УДК 004.051 **Информационные технологии**

Григорьев Е.А., *студент, факультет "Информатика и системы управления"*
Московский государственный технический университет
имени. Н.Э. Баумана, Россия, г. Москва
Климов Н.С., *студент, факультет "Информатика и системы управления"*
Московский государственный технический университет
имени. Н.Э. Баумана, Россия, г. Москва

ИСПОЛЬЗОВАНИЕ AHEAD-OF-TIME КОМПИЛЯЦИИ В ПЛАТФОРМЕ .NET, КАК АЛЬТЕРНАТИВА JUST-IN-TIME КОМПИЛЯЦИИ

Аннотация: В статье рассматривается Ahead-Of-Time компиляции для платформы .NET в качестве замены стандартной для платформы .NET Just-In-Time компиляции. Описываются принципы работы платформы .NET. Описываются оба вида компиляции и их преимущества и недостатки. Описываются существующие на сегодняшний день способы реализации Ahead-Of-Time компиляции для платформы .NET.

Ключевые слова: AOT компиляция,JIT компиляция, .NET, CLR, CIL, Mono AOT, CoreRT, NGEN

Abstract: This article discusses the ahead-Of-Time compilation for the .NET platform as a replacement for the standard .NET just-In-Time compilation. BOTH types of compilation and their advantages and disadvantages are Described. The article describes the existing methods of implementing Ahead-Of-Time compilation for the .NET platform.

Keywords: AOT compilation, JIT compilation,.NET, CLR, CIL, Mono AOT, CoreRT, NGEN.

Введение

.NET – это современная мощная платформа для разработки приложений на .NET совместных языках, таких как C#, F#, C++/CLI и тд. По умолчанию в .NET используется динамическая компиляция во время исполнения (JIT компиляция) программы, у которой есть свои недостатки, но также существует альтернатива, а именно AOT компиляция, о существующих вариантах для .NET и пойдет речь в данной статье.

Экосистема .NET

Прежде чем рассмотреть AOT-компиляцию в .NET стоит понять, как работает из чего состоит экосистема .NET.

Под экосистемой понимается все программное обеспечение, входящее в среду выполнения, то есть все ресурсы предназначенные для запуска .NET приложений.

В основные элементы .NET экосистемы можно выделить следующее:

1. CIL.
2. Сборки.
3. .NET компилятор.
4. CLR.
5. JIT компилятор.
6. Библиотека базовых классов.

CIL (common intermediate language)

CIL или промежуточный язык можно назвать высокоуровневым ассемблером для среды выполнения, в него компилируются все .NET совместные языки программирования на этапе создания сборок. Именно с CIL язык при помощи JIT компилятора преобразуется в машинные команды. CIL нужен для того, чтобы среде выполнения не приходилось уметь работать с несколькими языками сразу, что позволяет поддерживать сразу несколько языков.

Из аналогов можно привести байт-код для языка JAVA. CIL стандартизован и его синтаксис и мнемоника описаны в стандарте «ECMA-335».

Сборки

Сборка представляет собой dll/exe файл, вызываемый приложением или используемый другими сборками, который содержит в себе CIL инструкции для CLR и метаданные типов.

.NET компилятор

Задача .NET компилятора состоит в преобразование кода на .NET совместном языке в сборки, которые воспринимаются CLR.

CLR (common language runtime)

Основным элементом .NET экосистемы является среда исполнения, которая отвечает за выполнения и управление CIL кодом на устройстве.

Для исполнения CIL кода CLR во время выполнения программы при помощи JIT компилятора компилирует CIL код в машинный код.

В область ответственности CLR входит размещение объектов программы в памяти и управлением ссылками на них.

CLR использует сборщик мусора GC (garabge collector) для очистки памяти от объектов, которые больше не используются.

GC помогает избежать утечки памяти в программах, но вызывает простой программы на то время, пока выполняется очистка памяти, что может негативно сказаться на производительности программы.

На момент написания статьи можно выделить следующие наиболее популярные CLR:

1. .NET Framework CLR: первая среда выполнения, доступна только для операционных систем Windows.
2. CoreCLR: среда выполнения, которая является частью кроссплатформенного проекта .NET Core.
3. Mono Runtime: среда выполнения, которая является частью кроссплатформенного проекта Mono.

JIT компилятор

JIT (Just-In-Time) компилятор отвечает за преобразование CIL команд в последовательность машинных команд. Во время JIT компиляции также может

применяться адаптивная оптимизация, благодаря чему удается оптимизировать код под определенную платформу.

Библиотека базовых классов

Библиотека базовых классов, как видно из названия поставляет классы, которые являются основополагающими в .NET. Представляет собой набор сборок.

Связь между компонентами экосистемы .NET

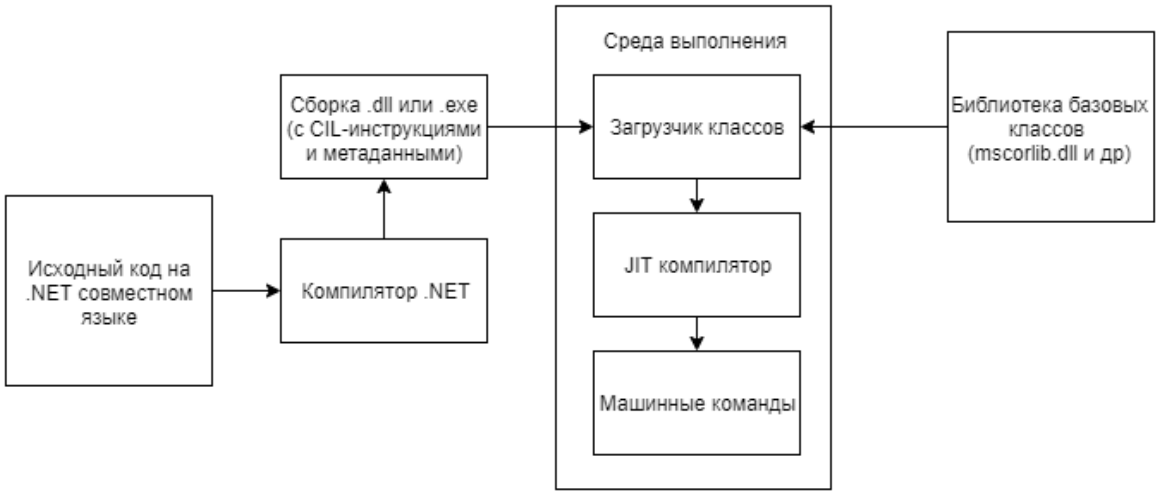

Рисунок 1. Схема взаимосвязи элементов экосистемы .NET

На рисунке 1 изображена взаимосвязь между элементами экосистемы .NET.

Как видно из рисунка вначале у нас имеется исходный код на совместном с .NET языке (C#, F#, C++/CLI, VB.NET), который при помощи компилятора .NET собирается в сборку, содержащую в себе инструкции на CIL (Common Intermediate Language) и метаданными.

После того, как программа собрана в сборки появляется возможность ее запустить.

При запуске .NET программы происходит загрузка ее сборки в память, после чего среда исполнения (CLR) по мере необходимости при помощи JIT компилятора получает из CIL специфичный для платформы код [1].

Помимо пользовательских сборок CLR подгружает сборки базовых классов. Очень важно отметить, что CLR компилирует CIL код не сразу весь, а только код, который нужен на данный момент, при необходимости с целью оптимизации код может перекомпилироваться во время исполнения программы.

Разница между AOT и JIT

JIT компиляция - это компиляция промежуточного кода в машинный код или в другой формат во время исполнения программы. Такой подход дает прирост производительности по сравнению интерпретацией промежуточного кода в машинный за счет того, что не каждая команда переводится только один раз, а также за счет применения адаптивной оптимизации кода.

Основная цель JIT компиляции - это достичь производительности статической компиляции и сохранить при этом все преимущества динамической компиляции

Достоинства JIT компиляции:

- Байт-код более переносим по сравнению с машинным кодом.
- Производительность выше чем у интерпретации.
- Оптимизация кода за счет статистики.

Недостатки JIT компиляции:

- Производительность ниже, чем у статической компиляции.
- Задержки при компиляции ранее незадействованного участка кода.
- Выделение памяти на компиляцию.
- Задержка при старте программы.

AOT компиляция - это компиляция промежуточного кода в машинный код до выполнения программы.

Основное отличие от JIT компиляции в том, что нативный код генерируется не во время выполнения программы, а заранее. В самых простых случаях, при помощи JIT компилятора промежуточный код перекомпилируется совместно используемая библиотека so/dll (в зависимости от платформы), которая подгружается средой выполнения во время запуска программы, в более

сложных случаях из среды выполнения и промежуточного кода компилируется один исполняемый файл.

Достоинства AOT компиляции:

- Производительность программ сравнима со статической компиляцией.
- Не требует расхода ресурсов на компиляцию во время исполнения.
- Быстрый по сравнению с JIT скомпилированными программами запуск.

Недостатки AOT компиляции:

- Как правило требуется собрать программы для всех целевых платформ заранее.

Имеющиеся реализации

Из имеющихся на данный момент реализаций AOT компиляции для .NET приложений достойны упоминания:

1. Mono AOT.
2. CoreRT.
3. NGEN.

Mono AOT

AOT компиляция в mono является особенностью кодо-генератора среды выполнения Mono.

AOT компиляция в Mono работает в два этапа. Первый этап состоит из прекомпилирования сборок при помощи JIT компилятора, этот процесс выполняется вручную через инструменты разработчика для каждой сборки [2].

После прекомпиляции сборки на выходе получается совместно используемая библиотека, которая содержит в себе бинарный код для текущей платформы.

Во время следующего этапа, среда выполнения Mono автоматически загружает прекомпилированный код.

Из плюсов данной реализации можно выделить следующие:

- Данная реализация AOT позволяет запускать .NET приложения на платформах, где недоступна динамическая генерация кода (таких, как ios).
- Уменьшает время запуска.
- Не тратит память на генерацию кода.
- Лишена остальных проблем связанных с JIT компиляцией.

Из минусов можно выделить следующие:

- Все еще требуется наличие среды Mono.
- Отсутствие оптимизации кода за счет статистики.

CoreRT

В отличие от Mono AOT, где сборки заранее прекомпилируются и загружается во внешнюю среду исполнения, CoreRT является отдельной средой выполнения, которая интегрируется в программу, за счет чего на выходе получается один исполняемый файл, не требующий для своего запуска наличия установленной среды исполнения.

Также за счет того, CoreRT по большей части написан на C#, неиспользуемый код среды будет отвязан на этапе сборки, что уменьшает размер программ.

Процесс сборки приложения выглядит следующим образом, вначале происходит анализ зависимостей между сборкой программы, общей библиотекой классов и средой выполнения CoreRT. После этапа анализа зависимостей возможно несколько развитий событий:

1. При помощи JIT компилятора RyuJIT компилируется исполняемый файл, содержащий, как код программы, так и код среды CoreRT.
2. Весь код программы и CoreRT конвертируется в C++ код.
3. На основе кода программы и CoreRT генерируются WebAssembly инструкции.

Из плюсов данной реализации можно выделить:

- Более быстрый старт программы по сравнению с Mono AOT.
- Меньшее потребление памяти по сравнению с Mono AOT.
- Не требует установленной среды выполнения.

Из минусов можно выделить:

- Нет возможности делить заранее скомпилированный код другим.

NGEN

NGEN представляет собой инструмент разработчика, позволяющий создавать из сборок совместно используемые библиотеки, которые устанавливаются в локальный кэш сборок на компьютере. Благодаря такому подходу, также, как и в случае с Mono AOT повышается эффективность приложений благодаря тому, что машинный код генерируется только один раз.

Из плюсов можно выделить следующие:

- увеличение скорости запуска программ.
- улучшает использование памяти, так, как JIT компилятор не используется.
- Полученные библиотеки могут использоваться несколькими процессами.

Из недостатков выделяются:

- Отсутствие версионности библиотек.
- Увеличенное использование дискового пространства.

Вывод

AOT компиляция помогает увеличить скорость запуска .NET приложений. На данный момент для .NET существуют, технологии, позволяющие использовать AOT компиляцию для .NET приложений.

Библиографический список:

1. Хранилище документации Microsoft [Электронный ресурс] https://docs.microsoft.com/ru (дата обращения 04.06.2019).

2. Официальный сайт разработчиков Mono [Электронный ресурс] https://www.mono-project.com (дата обращения 17.07.2019).