Concurrency in iOS

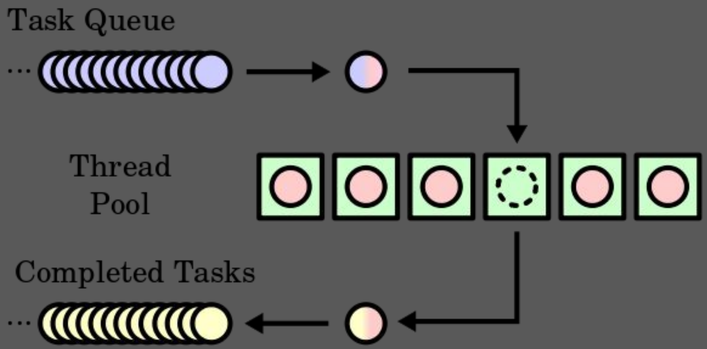
By Nathan Roberts

How do you manage threads?

You don't. Well, not directly.

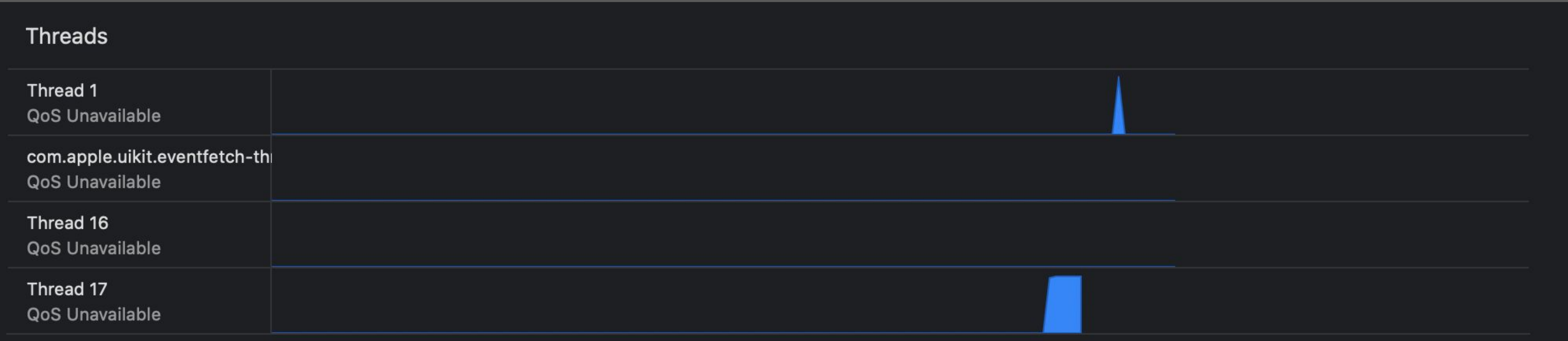
Grand Central Dispatch (GCD)

- Main idea is to move the manual management of threads away from the developer, and provide a low-level API to make it easy.
- Follows the thread pool pattern, meaning there are many threads on standby ready to be used.
- Thus, instead of messing with the threads themselves, we will instead be interacting with a queue of tasks.



Grand Central Dispatch (GCD) Continued

- Dispatch Queues (what we will be focusing on)
- Dispatch Sources (rarely used)
- Dispatch Groups (critical sections)
- Dispatch Semaphores (rarely used)



Dispatch Queues: The key to concurrency

- ➔ Using Dispatch Queues, we can execute blocks of code in a variety of ways which will allow us to achieve our goals. Some qualities are defined below...

Primary Characteristics:

- ➔ Concurrent Queue: All blocks of code start in the order added to the queue, but run at the same time.
- ➔ Serial Queue: All blocks of code added to the queue are run in the order they were added. One at a time.

Secondary Characteristics:

- ➔ Synchronous: Pauses the current thread and waits until the block is completed before resuming.
- ➔ Asynchronous: Current thread continues to run regardless of the block’s progress.

Dispatch Queues: More things to consider

- ➔ All code inside a block given to a dispatch queue will run in the order it is written, regardless of the characteristics.
- ➔ In practice, I've only ever seen dispatch queues being used out of the four tools offered by GCD. Things like dispatch group and sources provide unique functionality, but it should only be needed in very rare cases.
- ➔ Quality-of-Service(QoS): Rather than writing queues in order manually, you can specifically define the QoS which determines the initial order of the queue. I've personally never seen it used, but it does have a place. For example, you can specify a block to be a background task which isn't as critical. Or, you can say that it is something initialized by the user or is blocking interaction, and it will be the main priority.

Dispatch Queues: QoS

➔ **QoS Types (Highest Priority -> Lowest Priority):**

1. User-Interactive: has to do with the UI, instantaneous results (ex: animations, refreshing UI)
2. User-initiated: user has initiated it and requires immediate results (ex: opening a document)
3. Utility: takes time to complete and doesn't require an immediate result (ex: downloading data)
4. Background: not time crucial, not visible to the user, energy efficient (ex: backups, syncing, etc)

Last Notes

- **ALL** UI changes and updates take place on one thread, known as the main thread.
- Dispatch Queues can be run on the main thread, or can be called using global() which picks from the thread pool I mentioned earlier.
- Closures are commonly used when dealing with asynchronous, global Dispatch Queues.

Why is this important to understand?

Demo

- ➔ How to correctly use Dispatch Queues to prevent the UI from freezing
- ➔ How to prevent race conditions using default Dispatch Queues
- ➔ How QoS works, and how it can also be used to help prevent race conditions

The End. Demo Time.