Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра вычислительной техники

Разработка и реализация инструментов сборки программного обеспечения

Выпускная квалификационная работа

Студент: Калугин Федор Игоревич, группа P3410

Руководитель: Косяков Михаил Сергеевич, к.т.н., доцент кафедры ВТ, Университет ИТМО

Санкт-Петербург, 2018

Цели и задачи

Цель – разработать систему сборки программного обеспечения для уменьшения времени сборки C/C++ проектов.

Задачи:

- Исследовать существующие системы сборки и их возможности;
- Разработать и реализовать систему сборки, позволяющую уменьшить время сборки проектов;
- Оценить эффективность предложенной реализации.

Сравнение систем сборки

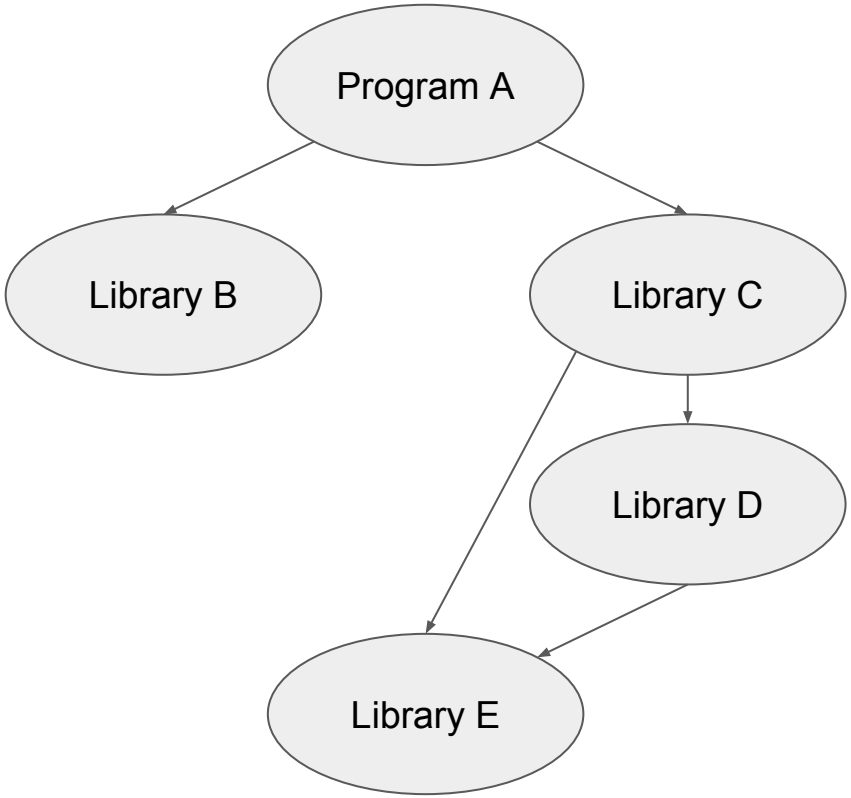
	Make	Ant	Maven	CMake	Gradle	Bazel
Сборка C/C++ проектов				+	+	+
Сборка Java проектов		+	+		+	+
Параллельная сборка	+	+	+	+	+	+
Модульное тестирование		+	+	+	+	+
Интеграция предкомпилированных заголовков (C++)					+	
Распределенная сборка						
Кэширование результатов компиляции						
Создание программного пакета						

Модульная сборка проектов

Этап 1: составление направленного ациклического графа зависимостей проектов.

Этап 2: обратный обход графа в глубину.

Пример графа зависимостей проектов:



Модульное тестирование

Отличия разработанной системы:

- Модульный тест – произвольный проект исполняемого файла;
- Не зависит от прикладной программной платформы тестирования;
- Тесты не исполняются если не было изменений;
- Тесты исполняются снова если они не завершились успешно во время предыдущей сборки.

Параметры тестового стенда

Тестовый стенд:

- Операционная система: Red Hat Enterprise Linux 6.8
- Процессор: Intel Xeon CPU E5-2666 v3 @ 2.90 GHz
- Количество ядер процессора: 8
- Оперативная память: 30 Гбайт

Собираемый проект:

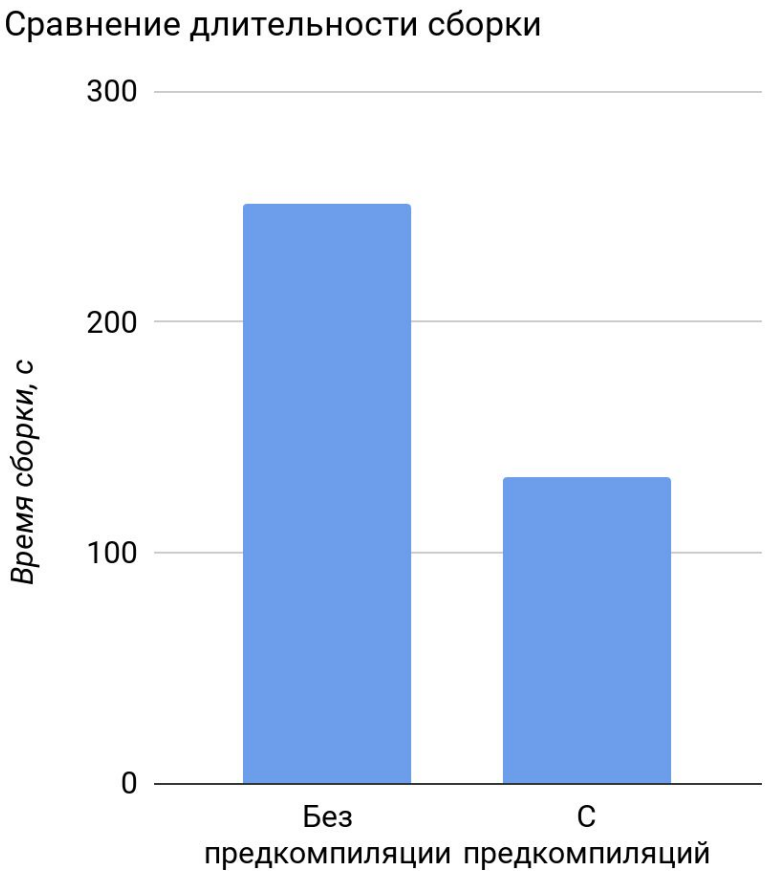
- Язык: C++
- Количество файлов: 665
- Количество строк кода: 163107
- Компилятор: clang 4.0

Предкомпилированные заголовочные файлы

Предкомпилируются крупные и часто использующиеся заголовочные файлы.

Подключаются вместо обычных заголовочных файлов.

При изменении любого из предкомпилированных заголовков перекомпилируются все файлы исходного кода.

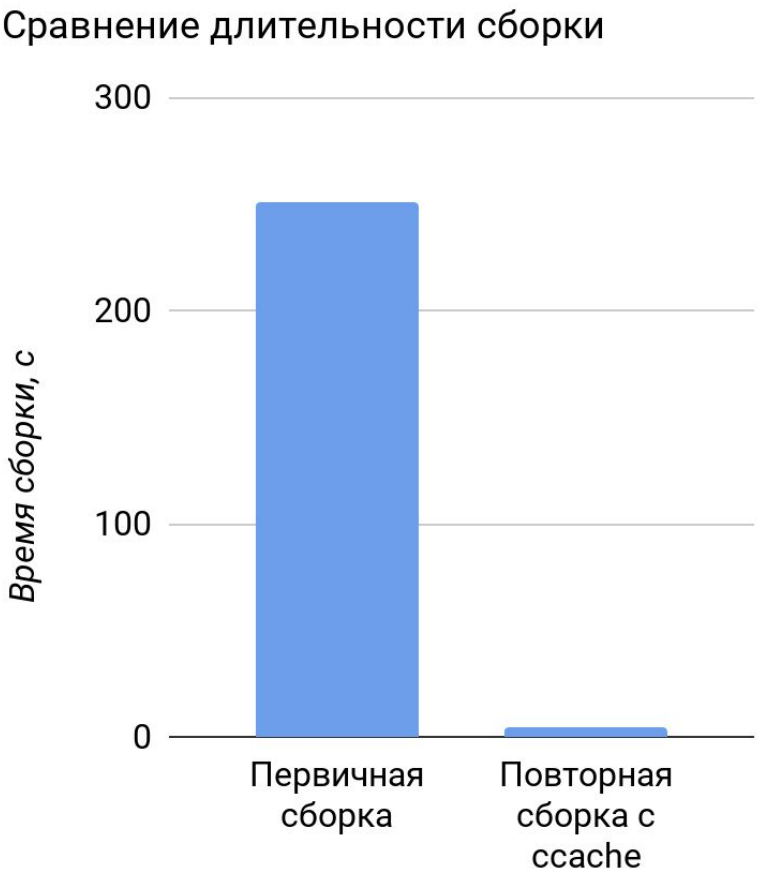


Кэширование результатов компиляции

Помогает при:

- Частой смене веток в системе контроля версий;
- Работе с несколькими копиями проекта;
- Сборке на серверах непрерывной интеграции.

Утилита ccache интегрирована как обертка над компилятором



Кэширование результатов компиляции

Считается хэш сумма:

- файл исходного кода;
- используемые заголовочные файлы;
- строка компиляции;
- информации о компиляторе.

Абсолютные пути препятствуют использованию кэша при сборке в других каталогах.

Распределенная компиляция

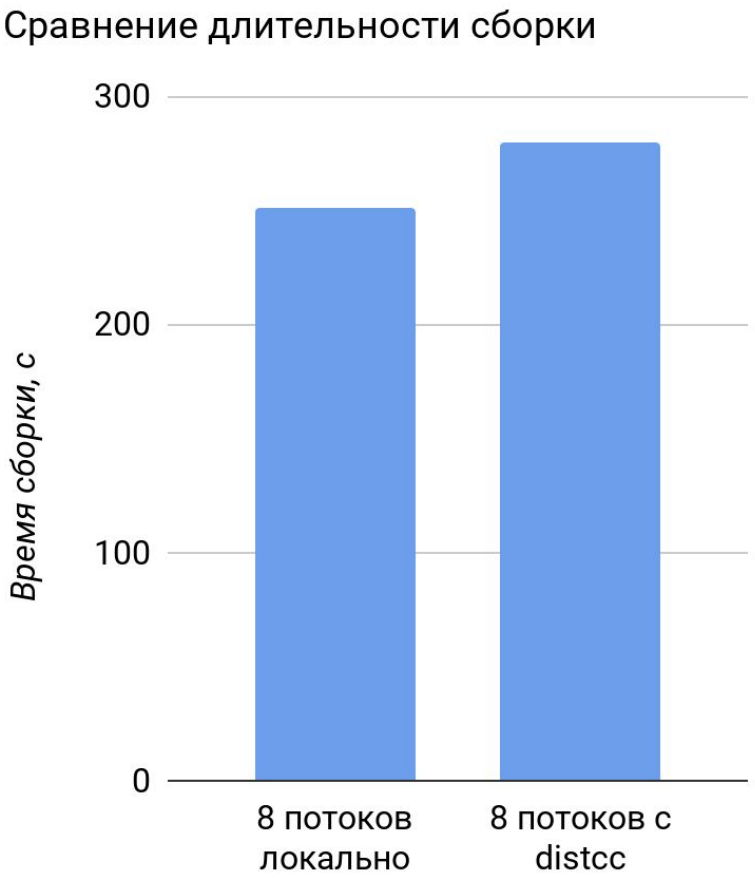
Утилита distcc интегрирована как обертка над компилятором.

Препроцессор выполняется локально. Компиляция на разных машинах.

Конфигурация машин должна быть одинаковая.

Сеть 10 Гбит/с в одном датацентре.

Накладные расходы: 12%



Создание программного пакета

Программный пакет:

- исполняемые файлы;
- библиотеки;
- файлы данных.

Пакет обновлений: только измененные и новые файлы между версиями.

Создание пакета обновлений

Сборка пакета обновлений:

1. Собирается полный пакет текущей версии;
2. Собирается полный пакет новой версии;
3. Сравнивается содержимое пакетов;
4. Новые и измененные файлы архивируются в пакет обновлений.

При сборке новой версии собираются только измененные проекты.

Полный пакет новой версии можно переиспользовать при сборке следующего пакета обновлений.

Результаты работы

- Реализована система сборки C/C++ проектов, позволяющая ускорить сборку до 2 раз с помощью предкомпилированных заголовков, и многократно ускорить повторную сборку проектов за счет кэширования результатов компиляции;
- Реализован механизм распределенной компиляции;
- Предложен механизм выполнения модульных тестов во время сборки;
- Предложен механизм автоматизированного создания пакетов обновлений программных продуктов.

Спасибо за внимание!