ОБЗОР ВАРИАНТОВ ДИСПЕТЧЕРИЗАЦИИ МЕТОДОВ И СООБЩЕНИЙ НА ПЛАТФОРМАХ iOS и macOS

Булыга И.М.

Булыга Игорь Михайлович - разработчик программного обеспечения для платформы iOS, Компания Booking.com, г. Амстердам, Нидерланды

Аннотация: *в статье рассматриваются и сравниваются различные варианты диспетчеризации методов и сообщений, которые используются на платформах iOS и macOS.*
Ключевые слова: *Swift, Objective-C, ios, macos, method dispatch.*

В данной статье будут рассмотрены варианты диспетчеризации методов под платформы iOS и macOS. Для начала мы рассмотрим, что такое методы в парадигме объектно-ориентированного программирования (ООП), затем углубимся в понятие диспетчеризации, а потом будут освещены три варианта диспетчеризации – прямой (по-английски – direct/static dispatch), и два динамических – табличный (по-английски – table dispatch) и отсылка сообщения (по-английски – message dispatch).

Метод

Метод — это набор инструкций, которые должна выполнить программа чтобы произвести какую-то полезную работу вместе с объектом в парадигме ООП. Это может быть изменение состояния объекта, взаимодействие с другими объектами или какие-либо полезные вычисления.

Диспетчеризация

Для того чтобы центральный процессор управления (ЦПУ) выполнил инструкции, которые содержатся в нашем методе он должен знать их адрес, именно процесс поиска адрес и называется диспетчеризацией. Этот процесс ответственен за то, чтобы хранить адрес, по которому хранятся правильные инструкции для нашего метода. В языке программирования Swift, при использовании на платформах iOS и macOS, доступны 3 варианта диспетчеризации – прямой, табличный и отсылка сообщений.

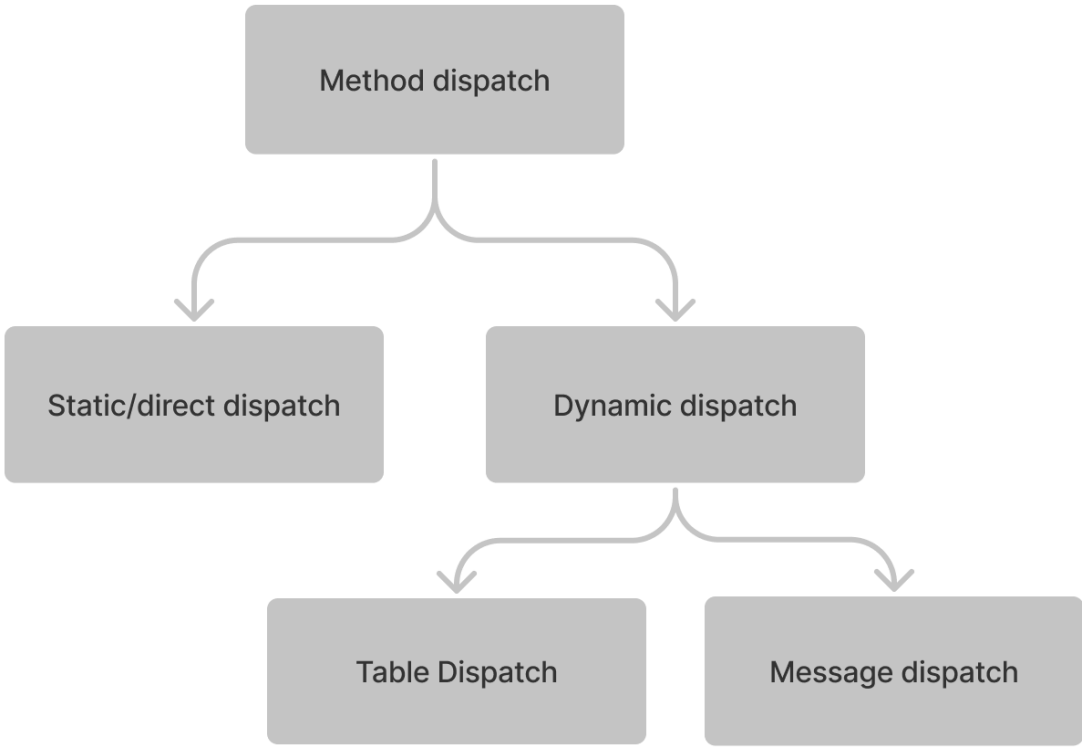

Рис. 1. Методы диспетчеризации

По сравнению с другими языками программирования, например, с Objective-C, который в основном использует отсылку сообщений, или с C, который использует прямой метод, в Swift используются все 3 варианта примерно равномерно.

Прямой вариант диспетчеризации

Прямой метод диспетчеризации является самым быстрым, так как адрес метода, по которому будут располагаться необходимые инструкции, известен на этапе компиляции программы, и во время

выполнения программа переходит к необходимым инструкциям сразу, без необходимости поиска. Так же благодаря тому, что компилятор знает все о методе, он может применить некоторые оптимизации, что ускорит выполнение этого метода. Однако, мы теряем в динамизме и такой вариант диспетчеризации может быть использован в ограниченном количестве случаев:

- В так называемых value types, к которым относятся структуры (по-английски – struct) и перечисления (по-английски – enum)
- Метод объявлен в расширении (по-английски – extension) класса, который не наследован от NSObject.
- Метод объявлен в классе, и помечен дополнительными атрибутами final, private или @inline.

Как видно, такой метод диспетчеризации позволяет достичь высокой скорости выполнения, однако накладывает ограничения на наследовании.

Табличный вариант диспетчеризации

Данный вариант наиболее распространен в компилируемых языках программирования, таких как Swift, Java, Perl, C++ и т.д. Он основан на том, что класс хранит таблицу с указателями на реализацию методов, чаще всего такое хранилище называется виртуальная таблица (по-английски – virtual table). Эта таблица создается на этапе компиляции и во время выполнения программа обращается к ней и ищет указатель на память, где хранится реализация необходимого метода. Данный вариант используется по умолчанию в классах, так как позволяет поддерживать наследование. В этом случае каждый класс будет иметь свою собственную таблицу с указателями на реализацию методов.

Рассмотрим пример, допустим у нас есть 2 класса, класс A и класс B, который является наследником класса A. У них есть свои методы, причем класс B переопределил метод bar класса A.

```
class A {
  func foo() {}
  func bar() {}
}
class B: A {
  override func bar() {}
  func zap() {}
}
```

В данном случае на этапе компиляции будет создано две виртуальные таблицы с адресами методов, см. Рис. 2.

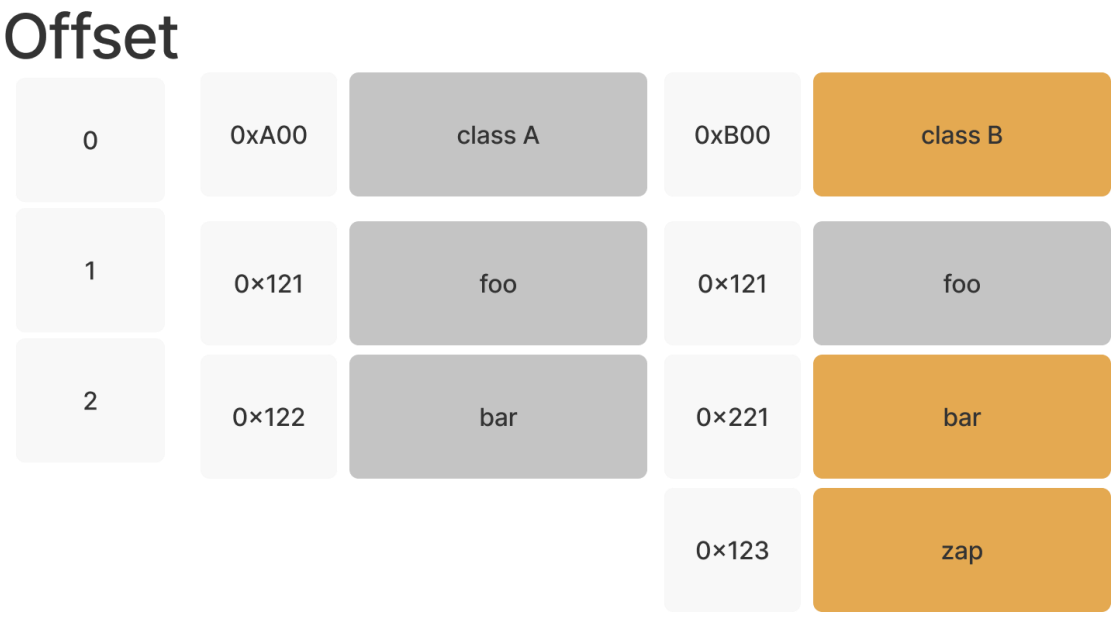

Рис. 2. Виртуальные таблицы классов A и B

Теперь, если мы создадим экземпляр класса B и вызовем метод bar, то запустится следующий алгоритм:

1. Будет прочитана виртуальная таблица класса B, которая располагается по адресу 0xB00.
2. Прочитать адрес по индексу расположения метода bar. Так как метод bar располагается под индексом 1, то будет прочитан адрес 0xB00 + 1 – 0x221.
3. Исполнить инструкции, располагающиеся по адресу 0x221.

Как видно алгоритм довольно примитивен, и его скорость работы предсказуема. Такой вариант диспетчеризации медленнее, чем прямая, так как необходимо искать индекс, по которому располагается адрес искомого метода, и компилятор не может произвести оптимизации, которые возможны при прямом варианте диспетчеризации. Однако, у такого метода есть и плюсы. Он позволяет нам воспользоваться преимуществами наследования, так как методы наследников будут добавляться в конец таблицы и их можно будет найти.

Отсылка сообщений

Отсылка сообщений, является наиболее динамичным вариантом, который позволяет менять детали реализации во время выполнения программы, но и самым медленным. Этот вариант является краеугольным камнем при написании программ под платформы iOS и macOS, потому что позволяет реализовать такие парадигмы как Key Value Observing (KVO), Key Value Coding (KVC), UIAppearance, CoreData и многих других компонентов. Такой вариант позволяет менять детали реализации по средствам подмены (по-английски – swizzling).

При таком варианте диспетчеризации, во время вызова метода программа будет искать реализацию, проходя по всему дереву наследников класса, пока не найдет необходимую реализацию метода. Это очень медленная операция, но есть механизмы кэширования, которые позволяет ускорить этот процесс.

Во время первого вызова метода, программа будет искать реализацию метода, для начала она посмотрит в кэш, если там нет необходимого метода, то будет произведен поиск по всей иерархии, когда метод найдется, то он будет добавлен в кэш и в следующий раз программа сразу возьмет реализацию их кэша.

Заключение

В статье я разобрал основные варианты диспетчеризации методов, которые используются при исполнении программ под платформу iOS и macOS. Также сравнил скорость их работы и выделил плюсы и минусы каждой из них.

Список литературы

1. Increasing Performance by Reducing Dynamic Dispatch. [Электронный ресурс], 2015. Режим доступа: https://developer.apple.com/swift/blog/?id=27/ (дата обращения: 22.01.2022).
2. *Эш Майк.* Method Replacement for fun and profit. [Электронный ресурс], 2010. Режим доступа: https://www.mikeash.com/pyblog/friday-qa-2010-01-29-method-replacement-for-fun-and-profit.html/ (дата обращения: 22.01.2022).
3. *Баллард Кевин*. Universal dynamic dispatch table. [Электронный ресурс], 2015. Режим доступа: https://lists.swift.org/pipermail/swift-evolution/Week-of-Mon-20151207/001922.html/ (дата обращения: 22.01.2022).