АРХИТЕКТУРА ВЕБ-ИНТЕРФЕЙСА РАСПАРАЛЛЕЛИВАНИЯ ПРОГРАММ НА УРОВНЕ ИСХОДНОГО КОДА

А.С. Быль, С.А. Немнюгин, Д.А. Пузырев, А. Ю. Петров

Введение

В настоящее время имеется тенденция создания веб-сервисов для взаимодействия пользователей с консольными приложениями. Рассматриваются приложения для модификации исходного кода программ.

Во время работы пользователя с приложением часто возникают дополнительные рутинные действия, например, проверка данных перед обработкой, перемещение файлов на сервер, уведомления о завершении некоторого этапа и так далее.

Иногда требуется участие специалиста для ручной обработки. Дополнительные требования бывают различными, и нелогично каждый раз изменять приложение.

Поэтому был создан сервис, в котором приложение является нижним уровнем, а пользователь взаимодействует исключительно с веб-интерфейсом в браузере.

В качестве начальной модели в нашей разработке было использовано ядро сервиса PARSERPLUS.COM. На этом сайте проиводится оптимизация и распарралеливание программного обеспечения на уровне исходного кода, предлагается услугу полуавтоматической обработки заданий через сеть Интернет. Задание включает передаваемый текст программы, служебную информацию, в случае необходимости, комментарии пользователя. Сам сервис требовал улучшений с точки зрения удобства работы с заявками для конечного потребителя, эксперта и администратора. К новым требованиям также добавилась необходимость разработки гибкой архитектуры для веб-сервиса взаимодействия с приложениями модификации программ.

Наша разработка позволила улучшить время отклика, расширяемость и добавить новые возможности для пользователей системы.

Описание задачи

Исходная модель работала с приложением, называемым Анализатор. Приложение модифицирует исходные коды программ на языке C++ для выполнения их в параллельной среде.

Добавляются конструкции стандарта OpenMP для распараллеливания циклов FOR, осуществляется поиск самих циклов. После запуска приложение обрабатывает файлы в директории и после помещает результат в выходную директорию. Видно, что само приложение предоставляет только базовые возможности. Поэтому ранее был создан сервис PARSERPLUS.COM, который обеспечил следующую функциональность: работу с файлами, уведомление экспертов и пользователей, создание базы данных о заданиях.

Архитектуры веб-сервиса дсоответствует поставленным требованиям:

1) Полный контроль и прозрачность процесса модификации кода (панель администратора, уведомления, проверки данных)
2) На выбор: требование эксперта в случае необходимости, или автоматическая модификация.
3) Сохранение статистики.
4) Избавление от рутинных операций.
5) Прозрачное подключение других приложений.
6) Подключение других веб-интерфейсов.

Задача реализована на следующей платформе:

- Веб-сервер Apache 2.2.14
- Серверный язык Php 5.3.13
- Javascript в качестве клиентского языка
- База данных MySQL 5.5.24
- Html в качестве средства представления

Веб-интерфейс сервиса для работы с Анализатором состоит из:

- Формы отправки исходного кода
- Странички управления
- Странички модификации кода

Для доступа к сервису достаточно веб-браузера с поддержкой Javascript.

Архитектура сервиса

При создании сервиса использован шаблон проектирования Model-View-Controller. Данный подход позволил структурировать программу и упростить разработку сервисов данного вида.

На [[m-v-c.jpg]] показан общий вид шаблона. Модель совершает выбор значений из DB и обработку данных. Вид – это конечное представление, интерфейс. Контроллер соединяет вид с моделью данных.

Достаточно легко можно поменять или подключить другую модель, представление или управляющую структуру.

При работе с требованиями к новому сервису были сформулированы принципы построения архитектуры.

1) Наличие управляющей структуры для заданий. Гарантирует, что задания будут нужным образом обработаны.
2) Использование шаблона M-V-C внутри сервиса.
3) Включение инструментов для эксперта. Возможность изменения исходных кодов вручную и тестирования их в приложении.

Программный комплекс для сервиса с Анализатором состоит из компонентов, хорошо видных на рисунке [[web-serv-diags.jpg]]. Прямоугольником обозначен компонент собственного программного комплекса. Эллипсом - используемый готовый компонент. Рассмотрим взаимодействие компонентов сервиса с момента подачи заявки пользователем. Когда пользователь заходит на веб-страничку, начинает работать Start-tool. Включает в себя страничку с формой, алгоритм проверки данных, создание новой записи в базе данных (DB). Пользователь загружает данные на сервер, где они проверяются. Если проверка успешна, то создается запись для задания в базе данных, попытка запустить обработку на Анализаторе (Analyzer), который, в свою очередь, при завершении задания обновляет запись в базе данных.

Стоит остановиться подробнее на самих заданиях. Задание считается назначенным, когда исходный код проходит проверку для запуска на Анализаторе. Внутри сервиса задание проходит несколько этапов, как того требует выбранный алгоритм работы с приложением. Отражается в виде поля Состояние для записей о задании. Состояния делятся на начальные, промежуточные и терминальные. Терминальный статус задания означает, что исполнение задания завершено.

Контроль выполнения заданий осуществляет компонент Daemon-tool. Компонент содержит Модель Состояний, которая определяет переходы между состояниями и соответствует алгоритму работы сервиса. В ней каждому состоянию соответствуют действия над записью и переход в следующее состояние.

Следующий компонент – это Admin-tool, показывает уведомления для эксперта и состояние сервиса. С помощью компонента Code Modifer эксперт работает с исходными кодами.

Подробное описание компонентов

6) DB. Реляционная база данных для хранения информации о заданиях.
7) Analyzer. Инструмент, модифицирующий исходный код. Переводит задание в терминальное состояние. Сервис позволяет использовать и другие приложения, но выбран был Анализатор.
8) Start-tool. Форма для сбора данных и начальный обработчик данных. На [[web-serv-diags1.jpg]] пример диаграммы деятельности для приложения "Анализатор". Алгоритм:

3.1 Проверить данные. Исходный код на расширение “.c” и “.cpp”, а также на наличие циклов “for”. Вводимые данные проверяются соответственно: адрес электронной почты на шаблона электронной почты и так далее. В случае неудачи сообщить ошибку пользователю.

3.2. Создать запись в DB примерно со следующими параметрами (id, condition, condition_time, files_url, email). Присвоить condition = (NEW, EXPERT).

3.3. Если condition = NEW, то обработать код с помощью Анализатора. В зависимости от результатов, присвоить condition = (REPEAT, ABORTED, GO)

4) Daemon-tool. Программа в виде демона, отслеживающая некоторые состояния. Алгоритм наглядно представлен на [[web-serv-diags2.jpg]]

4.1. Если condition = REPEAT, то, повторить обработку не позже, чем через время “t1”

4.2. Если condition = GO уже более времени “t2”, то требуется разобраться в причине долгого отклика, поставить состояние равно EXPERT.

4.3. Если condition = ABORTED, то послать пользователю причину ошибки.

4.4. Если condition = FINISHED, то послать пользователю обработанный код программы .

5 Admin-tool. Веб-интерфейс для мониторинга заданий (архив заданий, текущие исполняемые, последние добавленные). Уведомления эксперта о назначенных ему заданиях.

6 Code Modifier. Инструмент для модификации исходного кода экспертом.

Редактируется текст программы и состояние задания. Эксперт может получить результат для этого кода на Анализаторе, чтобы сравнить со своим результатом.

Заключение

В проделанной работе мы получили модель архитектуры онлайн-сервиса и улучшили текущий сервис распараллеливания исходного кода на PARSERPLUS.COM. Новый сервис служит примером нашей модели. Г

Главные ее преимущества.

- Алгоритм работы сервиса определяется в Модели состояний, что дает легкую перестройку алгоритма.
- Работа с исходным кодом и Анализатором в режиме отладки.

- Удобный и продуманный веб-интерфейс
- Полностью автоматическая работа, что позволяет в некоторых случаях обойтись без участия эксперта и быстро получить результат
- Подключение других приложений и интерфейсов

Наиболее интересным является использование других приложений и Модели состояний. При подключении к сервису нового приложения может потребоваться дополнительные действия над заданием. Допустим, если задание уже обработано, то нужно выложить его в общий доступ. Тогда требуется создать новое состояние, встроить его в последовательность с другими состояниями и определить для него действия над заданием.

Веб-сервис работает в качестве набора взаимосвязанных компонент. Интерфейс для обмена данными между компонентами максимально упрощен и стандартизирован, чтобы элементы сервисов, построенных на этой модели, были совместимы друг с другом. Повторное использование компонент позволяет сократить время разработки и проектирования новых сервисов.

ЛИТЕРАТУРА:

1. "Web Services Glossary", W3C, February 11, 2004. Retrieved 2011-04-22.
2. “Аутсорсинг распараллеливание и оптимизации, как услуга, предоставляемая пользователю через Интернет”, Д.А. Пузырев, А.Ю. Петров, С.А. Немнюгин, Международная суперкомпьютерная конференция, Новосибирск, 19-24 сентября 2011 г.
3. “Model-View-Controller Architecture”, John Deacon, “John Deacon Computer Systems Development, Consulting & Training”, 1995 г.
4. “Understanding MVC in PHP”, Joe Stump, “O’Really”, 2005 г.