Introduction

Concurrency is the notion of multiple things happening at the same time. With the proliferation of multicore CPUs and the realization that the number of cores in each processor will only increase, software developers need new ways to take advantage of them. Although operating systems like OS X and iOS are capable of running multiple programs in parallel, most of those programs run in the background and perform tasks that require little continuous processor time. It is the current foreground application that both captures the user's attention and keeps the computer busy. If an application has a lot of work to do but keeps only a fraction of the available cores occupied, those extra processing resources are wasted.

In the past, introducing concurrency to an application required the creation of one or more additional threads. Unfortunately, writing threaded code is challenging. Threads are a low-level tool that must be managed manually. Given that the optimal number of threads for an application can change dynamically based on the current system load and the underlying hardware, implementing a correct threading solution becomes extremely difficult, if not impossible to achieve. In addition, the synchronization mechanisms typically used with threads add complexity and risk to software designs without any guarantees of improved performance.

Both OS X and iOS adopt a more asynchronous approach to the execution of concurrent tasks than is traditionally found in thread-based systems and applications. Rather than creating threads directly, applications need only define specific tasks and then let the system perform them. By letting the system manage the threads, applications gain a level of scalability not possible with raw threads. Application developers also gain a simpler and more efficient programming model.

This document describes the technique and technologies you should be using to implement concurrency in your applications. The technologies described in this document are available in both OS X and iOS.

Organization of This Document

This document contains the following chapters:

- Concurrency and Application Design introduces the basics of asynchronous application design and the technologies for performing your custom tasks asynchronously.
- Operation Queues shows you how to encapsulate and perform tasks using Objective-C objects.
- Dispatch Queues shows you how to execute tasks concurrently in C-based applications.
- Dispatch Sources shows you how to handle system events asynchronously.
- Migrating Away from Threads provides tips and techniques for migrating your existing thread-based code over to use newer technologies.

This document also includes a glossary that defines relevant terms.

A Note About Terminology

Before entering into a discussion about concurrency, it is necessary to define some relevant terminology to prevent confusion. Developers who are more familiar with UNIX systems or older OS X technologies may find the terms "task", "process", and "thread" used somewhat differently in this document. This document uses these terms in the following way:

- The term *thread* is used to refer to a separate path of execution for code. The underlying implementation for threads in OS X is based on the POSIX threads API.
- The term *process* is used to refer to a running executable, which can encompass multiple threads.

- The term *task* is used to refer to the abstract concept of work that needs to be performed.

For complete definitions of these and other key terms used by this document, see Glossary.

See Also

This document focuses on the preferred technologies for implementing concurrency in your applications and does not cover the use of threads. If you need information about using threads and other thread-related technologies, see *Threading Programming Guide*.