СИСТЕМА РАСПРЕДЕЛЕННОЙ КОМПИЛЯЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ

Л.Ф. Зимянин, Ю.М. Мельничек
Беларусь, г. Минск

Введение. Как известно, проекты, написанные на C++, компилируются очень долго. Например, компиляция проекта KDE на компьютере с процессором *Athlon 2.5 ghz* может занимать более 7 часов. Можно выделить следующие причины длительной компиляции проектов на C++: агрессивные настройки оптимизации, плохая физическая архитектура проекта, активное использование шаблонов. Медленная компиляция приводит к тому, снижается качество вносимых в проект изменений, выбираются менее эффективные архитектуры системы.

Существует несколько способов уменьшения времени компиляции. Например, использование прекомпилированных заголовочных файлов (precompiled headers), позволяет повысить эффективность компиляции на 30 %. Одним их основных способов снижения скорости компиляции является распределенная компиляция.

В работе предложен проект системы распределенной компиляции и его реализация для компилятора Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 13, который идёт в поставке Microsoft Visual Studio 2003.

1. Системы распределенной компиляции. В настоящее время существует несколько систем, способных осуществлять распределённую компиляцию. К ним можно отнести универсальные системы распределения и миграции процессов *OpenMOSIX*, распределённые версии утилиты *make* (*pvmmake, dmake, Cook*), специализированные системы, рассчитанные на специфику компиляции *C* и *C++* (*distcc, Teambuilder*). Используются также и смешанные подходы (*IncrediBuild*).

Системы *pvmmake, dmake, Cook* используют сетевые диски для хранения исходных файлов, и, таким образом, системы решают только задачи распределения заданий, а всё сетевое взаимодействие обеспечивается операционной системой. Однако операционная система не обладает дополнительной информацией о специфике решаемой задачи, из-за чего не может обеспечить эффективное сетевое взаимодействие.

Системы *distcc, TeamBuilder*, основанные на распределении выхода препроцессора, требуют для представления промежуточного кода (после выполнения директив препроцессирования и макроподстановок) очень

много места, что дальнейшее распределение для компиляции приводит к существенным расходам на сетевое взаимодействие. Кроме того, препроцессирование также занимает некоторое время, зависящее от компилятора и исходного кода. Так как при данном подходе всё препроцессирование выполняется локально, это становится узким местом системы и может существенно ограничить масштабируемость системы.

Ряд систем (*OpenMOSIX, IncrediBuild*) реализуют идею удалённого выполнения процессов с поддержкой эмуляции виртуального окружения процесса, управляющего компиляцией. Под окружением понимается совокупность файловой системы, переменных окружения, сетевого имени, реестра и других параметров, уникальных для компьютера. Такой подход позволяет обеспечить высокую производительность и масштабируемость системы. Но, например, система *OpenMOSIX* неэффективна при выполнении большого количества процессов с небольшим временем жизни, что характерно для компиляции отдельных файлов. Следует отметить сложность реализации такой идеи из-за тесной интеграции с операционной системой.

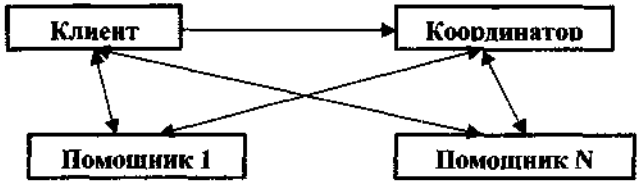
В некоторых системах распределённой компиляции (*TeamBuilder, IncrediBuild*) используется специальный сервер, называемый координатором или планировщиком, в функции которого входит балансировка нагрузки при обслуживании нескольких пользователей. Однако общий координатор представляет собой узкое место системы с точки зрения надёжности и безопасности. В тоже время бессерверные архитектуры (*distcc*) лишены единого центра управления, распределяющего нагрузку в кластере.

Выполнение распределенной компиляции накладывает определенные требования на безопасность этой работы. Из рассмотренных выше систем только *distcc* позволяет использовать *SSH* соединения для авторизации и шифрования передаваемых данных. Однако использование криптографически стойких линий связи приводит к существенным издержкам при передаче данных и снижению эффективности системы.

Поскольку исходные *C++*-файлы имеют очень высокий коэффициент при сжатии, ряд систем (*distcc, IncrediBuild*) использует архивацию файлов при передаче их удаленным компьютерам для компиляции. Однако, большинство алгоритмов архивации работают медленно относительно скорости передачи по сети, что приводит в конечном счёте к снижению производительности в достаточно быстрых сетях.

2. Архитектура системы распределенной компиляции C++-проектов.

Пусть $Pr = (F, C)$ – C++-проект, где F – множество файлов, а C – множество конфигураций. Любой паре (F_i, C_j), где $F_i \in F$ и $C_j \in C$, соответствует задание T_{ij}, которое надо выполнить для компиляции файла F_i в конфигурации C_j. Задание представляет собой тройку (p, s, e), где p – процесс, который нужно выполнить, s – командная строка, содержащая параметры процесса, e – совокупность переменных окружения процесса



p.

Общая схема распределенной компиляции представлена на рис.1.

Рис.1. Компоненты системы распределенной компиляции

Система включает следующие компоненты:

- клиент;
- координатор;
- помощник.

Клиент – это модуль, который производит компиляцию. Помощник отвечает за предоставление распределенных вычислительных ресурсов. Помощник может быть локальным, т.е. Запущенным на узле клиента либо удаленным. Координатор планирует распределение помощников между клиентами.

В системе клиент выполняет следующие задачи:

- на основе файла проекта создаёт список заданий;
- устанавливает соединение с координатором, который определяет подмножество помощников, чьи вычислительные ресурсы клиент будет использовать;
- принимает соединения помощников и обслуживает их запросы.

Помощник является службой работающей в фоновом режиме и выполняет следующие действия:

- устанавливает соединения с клиентом;
- запрашивает у клиента задание на компиляцию;
- запускает процесс компиляции;

- возвращает результаты компиляции (созданные при компиляции файлы и содержимое *stdout*-потока компилятора) и запрашивает новое задание.

Если в процессе компиляции потребуются дополнительные файлы (например, заголовочные файлы или файлы с отладочной информацией), то помощник обеспечивает доступ к ним в режиме эмуляции среды выполнения клиента, который содержит все файлы проекта.

Режим эмуляции реализуется с помощью механизма перехвата запросов компилятора к функциям файловой системы *ОС*, которые перенаправляется клиенту /1/. Все затребованные файлы помощником кэшируются. Для перехвата обращений компилятора к функциям *ОС* используется библиотека *Detours*.

Взаимодействие с клиентом построено на основе портов завершения ввода-вывода. Для повышения безопасности при выполнении заданий используется объект ядра *ОС* «задание», ограничивающий доступ выполняемых процессов к ресурсам компьютера. Для оптимизации операций, связанных с записью и чтением файлов на диск, файлы создаются с пометкой «временный», что указывает диспетчеру кэша *Windows* выполнять их сброс на диск в последнюю очередь /1/.

Заключение.

Тестирование системы производилось в *LAN* с пропускной способностью *100 Мбит/с*. Координатор был установлен на компьютере с процессором *Athlon XP 2800+ 2,09ghz*. Помощники работали на компьютерах с процессором *Intel Pentium 4* с тактовой частотой *3,00 ghz* или *2,80 ghz* с включенной технологией *Hyper Threading*. Все компьютеры имели *1* гигабайт оперативной памяти и работали под управлением операционной системы *Windows XP Professional SP2* или *Windows 2000 Professional SP4*.

Таблица

Число процессоров	***Время компиляции (с)***			
	Visual Studio 2003	***Система с локальным помощником***	***Система без локального помощника***	***IncrediBuild 2.30***
1	140	140	-	150
2	-	80	190	85
5	-	47	55	48
8	-	30	28	-
10	-	22	20	-

В качестве теста была взята отладочная версия библиотеки BCG 6.4 (Business Components Gallery). Тестирование проводилось в двух режимах: с локальным помощником и без него. Данный тест также был выполнен с использованием стандартного метода компиляции на одном узле с помощью *Visual Studio 2003* и с помощью коммерческой системы *IncrediBuild 2.30*, в которой всегда используется локальный помощник. Результаты тестирования представлены в таблице. Качественное поведение систем компиляции представлено на рис.2.

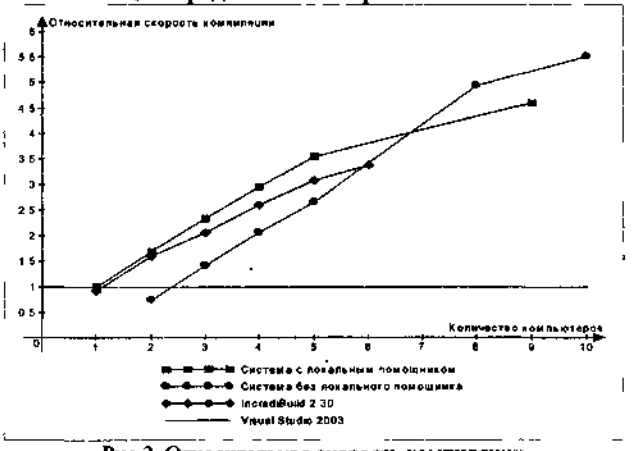


Рис.2. Относительная скорость компиляции

На тесте разработанная система распределенной компиляции на 10 компьютерах снижает скорость компиляции в *5-6* раз. Кроме того, с увеличением числа компьютеров, привлекаемых для распределенной компиляции, система имеет тенденцию к более быстрому наращиванию производительности, по сравнению с другими системами компиляции, в том числе, и коммерческими.

Предложенные в работе методы распределенной обработки могут быть применены и для решения других задач, например, по ускорению работы программных продуктов, решающих различные ресурсоёмкие задачи, таких как, сжатие видео и звука, обработка изображений.

Литература

1 Соломон Д., Руссинович М. Внутреннее устройство Microsoft Windows 2000. Мастер-класс. / Пер. С англ. - Спб: Питер; М.: Издательско-торговый дом «Русская редакция», 2004г. - 746с.