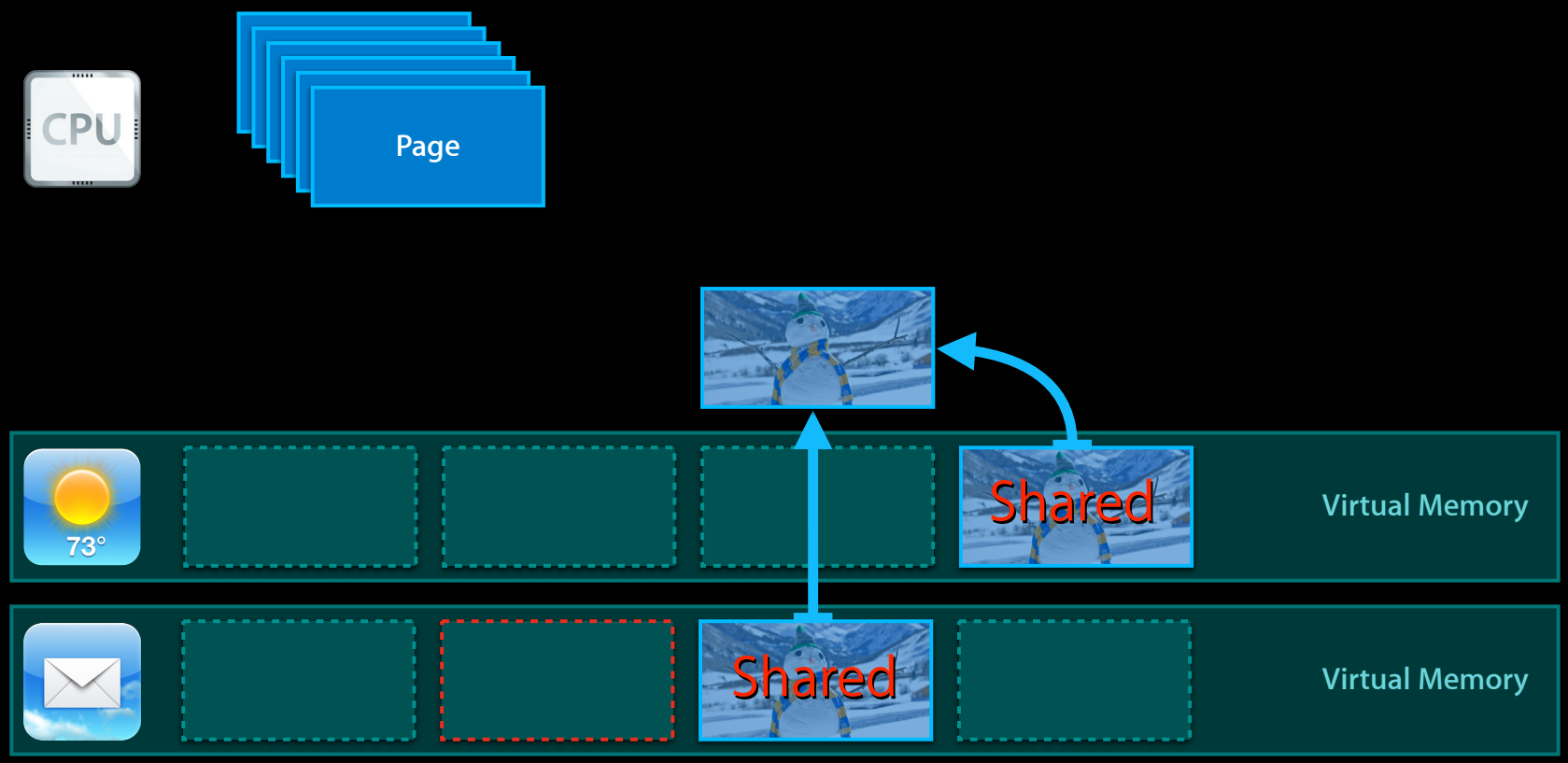
Virtual Memory Events Explained

What happens when someone writes to a shared page?



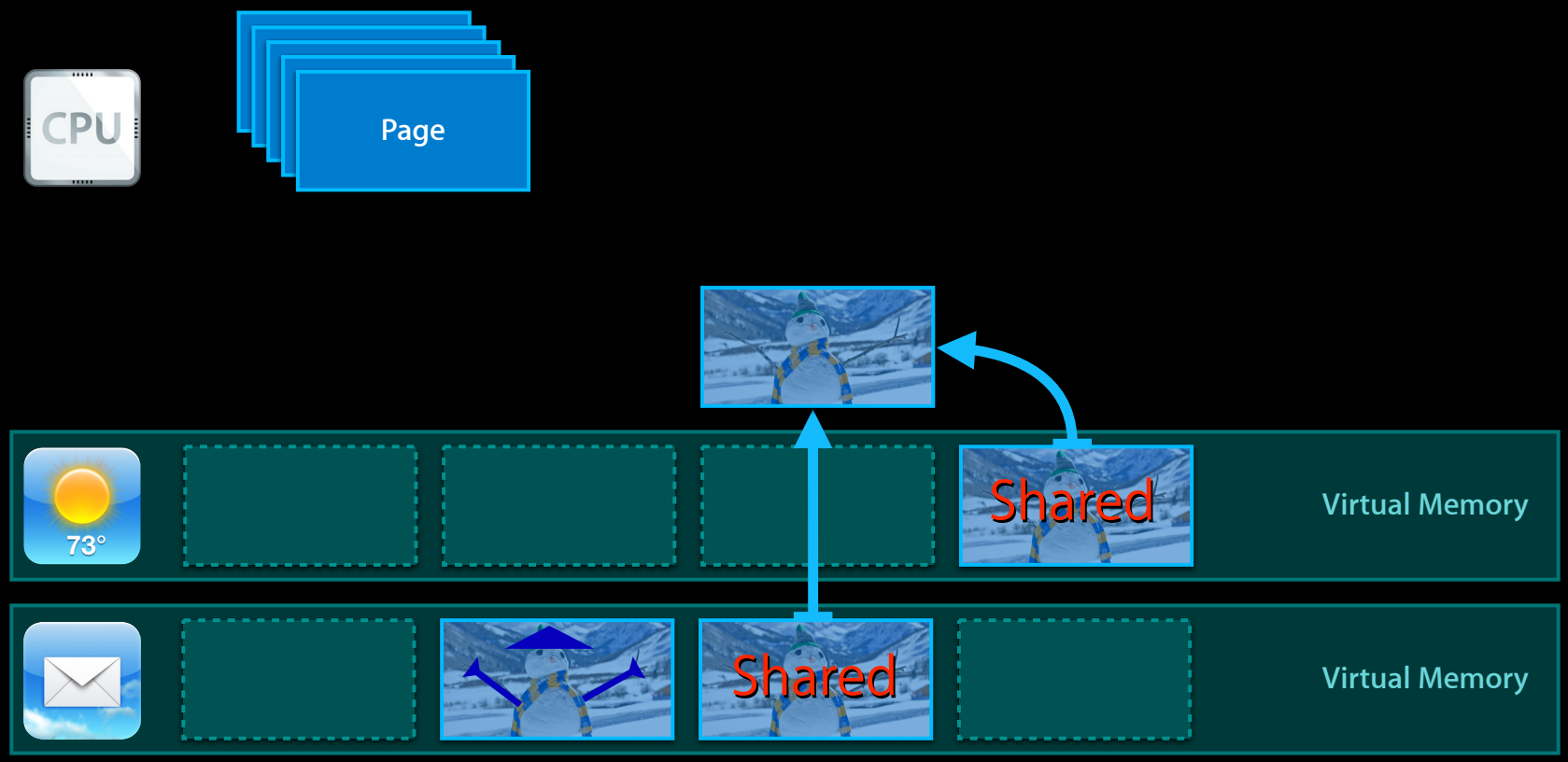
Virtual Memory Events Explained

A copy is made of the shared page



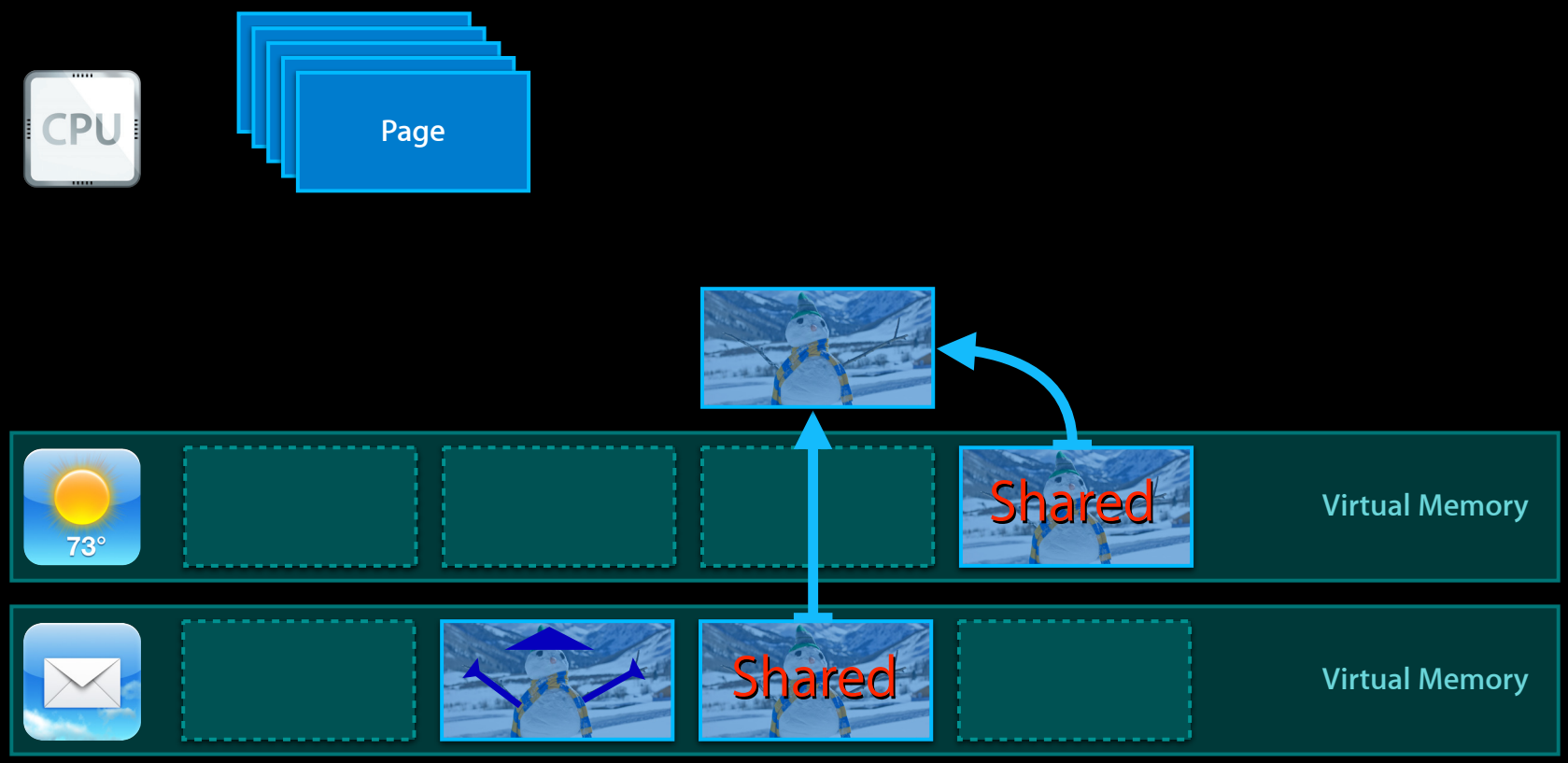
Virtual Memory Events Explained

The write completes, thus a “Copy On Write” page fault



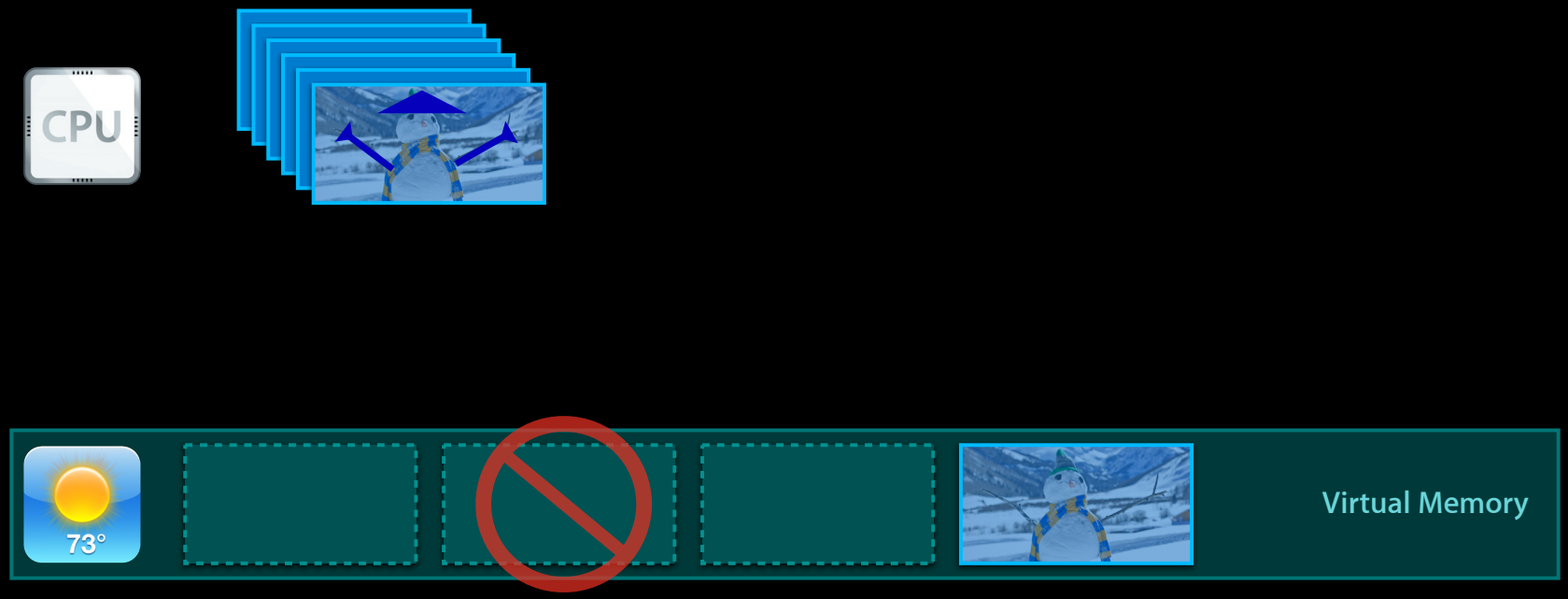
Virtual Memory Events Explained

Pages no longer used are returned to the VM system



Virtual Memory Events Explained

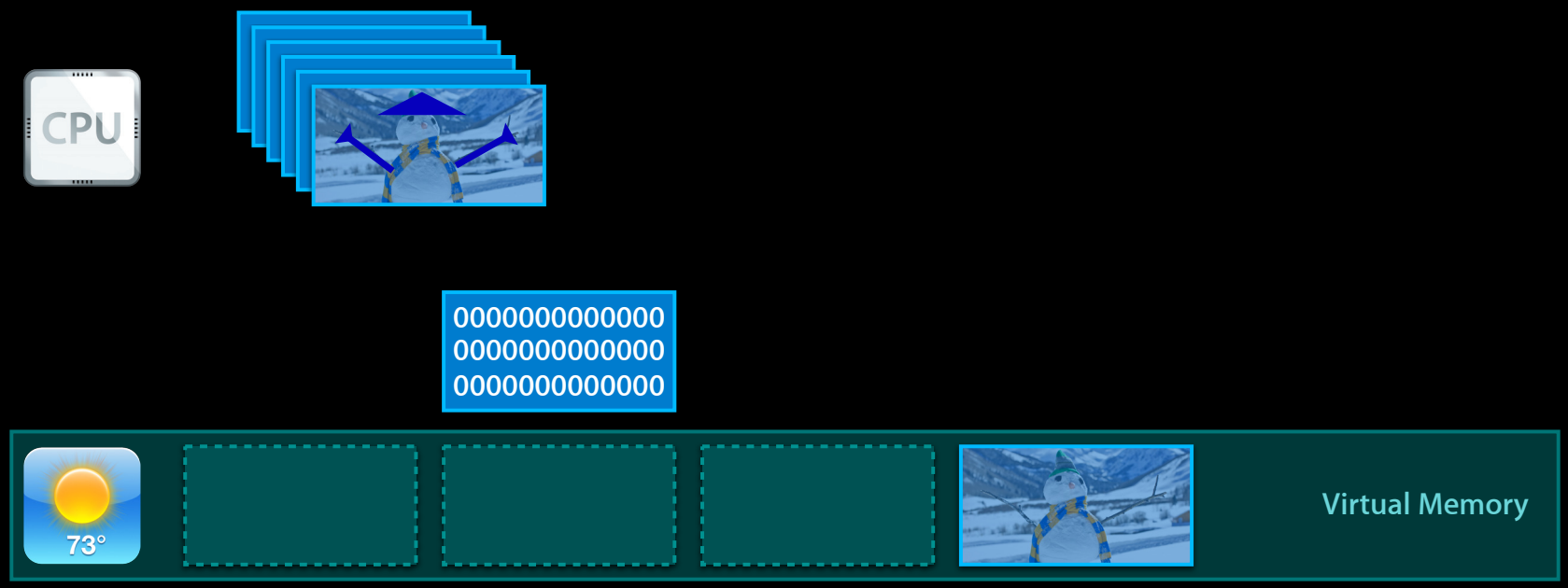
What happens when that memory is reused?



Clearly, this would be a problem!

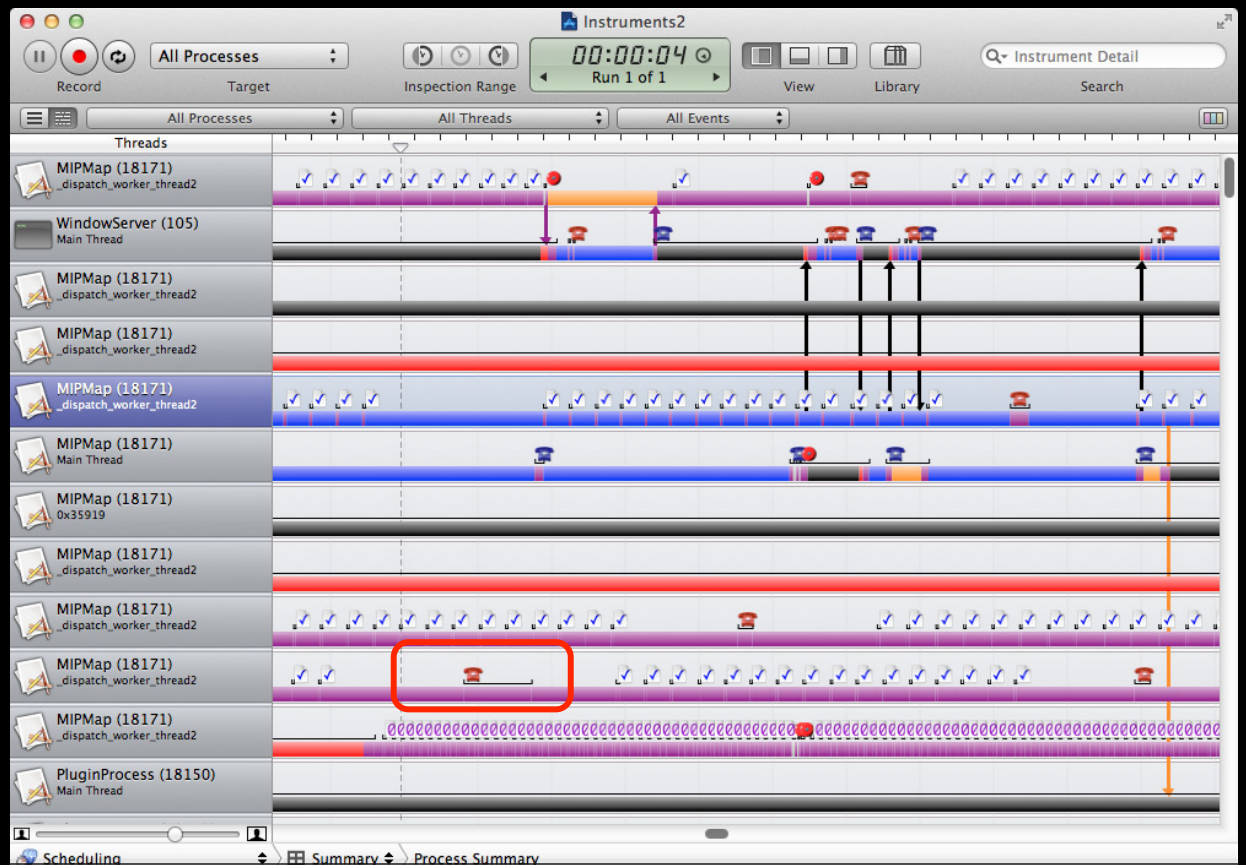
Virtual Memory Events Explained

The system performs a “Zero Fill” page fault



System Trace

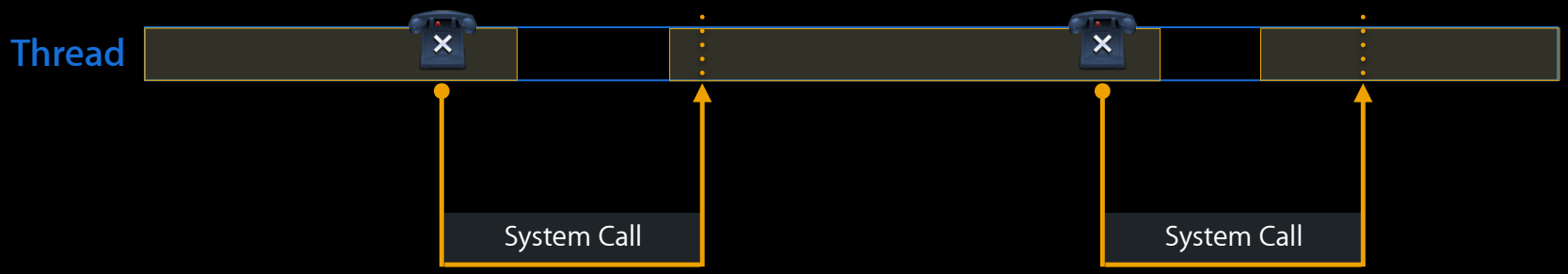
"Threads" strategy



System Calls
Telephones indicate calls from user space into the kernel

System Trace

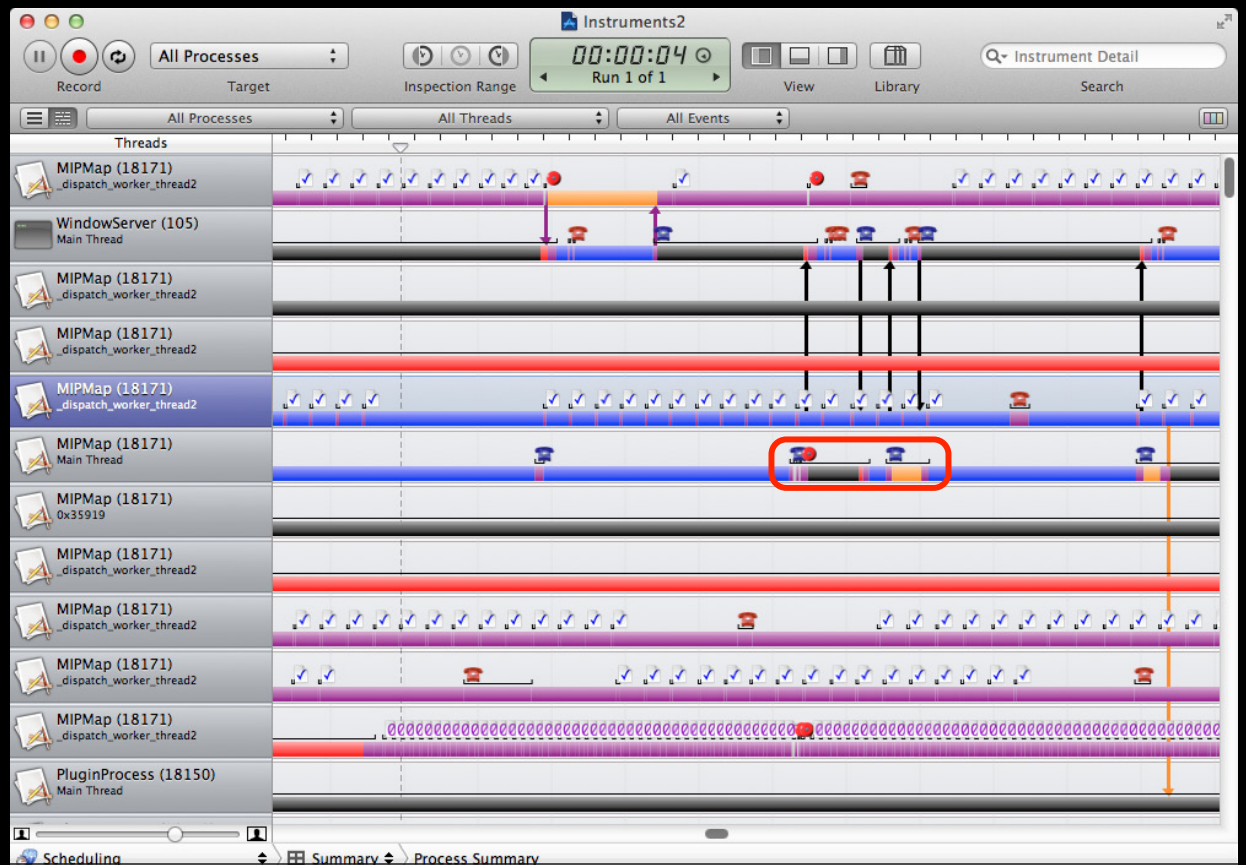
System calls



System Trace

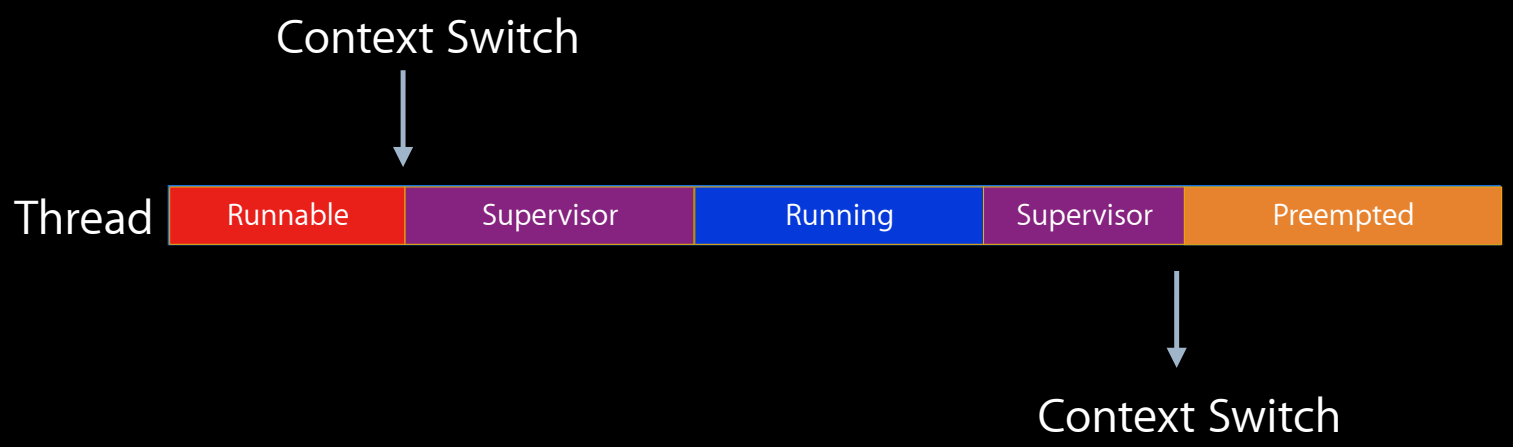
"Threads" strategy

Thread States
Colored bars indicate thread state



System Trace

Thread states

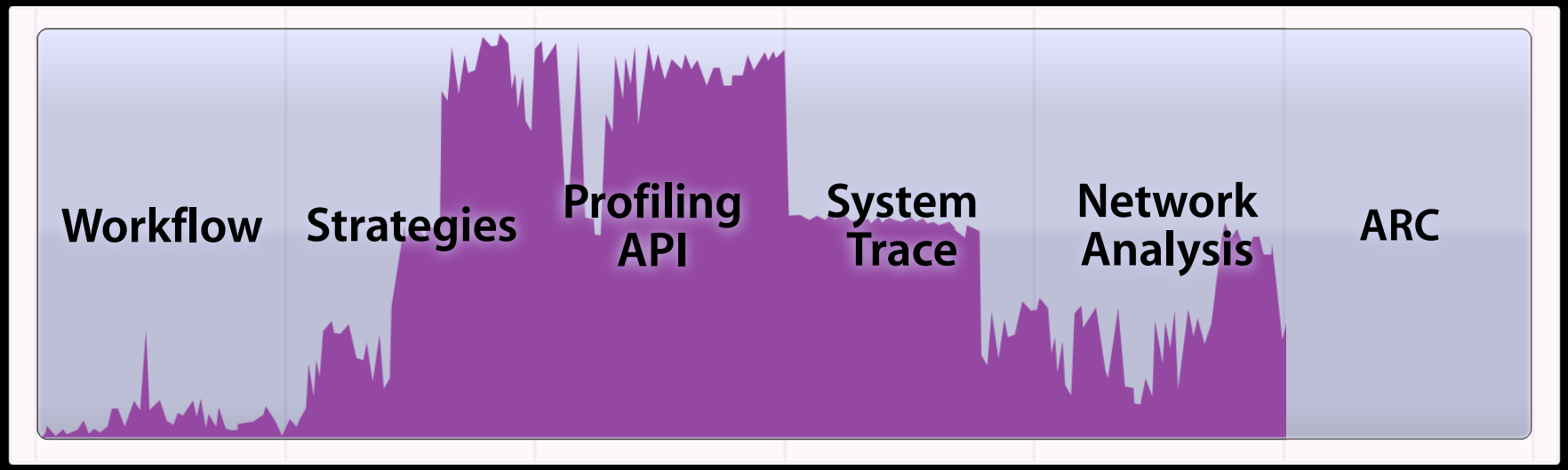


System Trace Demo

Daniel Delwood
Performance Tools Engineer

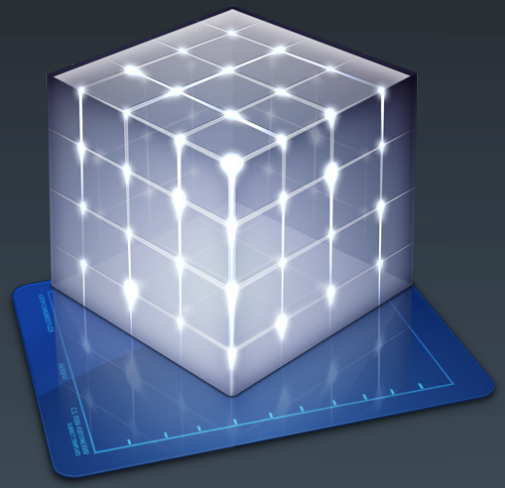
Network Instrumentation

Track data flow and radio usage

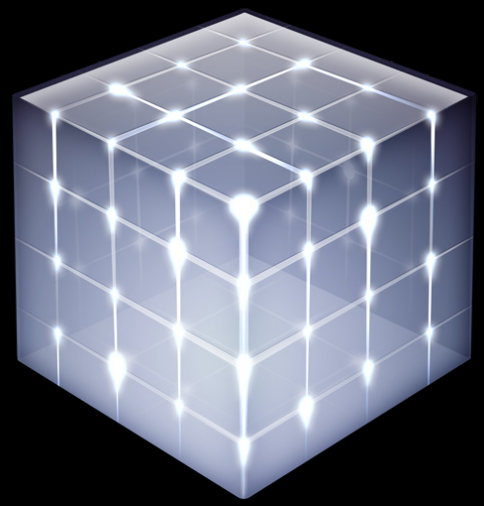


Network Connections

Data flow statistics



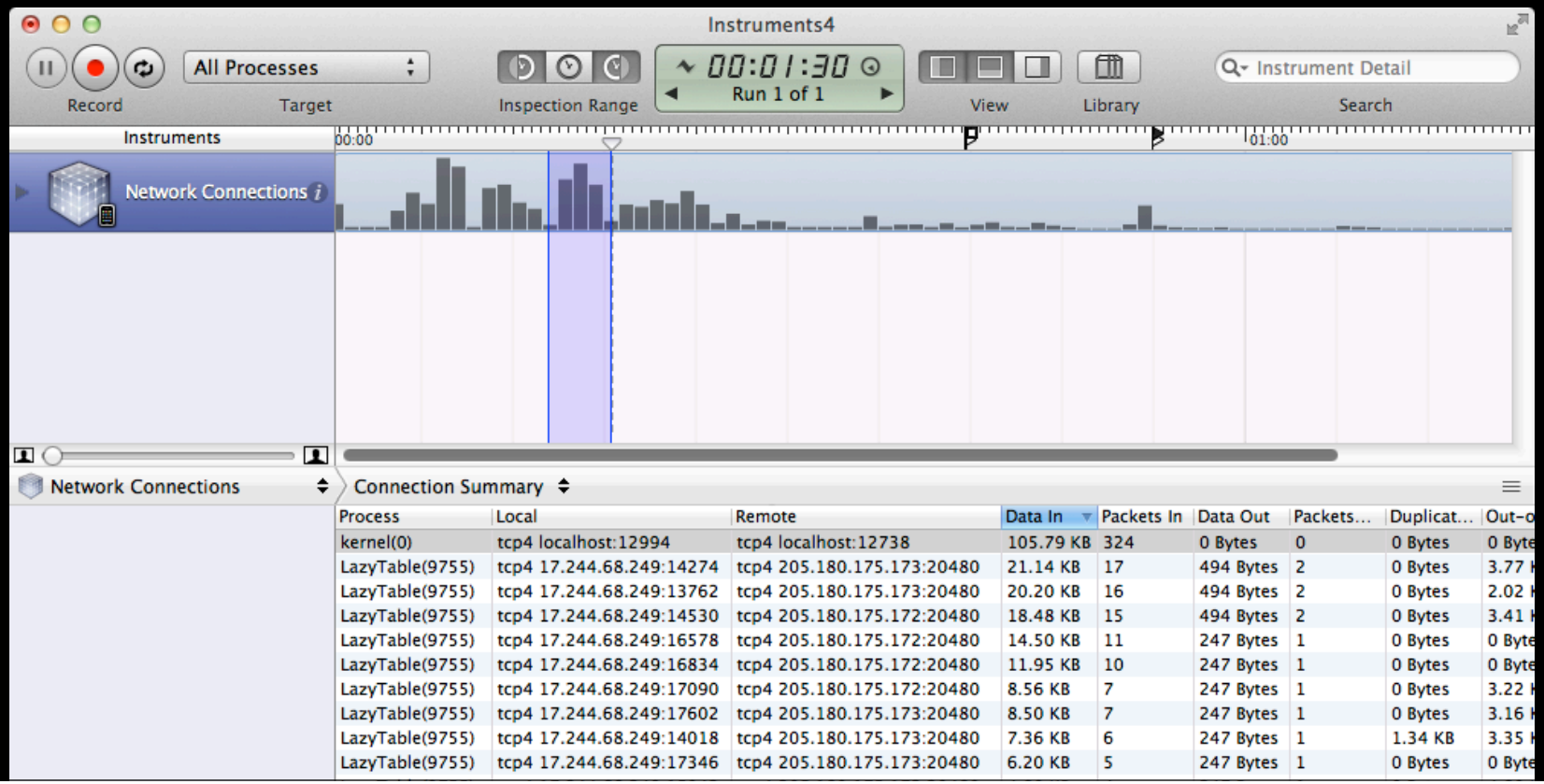
Network Connections Instrument



- Measures data volume
  - TCP/IP and UDP/IP
  - Performance metrics
- Debug latency
- Identify lost packets and re-transmission of data

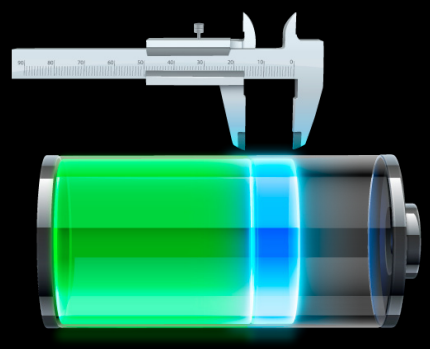
Network Connections Instrument

New in iOS 5



Process	Local	Remote	Data In	Packets In	Data Out	Packets...	Duplicat...	Out-o
kernel(0)	tcp4 localhost:12994	tcp4 localhost:12738	105.79 KB	324	0 Bytes	0	0 Bytes	0 Byte
LazyTable(9755)	tcp4 17.244.68.249:14274	tcp4 205.180.175.173:20480	21.14 KB	17	494 Bytes	2	0 Bytes	3.77
LazyTable(9755)	tcp4 17.244.68.249:13762	tcp4 205.180.175.173:20480	20.20 KB	16	494 Bytes	2	0 Bytes	2.02
LazyTable(9755)	tcp4 17.244.68.249:14530	tcp4 205.180.175.172:20480	18.48 KB	15	494 Bytes	2	0 Bytes	3.41
LazyTable(9755)	tcp4 17.244.68.249:16578	tcp4 205.180.175.172:20480	14.50 KB	11	247 Bytes	1	0 Bytes	0 Byte
LazyTable(9755)	tcp4 17.244.68.249:16834	tcp4 205.180.175.172:20480	11.95 KB	10	247 Bytes	1	0 Bytes	0 Byte
LazyTable(9755)	tcp4 17.244.68.249:17090	tcp4 205.180.175.172:20480	8.56 KB	7	247 Bytes	1	0 Bytes	3.22
LazyTable(9755)	tcp4 17.244.68.249:17602	tcp4 205.180.175.173:20480	8.50 KB	7	247 Bytes	1	0 Bytes	3.16
LazyTable(9755)	tcp4 17.244.68.249:14018	tcp4 205.180.175.173:20480	7.36 KB	6	247 Bytes	1	1.34 KB	3.35
LazyTable(9755)	tcp4 17.244.68.249:17346	tcp4 205.180.175.173:20480	6.20 KB	5	247 Bytes	1	0 Bytes	0 Byte

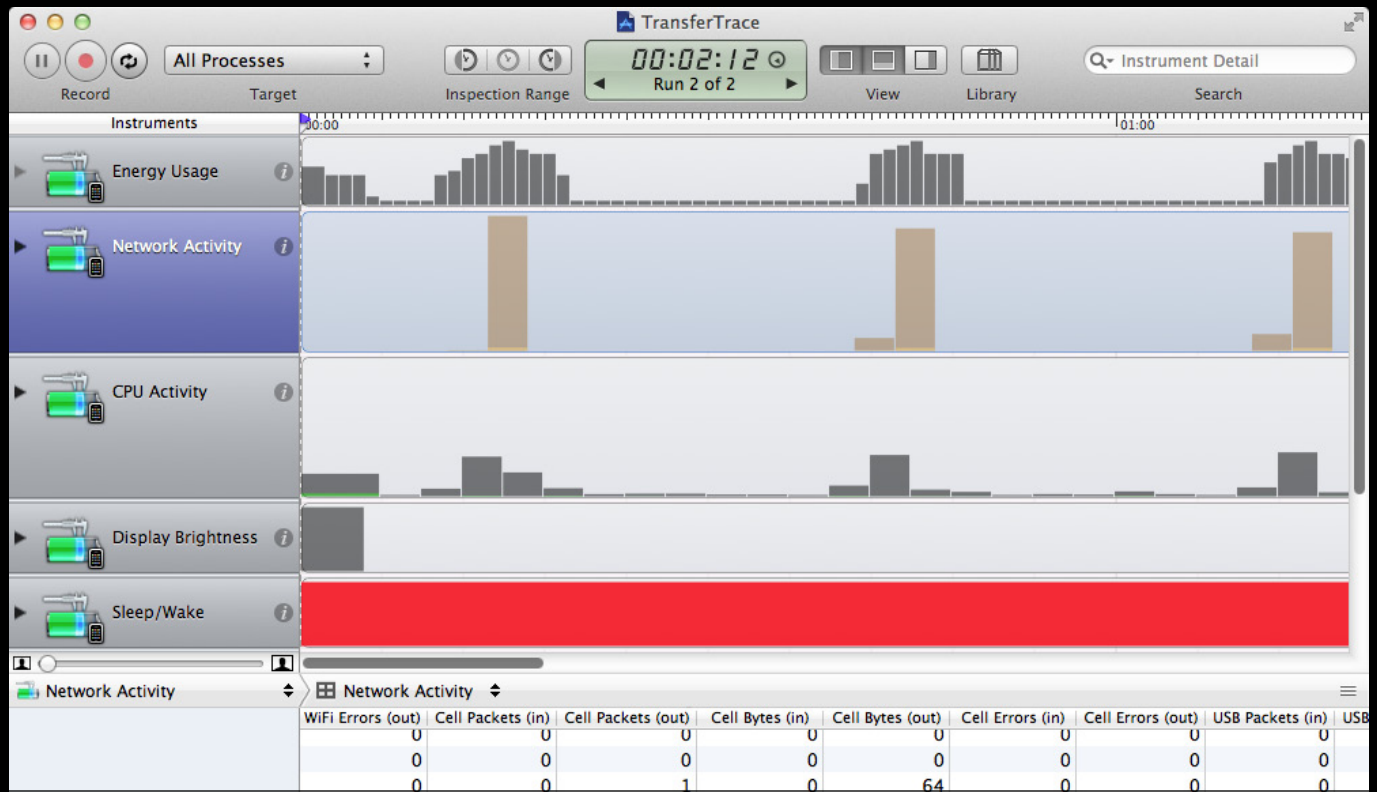
Network Activity Instrument



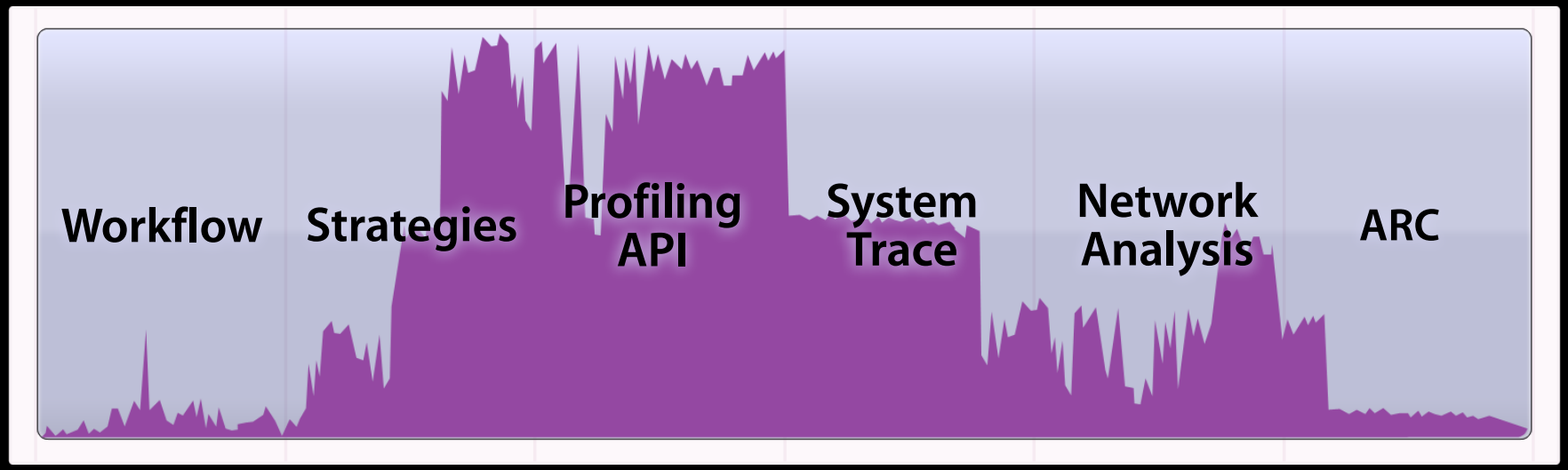
- Identify radio traffic in and out
  - Wi-Fi
  - Cellular
- Correlate power use with radio use

Network Activity

New in iOS 5

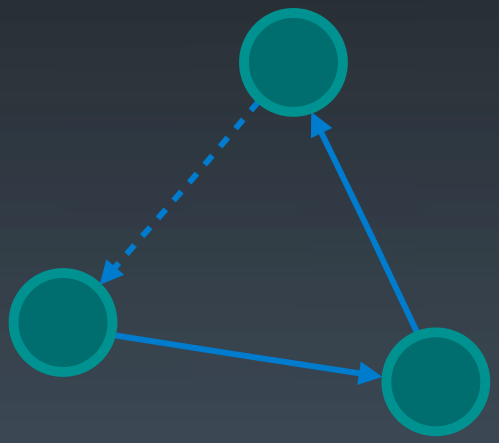


ARC Instrumentation



ARC Instrumentation

Leak cycle detection



Automatic Reference Counting (ARC)

What's automatic about it?

- No more writing `-retain`/`-release`/`-autorelease`
- Enforces previous "convention"
- Eliminates many simple bugs
- Allows you to think about object relationships

Automatic Reference Counting (ARC)

What's not automatic about it?

- It's not a garbage collector
- Runs alongside manual reference counting (MRC) code
- Cannot break retain cycles
- Cannot prevent leaks

Automatic Reference Counting (ARC)

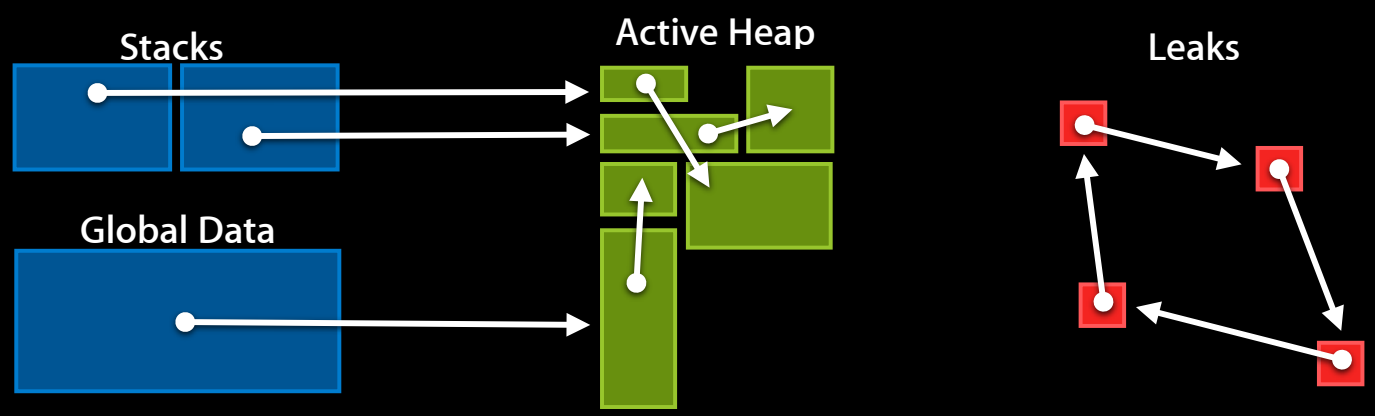
But Instruments can help!

- Leaks instrument identifies
  - "Allocated memory that can no longer be reached"
    - No more pointers to it from the active heap

Automatic Reference Counting (ARC)

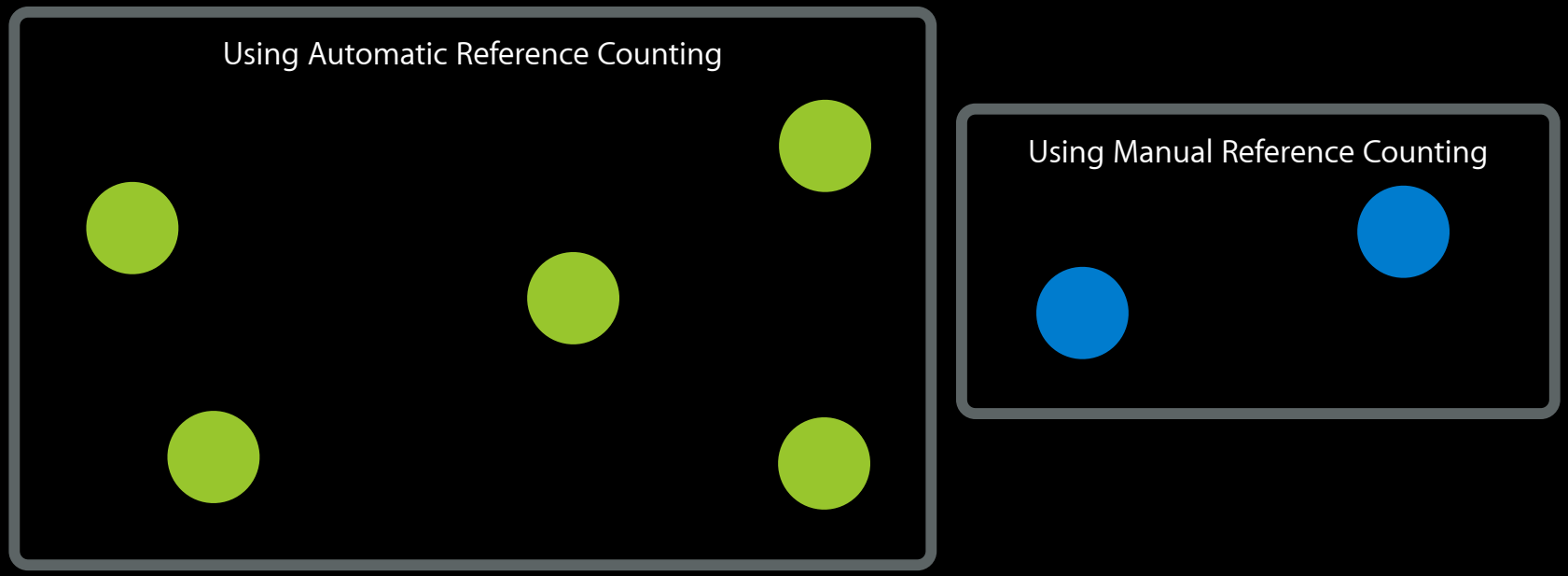
But Instruments can help!

- Leaks instrument identifies
  - "Allocated memory that can no longer be reached"
    - No more pointers to it from the active heap
  - Cycle detection



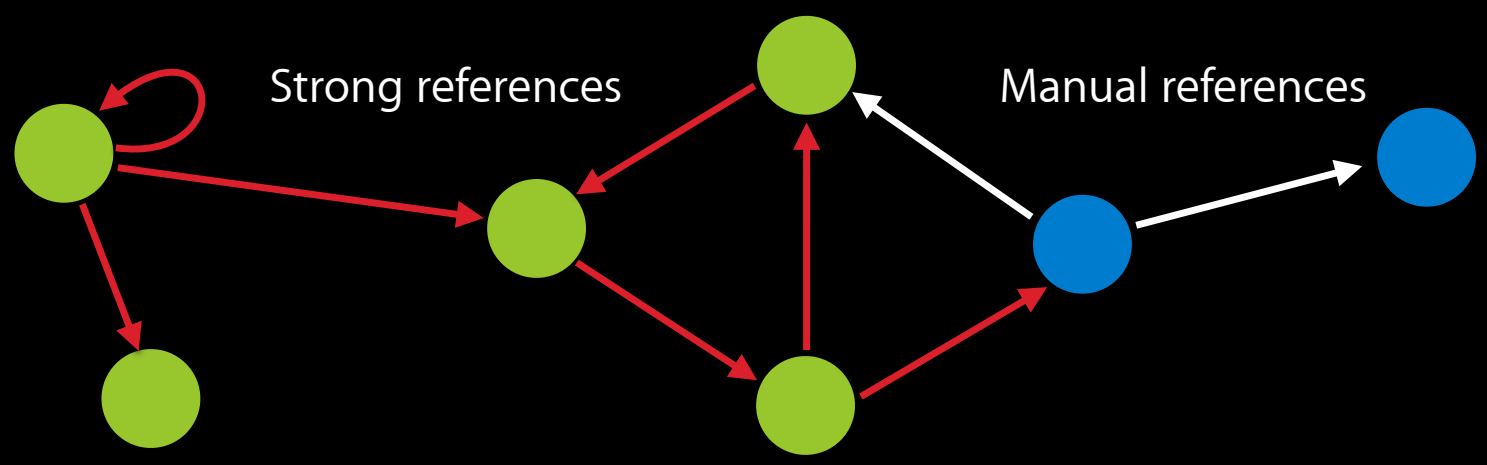
Fixing Leaks with ARC

Starting at the right spot



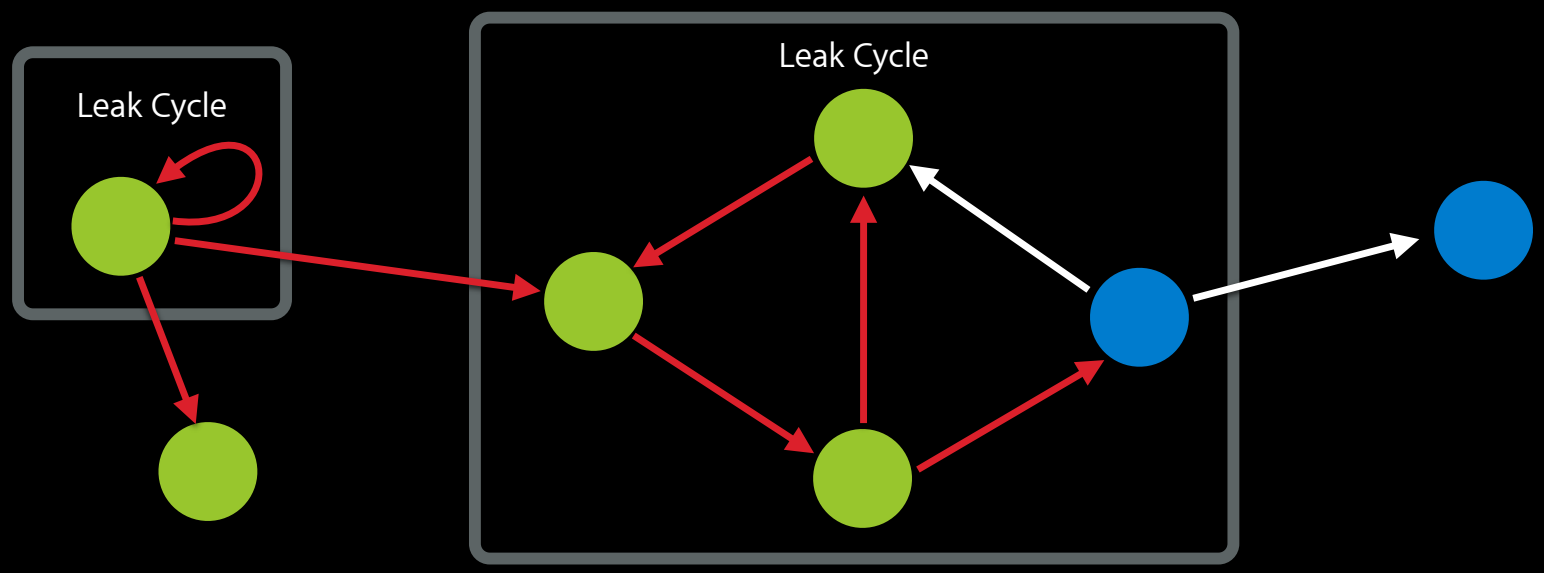
Fixing Leaks with ARC

Starting at the right spot



Fixing Leaks with ARC

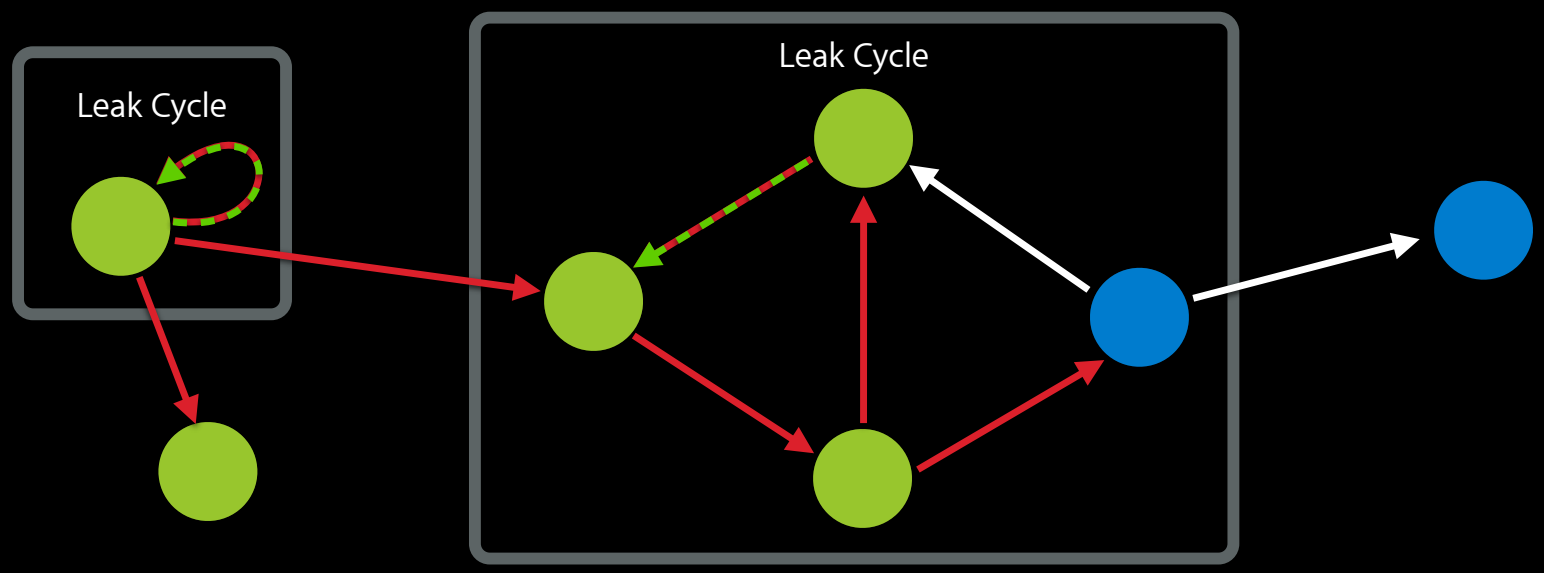
Starting at the right spot



Fixing Leaks with ARC

Starting at the right spot

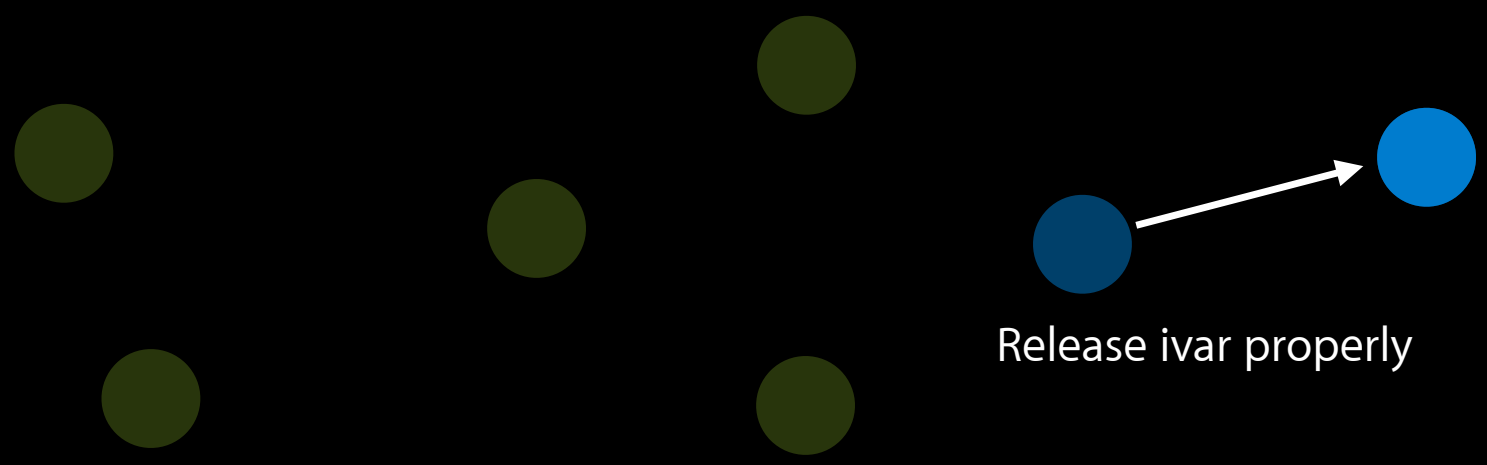
1. Fix the cycles



Fixing Leaks with ARC

Starting at the right spot

1. Fix the cycles
2. Fix remaining manual reference counting leaks



ARC Demo

Daniel Delwood
Performance Tools Engineer

In Conclusion

- Tackle concurrency issues with the CPU strategy
- Examine the efficiency of your application via System Trace
- Leverage the new DTPerformanceSession framework
- Eliminate your leaked reference cycles with ARC instrumentation

More Information

Michael Jurewitz
Developer Tools & Performance Evangelist
jurewitz@apple.com

Instruments Documentation
Instruments User Guide (Xcode documentation)
Instruments New Features User Guide

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

Related Sessions

iOS Performance and Power Optimization with Instruments	Presidio Wednesday 4:30PM
iOS Performance in Depth	Presidio Thursday 4:30PM
Objective-C Advancements In-Depth	Mission Friday 11:30AM

Labs

Mac OS X Performance Lab	Developer Tools Lab B Thursday 9:00AM
iOS App Performance Lab	Developer Tools Lab A Thursday 9:00AM
Objective-C and Automatic Reference Counting Lab	Developer Tools Lab B Thursday 2:00PM

WWDC2011