УДК 004.4'23

МЕТОДЫ РАСПРЕДЕЛЕННОГО КЭШИРОВАНИЯ РЕЗУЛЬТАТОВ КОМПИЛЯЦИИ

Акатьев Н.Л. (Университет ИТМО)
Научный руководитель – к.т.н., доцент Косяков М.С. (Университет ИТМО)

В работе над крупными программными продуктами, написанными на компилируемых языках программирования, процесс сборки кода является одним из самых затратных по времени и используемой памяти. Каждый разработчик в отдельно взятый момент работает над малой частью продукта – несмотря на это, его рабочее пространство должно содержать его большую часть. Для группы разработчиков, одновременно работающих над разными частями продукта, результаты компиляции очень часто дублируются – важно переиспользовать результаты компиляции путем кэширования и распределенного хранения их в сети. В работе реализовано комплексное решение по распределенному кэшированию компиляции проекта на языках C и C++, показавшее уменьшение общего времени сборки проекта в локальной сети разработчиков.

Введение. Компиляция – это процесс преобразования исходного кода в набор машинных инструкций и используемых ими данных. Для высокоуровневых языков программирования, таких как C++, компиляция также включает в себя задачи кодогенерации, трансляции в промежуточное представление, оптимизации времени выполнения и/или размера машинного кода. Все эти задачи делают компиляцию крайне затратной по вычислительным ресурсам, из-за чего она часто становится узким местом в процессах сборки, разработки, тестирования и доставки программного продукта клиентам.

В связи с этим встает задача уменьшения времени сборки проекта за счет ускорения процесса компиляции. Самыми популярными решениями ускорения являются инкрементальная сборка, параллельная компиляция, распределенная компиляция. Данные методы могут значительно ускорить отдельно взятую сборку продукта. Однако, в промышленной разработке, когда над одним исходным кодом работает множество разработчиков, в одной локальной сети может происходить до нескольких сотен сборок в день, при этом каждая версия собранного кода отличается изменениями в одном или нескольких программных модулях, при этом большая часть общей бизнес-логики остается неизменной.

Чтобы решить вопрос переиспользования объектов, применяется кэширование результатов компиляции. В этом случае в начало процесса компиляции добавляется задача поиска объекта в кэше путем вычисления идентификатора объекта и сопоставления его с существующими. Идентификатор получают либо из метаданных, либо из содержимого самого исходного кода.

Для эффективного хранения кэша объектов в локальной сети могут применяться распределенные хранилища, нацеленные на быстрый доступ и быстрое добавление новых объектов.

Основная часть. Целью работы является разработка и реализация механизма кэширования компиляции для языков C/C++ и распределенного хранения объектов, позволяющего эффективно переиспользовать результаты компиляции в многопользовательской локальной сети.

Для C/C++ оптимальным алгоритмом вычисления идентификатора кэша является формирование его из метаинформации: название и хэш-сумма компилятора, переданные флаги компиляции, хэш-сумма исходного кода и заголовочных файлов. Для переиспользования объектов из кэша, сформированного этим алгоритмом, система сборки должна гарантировать независимость метаданных компиляции от окружения, в котором запущена сборка. Такая независимость достигается с помощью контейнеризации.

В качестве распределенного хранилища лучше всего подходят базы данных типа «ключ-значение» – архитектура таких баз позволяет быстро расширить их в распределенных

системах, а сами пары ключ-значение занимают значительно меньше места, чем аналогичные структуры в реляционных БД. Клиентом для такой БД здесь является сам компилятор. Очистка БД происходит по принципу Least-Recently Used при достижении установленного размера хранилища. Размер хранилища выбирается исходя из размера кодовой базы и частоты ее обновления.

Реализованный механизм представляет собой связную систему, настроенную внутри локальной сети – в основу реализации легли резидентная БД Redis и обертка для компилятора CCache в качестве клиента БД. Благодаря тому, что БД находится полностью в оперативной памяти, а связь с БД осуществляется в рамках локальной сети, дополнительные затраты времени по распределенному хранению объектов минимальны.

С реализованным механизмом дополнительные расходы на распределенное кэширование составляют 10–20% от общего времени компиляции объекта. Но при этом полученный объект может быть переиспользован в других сборках – это означает, что в случае попадания в кэш процесс компиляции занимает от 5 до 10 раз меньше времени, чем если бы кэширование не использовалось.

Выводы. В результате работы был разработан и реализован комплексный механизм распределенного кэширования результатов компиляции. Реализованный механизм был протестирован на сборках крупного продукта в локальной сети и показал значительный прирост общей производительности.

Акатьев Н.Л. (автор) Подпись

Косяков М.С. (научный руководитель) Подпись