Ю.С. Артамонов

КОЛЛЕКТИВНАЯ РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ

(ФГБОУ ВПО «Самарский государственный аэрокосмический университет им. академика С.П. Королева (национальный исследовательский университет)»)

Одной из актуальных проблем в разработке приложений для высокопроизводительных вычислений является быстро возрастающая сложность алгоритмов и систем, что ведёт к замедлению темпов разработки. В настоящее время очень востребована командная разработка, как в промышленном ПО, так и в научном. Во многом это обусловлено тем, что команда может разрабатывать многие модули и подсистемы параллельно, а одному разработчику сложно держать под контролем все аспекты работы приложения.

Общепризнанным стандартом для коллективной разработки ПО являются системы контроля версий (VCS), позволяющие вести работы над проектом большому числу разработчиков. Существует ряд сервисов сети Интернет, позволяющих бесплатно использовать VCS для работы над свободными проектами: Google Code, Github. Опыт показывает, что научные работы можно успешно вести в виде проекта с открытым исходным кодом, используя один из публичных хостингов VCS. Примеры: библиотека OpenCV [1], язык функционального программирования Scala [2].

Другим аспектом коллективной разработки является поддержание работоспособности проекта при частых изменениях различных подсистем разными разработчиками. Значит, необходим этап всестороннего тестирования, как отдельных модулей, так и их интеграции. Для этого разработчики применяют набор практик непрерывной интеграции - Continuous Integration (CI) [3]. Процесс работы над проектом с применением CI показан на рисунке 1. В дополнение к CI существует набор практик по осуществлению эффективной поставки ПО – Continuous Delivery (CD) [4], определяющих то, как проводить поставку ПО, приёмочное тестирование и введение в эксплуатацию максимально эффективно и при помощи средств автоматизации. Практики CI и CD возможно адаптировать для работы над научными проектами в сфере высокопроизводительных вычислений.

Перейдём к специфике работы над научными проектами, в частности при разработке параллельных приложений.

Под окружением мы будем понимать некоторый программно-аппаратный комплекс: совокупность физических ЭВМ, каналов связи, периферийных устройств и ПО, необходимого для работы системы. Хорошим примером является кластер серверов: узлы кластера, высокоскоростные сети передачи данных, система пакетной обработки.

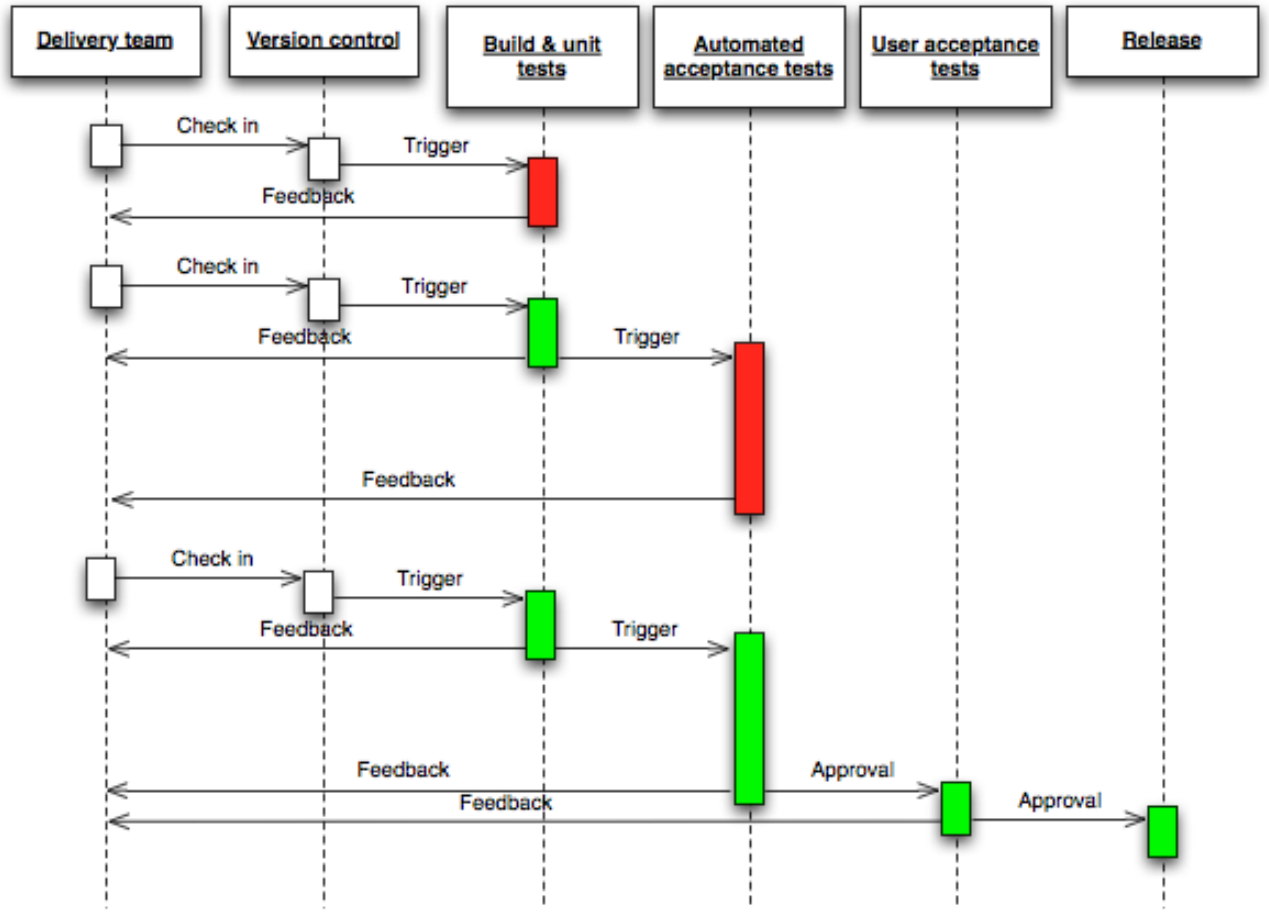

Рис. 1. Диаграмма последовательности процесса непрерывной интеграции

В разработке приложений для научных вычислений есть ряд проблем, отсутствующих в массовой коммерческой разработке:

- сильная зависимость приложений от окружения: нестандартный набор ПО, периферийных устройств, сложная конфигурация оборудования;
- необходимость обеспечения заданной производительности даже для тестовых окружений;
- сложный (длительный) процесс развёртывания ПО в целевом окружении;
- необходимость соблюдать регламент работы с целевым окружением, которое часто является общим для многих групп разработчиков и проектов, а также требует оформления доступа.

Все эти проблемы характерны и при разработке параллельных приложений на базе суперкомпьютера «Сергей Королёв» в СГАУ.

Для автоматизации командной работы над проектами параллельных приложений нами был разработан веб-сервис Templet, работающий на базе Медиацентра СГАУ. Он позволяет:

- просто и быстро начать разработку проекта высокопроизводительного приложения при помощи шаблонов вычислительных методов пакета Templet SDK [5]: портфель задач, конвейер и другие;
- организовать работу команды над проектом с применением VCS, поддерживается Subversion, планируется поддержка Git;

- вести обсуждения проекта, получить доступ к общим окружениям развёртывания и запущенным задачам;
- разворачивать приложение в тестовом и целевом окружениях (поддерживается неограниченное число окружений для проекта);
- отслеживать работу приложения во время продолжительных вычислений;
- получать уведомления о статусе задач в целевых окружениях.

Этапы развёртывания проекта:

- получение исходного кода или бинарных артефактов из VCS;
- добавление задачи в очередь развёртывания;
- загрузка пакета развёртывания в целевое окружение;
- опциональная сборка проекта в целевом окружении;
- запуск.

Далее по расписанию специальный демон-процесс проверяет статус запущенной задачи, добавляет записи в журнал событий задачи, а также высылает уведомления о статусе по e-mail. Поддерживаемые окружения:

- Виртуальные окружения Linux, на базе Debian: возможно использовать любые настроенные машины пользователя, необходим доступ из сети Интернет по протоколу SSH;
- Суперкомпьютер «Сергей Королев»: необходима учётная запись, предоставленная Суперкомпьютерным центром СГАУ.

Проект может быть написан на любом языке программирования и с использованием всех необходимых библиотек, требуется лишь соблюдать минимальные требования к его структуре.

В перспективе мы планируем добавить тип окружений «Облако», скрывающий расположение и технические параметры целевого окружения, а также реализовать управление облаком и потоком вычислительных задач.

Литература

1. http://opencv.willowgarage.com/wiki/ - Open Source Computer Vision Library
2. http://www.scala-lang.org/node/8579 - Scala Team Wins ERC Grant
3. http://martinfowler.com/articles/continuousIntegration.html - Martin Fowler - Continuous Integration
4. http://jamesbetteley.wordpress.com/2011/08/04/continuous-delivery/ - James Betteley - 8 Principles of Continuous Delivery
5. Востокин С.В. Templet – метод процессно-ориентированного моделирования параллелизма // Программные продукты и системы, 2012. №3. с. 9 – 12