Using Time Profiler in Instruments

Profile early, profile often

Session 418

Kris Markel
Agenda

Intro to profiling
Going faster
Doing less
Improving responsiveness
A Better User Experience
Profiling

How much and what kind of work is my app doing?
Instruments
Measure, adjust, repeat
## Getting Started

<table>
<thead>
<tr>
<th>Editor</th>
<th>Product</th>
<th>Debug</th>
<th>Source Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Profile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Archive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build For</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perform Action</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scheme</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Bot...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Demo
Profiling an app
main()
method1()
method2()
main()
  method1()
  method2()
main()
  method1()
  method2()

1 main()
1  method1()
1   method2()
main()
  method1()
  method2()
main()
  method1()
  method2()

main()
  method1()
  method2()

Time

1 ms

1 main()
1 method1()
1 method2()
main()
method1()
method2()

main()
method1()
method2()

Time
1 ms

main

method1

method2

method2

3

method2

2 main()

2 method1()

2 method2()
main()
    method1()
    method2()

Time
1 ms

main

method1

method2

method2

method2

method2

method2

2 main()

2 method1()

2 method2()
main()  
  method1()  
  method2()  

main()  
  method1()  
  method2()  

main()  
  method2()  

```
2 main()
2   method1()
2   method2()
2
```
main()
  method1()
  method2()

main()
  method1()
  method2()

main()
  method1()
  method2()

main()
  method1()
  method2()

main()
  method1()
  method2()

main()
  method1()
  method2()

main()
  method1()
  method2()

main()
  method1()
  method2()

main()
  method1()
  method2()

Time

1 ms
main()
method1()
method2()
main()
method1()
method2()
main()
method1()
method2()
main()
method2()

Time

1 ms

main method1 method2

method2
3
method2
3
method2
3
method2

5  main()
4  method1()
4  method2()
1  method2()
The Call Tree

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>main()</td>
</tr>
<tr>
<td>4</td>
<td>method1()</td>
</tr>
<tr>
<td>4</td>
<td>method2()</td>
</tr>
<tr>
<td>1</td>
<td>method2()</td>
</tr>
</tbody>
</table>
The Call Tree

Doesn't measure duration
The Call Tree

Doesn’t measure duration
Aggregates samples into a useful summary

<table>
<thead>
<tr>
<th></th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>main()</td>
</tr>
<tr>
<td>4</td>
<td>method1()</td>
</tr>
<tr>
<td>4</td>
<td>method2()</td>
</tr>
<tr>
<td>1</td>
<td>method2()</td>
</tr>
</tbody>
</table>
The Call Tree

Doesn’t measure duration
Aggregates samples into a useful summary
  • Long running vs. repetitive

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>main()</td>
</tr>
<tr>
<td>4</td>
<td>method1()</td>
</tr>
<tr>
<td>4</td>
<td>method2()</td>
</tr>
<tr>
<td>1</td>
<td>method2()</td>
</tr>
</tbody>
</table>
The Call Tree

Doesn’t measure duration
Aggregates samples into a useful summary

• Long running vs. repetitive
The Call Tree

Doesn’t measure duration
Aggregates samples into a useful summary

• Long running vs. repetitive

Focuses on CPU usage
The Call Tree

Doesn’t measure duration

Aggregates samples into a useful summary

• Long running vs. repetitive

Focuses on CPU usage

• Doesn’t capture everything

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>main()</td>
</tr>
<tr>
<td>4</td>
<td>method1()</td>
</tr>
<tr>
<td>4</td>
<td>method2()</td>
</tr>
<tr>
<td>1</td>
<td>method2()</td>
</tr>
</tbody>
</table>
Demo
Going faster and doing less
Starting the Investigation

Look for the unexpected

• Trackpad friendly
• Drag to apply a filter
• Draggable filter edges
• Option-drag to zoom in control-drag to zoom out
• Data values on hover
What We Did
What We Did

Going faster
What We Did

Going faster

• Focused on an area of high CPU usage
What We Did

Going faster

• Focused on an area of high CPU usage
• Examined the call tree, looking for where the work was happening
What We Did

Going faster

• Focused on an area of high CPU usage
• Examined the call tree, looking for where the work was happening
• Walked back to our code
What We Did

Going faster

• Focused on an area of high CPU usage
• Examined the call tree, looking for where the work was happening
• Walked back to our code
• Inspected our code
What We Did

Going faster

• Focused on an area of high CPU usage
• Examined the call tree, looking for where the work was happening
• Walked back to our code
• Inspected our code
• Made it faster
What We Did

Going faster

• Focused on an area of high CPU usage
• Examined the call tree, looking for where the work was happening
• Walked back to our code
• Inspected our code
• Made it faster
• Verified the changes
What We Did

Going faster

• Focused on an area of high CPU usage
• Examined the call tree, looking for where the work was happening
• Walked back to our code
• Inspected our code
• Made it faster
• Verified the changes
• Saved the user’s time
What We Did
What We Did

Doing less
What We Did

Doing less

• Focused on low, but unexpected CPU usage
What We Did

Doing less

• Focused on low, but unexpected CPU usage
• Examined the call tree
What We Did

Doing less

• Focused on low, but unexpected CPU usage
• Examined the call tree
• Determined the frameworks involved
What We Did

Doing less

• Focused on low, but unexpected CPU usage
• Examined the call tree
• Determined the frameworks involved
• Stopped doing unnecessary work
What We Did

Doing less

- Focused on low, but unexpected CPU usage
- Examined the call tree
- Determined the frameworks involved
- Stopped doing unnecessary work
- Verified the changes
What We Did

Doing less
• Focused on low, but unexpected CPU usage
• Examined the call tree
• Determined the frameworks involved
• Stopped doing unnecessary work
• Verified the changes
• Improved battery life
Responsiveness
Responsiveness
Responsiveness

The main thread does all the UI work
Responsiveness

The main thread does all the UI work

• Run loop waiting for events
Responsiveness

The main thread does all the UI work

• Run loop waiting for events
• Sends events to your UIApplication instance
Responsiveness

The main thread does all the UI work

- Run loop waiting for events
- Sends events to your UIApplication instance
- Passes through the responder chain
Responsiveness

The main thread does all the UI work

• Run loop waiting for events
• Sends events to your UIApplication instance
• Passes through the responder chain
• Your code gets invoked
Responsiveness

The main thread does all the UI work

- Run loop waiting for events
- Sends events to your UIApplication instance
- Passes through the responder chain
- Your code gets invoked
Responsiveness

The main thread does all the UI work

- Run loop waiting for events
- Sends events to your UIApplication instance
- Passes through the responder chain
- Your code gets invoked
Responsiveness

The main thread does all the UI work
- Run loop waiting for events
- Sends events to your UIApplication instance
- Passes through the responder chain
- Your code gets invoked

![Image of stack trace]
Responsiveness

The main thread does all the UI work
- Run loop waiting for events
- Sends events to your UIApplication instance
- Passes through the responder chain
- Your code gets invoked
Responsiveness

The main thread does all the UI work
• Run loop waiting for events
• Sends events to your UIApplication instance
• Passes through the responder chain
• Your code gets invoked
Responsiveness

The main thread does all the UI work

- Run loop waiting for events
- Sends events to your UIApplication instance
- Passes through the responder chain
- Your code gets invoked
Responsiveness
Responsiveness

When busy, the main thread can’t process events
Responsiveness

When busy, the main thread can’t process events

• The queue backs up
Responsiveness

When busy, the main thread can’t process events

• The queue backs up
• Stuttering and hiccups
Responsiveness

When busy, the main thread can’t process events

- The queue backs up
- Stuttering and hiccups
- App becomes unresponsive
Responsiveness

When busy, the main thread can’t process events
• The queue backs up
• Stuttering and hiccups
• App becomes unresponsive

Keep the main thread free
Demo
Improving responsiveness
What We Did
What We Did

Responsiveness
What We Did

Responsiveness

- Examined the CPU spikes
What We Did

Responsiveness

• Examined the CPU spikes
• Focused on the main thread
What We Did

Responsiveness

• Examined the CPU spikes
• Focused on the main thread
• Identified non-UI work happening on the main thread
What We Did

Responsiveness

• Examined the CPU spikes
• Focused on the main thread
• Identified non-UI work happening on the main thread
• Distributed the work across multiple threads
What We Did

Responsiveness

• Examined the CPU spikes
• Focused on the main thread
• Identified non-UI work happening on the main thread
• Distributed the work across multiple threads
• Verified the changes
What We Did

Responsiveness

- Examined the CPU spikes
- Focused on the main thread
- Identified non-UI work happening on the main thread
- Distributed the work across multiple threads
- Verified the changes
- Achieved a better user experience
Regarding Optimization
Regarding Optimization

The fixes were simple, but added complexity
Regarding Optimization

The fixes were simple, but added complexity

By finding them early in the development process, there’s more time to verify correctness
Regarding Optimization

The fixes were simple, but added complexity

By finding them early in the development process, there’s more time to verify correctness

Profile early, profile often!
Regarding Optimization

The fixes were simple, but added complexity

By finding them early in the development process, there’s more time to verify correctness

Profile early, profile often!

Sometimes get big gains for little effort
Best Practices

Always profile release builds
Always profile on the device
Run with old devices
Use large data sets where it makes sense
  • Look for poorly scaling code (O(n²), etc.)
Summary

If you want …
If you want …

• The best experience for your users across all the devices you support
Summary

If you want …

• The best experience for your users across all the devices you support
  - A faster application
Summary

If you want …

• The best experience for your users across all the devices you support
  - A faster application
  - Better battery life
If you want …

• The best experience for your users across all the devices you support
  - A faster application
  - Better battery life
  - More responsive UI
Summary

If you want …

• The best experience for your users across all the devices you support
  - A faster application
  - Better battery life
  - More responsive UI

Then you should …
Summary

If you want …

• The best experience for your users across all the devices you support
  - A faster application
  - Better battery life
  - More responsive UI

Then you should …

• Profile early
Summary

If you want …

• The best experience for your users across all the devices you support
  - A faster application
  - Better battery life
  - More responsive UI

Then you should …

• Profile early
• Profile often
More Information

## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizing App Startup Time</td>
<td>Mission</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>System Trace in Depth</td>
<td>Nob Hill</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Thread Sanitizer and Static Analysis</td>
<td>Mission</td>
<td>Thursday 10:00AM</td>
</tr>
<tr>
<td>Optimizing I/O for Performance and Battery Life</td>
<td>Nob Hill</td>
<td>Friday 11:00AM</td>
</tr>
<tr>
<td>Concurrent Programming with GCD in Swift 3</td>
<td>Pacific Heights</td>
<td>Friday 4:00PM</td>
</tr>
<tr>
<td>Unified Logging and Activity Tracing</td>
<td>Nob Hill</td>
<td>Friday 5:00PM</td>
</tr>
<tr>
<td>Profiling in Depth</td>
<td>WWDC 2015</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td>Xcode Open Hours</td>
<td>Swift Open Hours</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>Developer Tools Lab B</td>
<td>Developer Tools Lab A</td>
</tr>
<tr>
<td></td>
<td>Friday 3:00PM</td>
<td>Friday 3:00PM</td>
</tr>
</tbody>
</table>