ИССЛЕДОВАНИЕ МЕХАНИЗМОВ РАСПАРАЛЛЕЛИВАНИЯ ЗАДАЧ В IOS

Лапковская Екатерина Ивановна

студент, Брянский государственный технический университет, РФ, г. Брянск

Сизый Денис Андреевич

магистрант, Брянский государственный технический университет, РФ, г. Брянск

Дергачёв Константин Владимирович

научный руководитель, канд. техн. наук, доцент Брянского государственный технический университет, РФ, г. Брянск

Актуальность исследования. В настоящее время мобильные приложения имеют огромную популярность в решении как прикладных задач, так и задач для бизнеса. Эта ветка информационных технологий активно развивается, появляется все больше различных приложений, выполняющих огромное количество процессов.

Пользователь мобильного приложения ожидает, что устанавливаемые им приложения имеют быструю реакцию на действия пользователя, а ее интерфейс не "зависает", что все выплоняемые приложением действия происходят быстро. А качество приложения является одним из критериев его конкурентоспособности.

Степень разработанности темы. Изучению вопросов о методах распараллеливания задач в iOS посвящено множество статей как отечественного авторства, так и зарубежного. Однако, множество статей далеко не полностью описывают данную тему, либо не дают никаких рекомендаций по использованию механзмов.

Цель исследования – анализ и применение механизмов и средств распараллеливания задач на языке Objective-C для платформы iOS.

Задачи исследования – проанализировать методы и средства для распараллеливания задач в iOS и разработать рекомендации по их использованию.

Объектом исследования являются методы и средства распараллеливания задач в iOS.

Apple предоставляет несколько инструментов для распараллеливания задач:

1. механизм Grand Central Dispatch;

2. технологии NSOperation и NSOperationQueue;

3. метод performSelectorInBackground.

Рассмотрим их более подробно.

Grand Central Dispatch (далее - GCD) – официальное название для libdispatch, библиотеки, которая предоставляет поддержку параллельного выполнения кода на многоядерном оборудовании на iOS и OS X. Она имеет следующие приемущества.

1. Может улучшить время отклика приложения, помогая отложить дорогостоящие в плане вычислений задачи и выполнить их в фоне.

2. Обеспечивает более легкую модель параллеллизма по сравнению с блокировками и потоками и помогает избежать ошибок параллелизма.

3. Может оптимизировать код с помощью примитивов для таких паттернов проектирования, как, например, одиночка (Singletone).

GCD предоставляет очереди отправки (dispatch queues) для обработки блоков кода; эти очереди выполняют предоставленные им задачи по принципу FIFO. Все очереди отправки являются потокобезопасными, к ним можно обращаться из нескольких потоков.

Также доступны объекты источники отправки (dispatch sources), с помощью которых можно регистрировать функции или блоки для асинхронного выполнения при возникновении какого-то события.

Еще GCD объявляет группы отправки (dispatch groups), позволяющие собирать задачи в отдельные группы. В очередь можно добавить задачи-члены группы, а сам объект группы потом можно использовать для ожидания всех задач.

Дополнительно GCD предоставляет объекты семафоры отправки (dispatch semaphores), с помощью которых можно запретить числу задач, превышающему заданное, выполняться одновременно.

NSOperation представляет собой отдельную единицу задачи. Это абстрактный класс, который предлагает полезную и потокобезопасную структуру для моделирования состояний, приоритета, зависимостей и управления. Примеры задач, которые можно решать с помощью этого класса – сетевые запросы, изменение размеров изображения и другие. Но простое оборачивание вычислений в объект не принесет результата без контроля. Для этого и нужен NSOperationQueue.

NSOperationQueue регулирует параллельное выполнение операций. Этот класс выступает в качестве очереди с приоритетом, так что операции выполняются примерно по принциипу FIFO. Также NSOperationQueue может ограничивать количество параллельных операций, выполняемых в данный момент через свойство maxConcurrentOperationCount.

NSOperation имеет следующие преимущества:

1. поддержка зависимостей с помощью метода addDependency:(NSOperation *)operation. Используется, если нужно запустить операцию, которая зависит от выполнения другой.

2. Возможность изменения приоритета операции с помощью изменения свойства queuePriority, которое может принимать одно из следующих значений: VeryLow, Low, Normal, High, VeryHigh.

3. Можно отменить операцию или целую очередь операций. Это делается посредством вызова метода cancel. Далее возможны три ситуации.

1) Операция уже завершена. В этом случае метод завершения никак не повлияет.

2) Операция в данный момент выполняется. Система не выполнит немедленной остановки операции, но будет выставлен флаг, что операция была отменена.

3) Операция ждет выполнения в очереди. В этом случае опреация не будет выполнена.

4. В этом классе есть три полезных булевских свойства, такие как finished (завершена), cancelled (отменена) и ready (готова к выполнению).

5. Каждому экземпляру NSOperation можно добавить блок, который будет вызван сразу после

завершения операции, то есть, после присвоения свойству finished значения истины.

Метод performSelectorInBackground класса NSObject позволяет неявно создать и запустить новый поток. Поток начинает свою работу немедленно как отдельный фоновый поток, возвращая управление вызвавшему потоку. Метод performSelectorInBackground является одним из простейших способов обработки данных вне родительского потока.

Метод performSelectorInBackground самый простой – его нужно использовать в ситуации, когда в отдельный поток нужно вынести один метод, выполняющий какую-то единицу работы.

Сложнее выбрать между NSOperation и Grand Central Dispatch. В общих случаях Apple рекомендует разработчикам использовать как можно более высокий уровень абстракции. Если следовать данной рекомендации, выбор падает на NSOperation.

Почему стоит этому следовать? С каждым выпуском Apple оптимизируют и вносят изменения в библиотеки и фреймворки, улучшая тем самым и работу операционной системы. Обычно это включает в себя и изменения в низкоуровневых библиотеках. Какие-то методы могут быть изменены или вовсе исключены. Но стоит ли отказываться от GCD только потому, что это низкоуровневая библиотека? Конечно, нет. Многие разработчики активно используют комбинацию технологий.

Когда стоит использовать NSOperation? NSOperation отлично подходит для инкапсуляции четко определенных блоков функциональности. Например, можно использовать NSOperation для инкапсуляции авторизации в приложении. Если нужно выполнить несколько задач в определенном порядке, значит нужно управление зависимостями. Такой функцией обладает NSOperation. Но при слишком большом количестве операций (десятки и более) могут возникнуть проблемы с производительностью.

Когда использовать GCD? GCD идеально подходит, если нужно просто выполнить блок кода в последовательной или параллельной очереди. Если нет желания создавать NSOperation для тривиальной задачи, GCD - отличная альтернатива. Еще одно преимущество GCD перед NSOperation – код обработки не расползается по всему файлу, а держится в одном месте. Все очереди отправки, семафоры и так далее представляют собой необходимый набор примитивов параллелизма, поверх которых и строятся все системные структуры. Для одноразового вычисления или просто ускорения существующего метода часто будет удобнее использовать GCD.

Список литературы:

1. Campbell M. Objective-C Recipes. A Problem-Solution Approach, 2012. – 460 с.

2. Conway J., Hillegass A. iOS Programming: The Big Nerd Ranch Guide (3rd Edition). 2012 – 590 с.

3. Deitel P., Deitel H.M. iPhone For Programmers: An App-Driven Approach. 2009. – 456 с.

4. Knaster S., Dalrymple M., Malik W. Learn Objective-C on the Mac For OS X and iOS. 2012 – 384 с.

5. Pilone D., Pilone T. A Learner's Guide to Creating Objective-C Applications for the iPhone. 2009 – 560 с.

6. Аллан А. Программирование для мобильных устройств на iOS. 2013 – 406 с.

7. Кочан С. Программирование на Objective-C 2.0, 2014. – 608 с.

8. Нахавандипур В. iOS. Приемы программирования. 2015 – 832 с.