УДК 004.415

ПРИМЕНЕНИЕ КОМПОНЕНТНОГО ПОДХОДА ПРИ ПРОЕКТИРОВАНИИ АРХИТЕКТУРЫ КЛИЕНТСКОГО ПО РЕНДЕРФЕРМЫ

Никитин В. Н.
научный руководитель канд. техн. наук. Кузьмин Д. А.
Сибирский федеральный университет

Введение

Современное развитие медиаискусства породило ряд сложных задач, связанных с применением компьютерной графики и анимации. Примерами могут служить реклама и кинематограф, в которых требуется создание реалистичного видеоряда, либо статических изображений.

Процесс рендеринга считается одной из наиболее сложных вычислительных задач, так как изображения получают путём обсчёта математической модели с применением довольно сложных рекурсивных алгоритмов. Например, на среднестатистическом компьютере рендеринг пятиминутного видеоролика разрешением Full HD может занимать до нескольких дней [1].

Задачи подобного рода хорошо решаются с использованием распределённых вычислений [1]. Применение такого подхода позволяет существенно сократить время рендеринга, а также снизить расходы организации.

С этой целью в настоящее время создаётся большое количество сервисов, предоставляющих возможность использовать объединённые в кластер компьютеры для решения задач рендеринга. Однако, ввиду пока ещё недостаточной развитости такого рода облачных сервисов, разработчики каждый раз вынуждены решать одни и те же задачи, которые в большинстве своём могут быть реализованы стандартным образом. Поэтому лучшим выходом было бы иметь готовый набор программного обеспечения, на основе которого можно построить собственный сервис. Он позволил бы разработчикам сконцентрироваться не на проектировании общей базы, а на реализации конкретного необходимого функционала.

Сервис распределённого рендеринга СФУ

С целью упрощения развёртывания и разработки основы сервиса распределённого рендеринга (далее - рендерферма) было решено создать универсальное решение и проверить его работоспособность на базе суперкомпьютерного центра Сибирского федерального университета [2].

Архитектура сервиса включает несколько компонентов (рисунок 1):

1) Сервер управления рендерингом и вычислительные узлы. На сервере работает серверное ПО, управляющее процессом рендеринга и взаимодействующее с вычислительными узлами;

2) Концентратор клиентов. Отвечает за обработку запросов, поступающих от пользователя;

3) Клиентское ПО. Выполняет работу по упаковке проекта для последующей его передачи и расчёта на рендерферме. Также включает клиент для управления процессом расчёта проекта.

Отличительной особенностью архитектуры рендерфермы СФУ и разработанного на её основе ПО является то, что она универсальна и может быть применена к любой платформе. Например, может присутствовать любое количество серверных компонентов, что позволяет объединять распределённые вычислительные ресурсы в единую вычислительную среду. Это позволяет построить

специализированную GRID-систему для выполнения задач рендеринга, имеющую значительную вычислительную мощность.

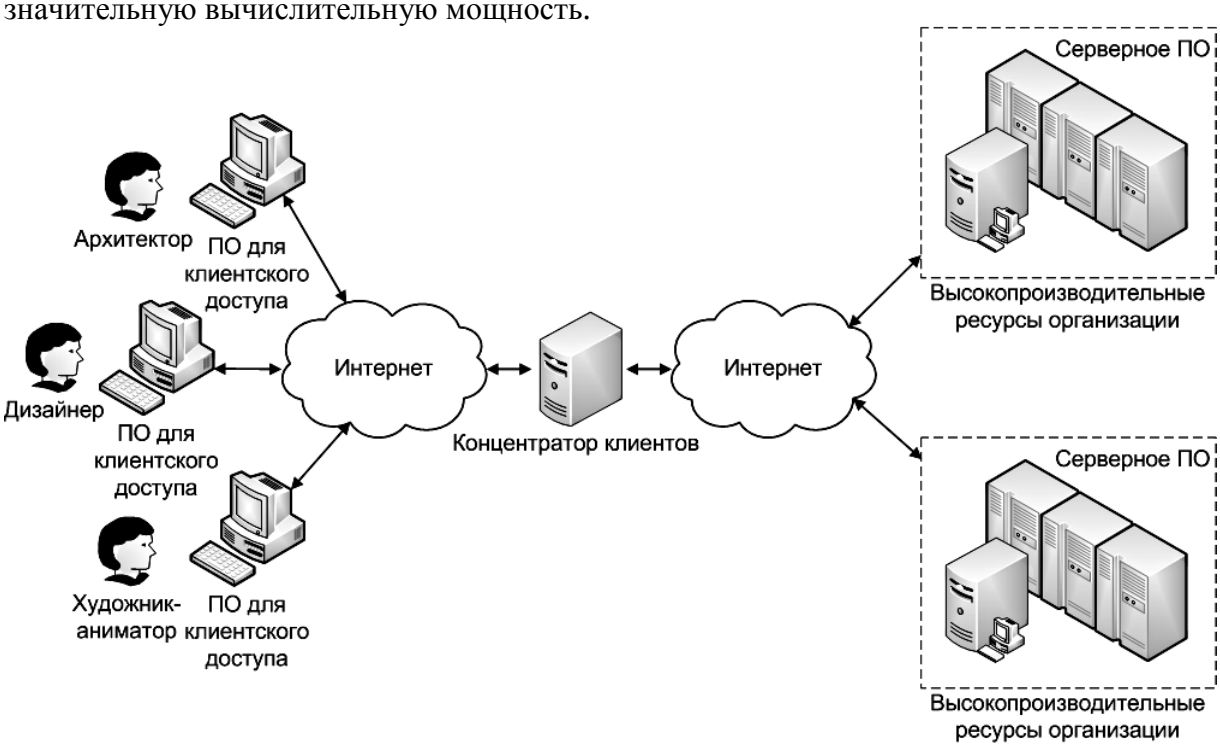

Рисунок 1 - Архитектура сервиса распределённого рендеринга СФУ

Проектирование архитектуры клиентского ПО рендерфермы СФУ

Комплекс клиентского программного обеспечения разрабатывается для удобного взаимодействия пользователей с рендерфермой.

Комплекс состоит из нескольких приложений:

1) Встраиваемый модуль (плагин) - работает в среде пакета 3D моделирования. В обязанности встраиваемого модуля входит:

- внедрение в пакет 3D моделирования;
- получение информации о проекте пользователя;
- сбор ресурсов проекта (файл сцены, текстуры и др.), упаковка и отправка сцены;

2) Менеджер проектов - предоставляет интерфейс для управления процессом рендеринга. Это графическое приложение, в задачи которого входит:

- отображение списка проектов пользователя;
- управление процессом рендеринга проектов;
- скачивание результата.

3) Менеджер пакетов – производит автоматическое обновление клиентского ПО. Выполняет функции универсального установщика и программы обновления.

Основными требованиями при проектировании архитектуры являлись расширяемость и возможность повторного использования кода. Для их удовлетворения хорошо подходит компонентная модель [3]. Её суть заключается в том, что всё ПО строится из отдельных модулей-компонентов. Каждый компонент реализует определённый интерфейс и, таким образом, представляет собой некий “чёрный ящик”, реализация которого скрыта. Компоненты могут зависеть только от интерфейсов, и независимы от реализации.

Также при построении архитектуры такой системы удобно применить ряд архитектурных принципов и паттернов, позволяющих уменьшить зависимости между компонентами. Одним из основных принципов при построении подобных систем

является инверсия управления [4]. Также применяются паттерны «отделённый интерфейс» и «дополнительный модуль» [5].

С целью упрощения подключаемых модулей, а также упрощения конфигурирования приложения, может применяться специальный объект, называемый IoC-контейнером. В зависимости от конфигурации приложения, которая может находиться в файле, базе данных или другом хранилище, он загружает соответствующие компоненты и осуществляет их связывание во время выполнения программы.

Применение данных техник позволяет построить расширяемую слабосвязанную архитектуру, на основе которой можно реализовать легко поддерживаемые приложения, а применение компонентного подхода обеспечивает повторное использование кода.

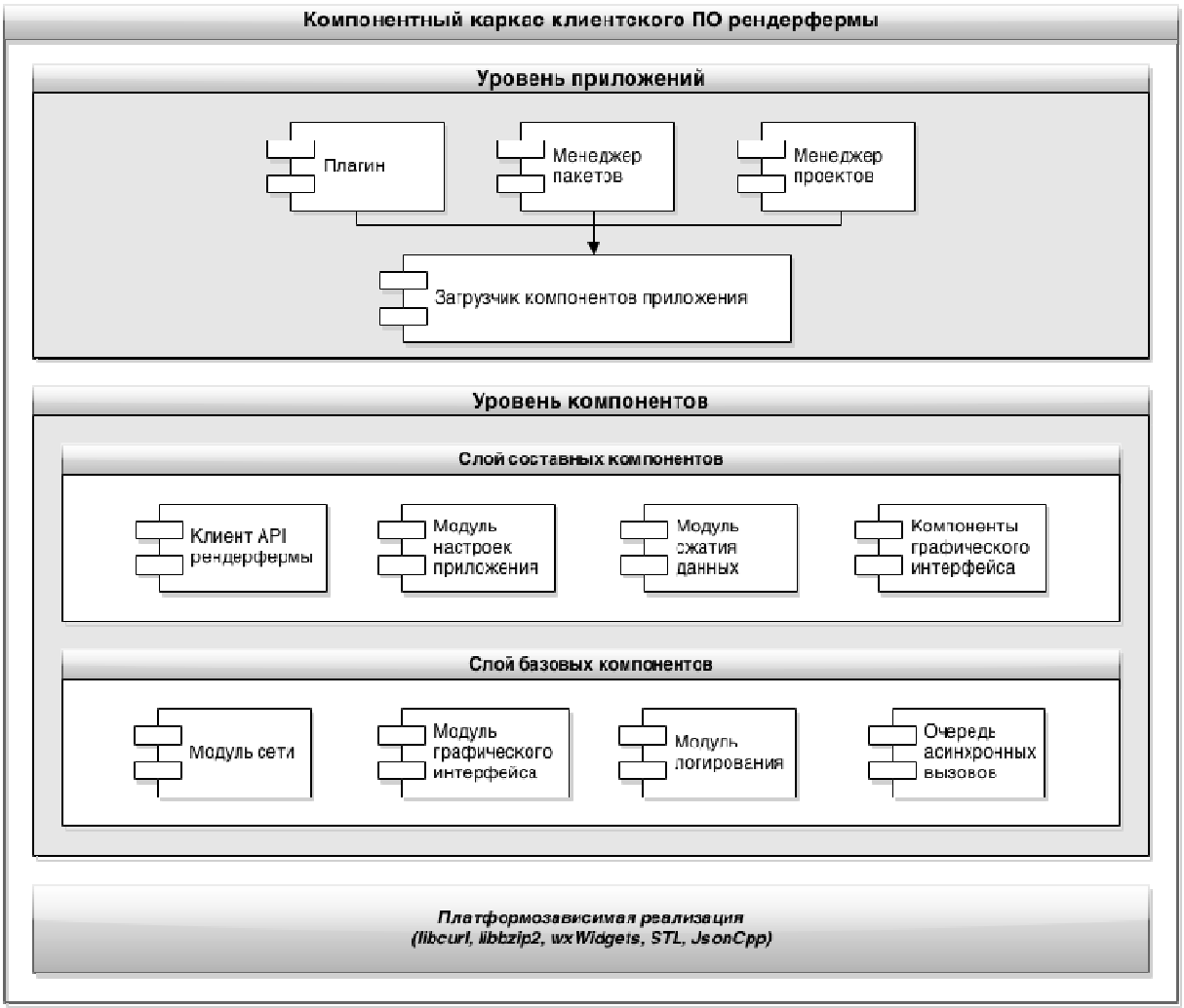

Рисунок 2 - Архитектурная модель клиентского ПО рендерфермы

При определении конкретных компонентов следует отталкиваться от функционала, реализуемого данной системой. В частности:

1) Сжатие данных
2) Взаимодействие с сетью
3) Взаимодействие с API концентратора клиентов
4) Настройки приложения
5) Графический интерфейс

6) Взаимодействие с пакетами моделирования

7) Очередь асинхронных вызовов

8) Логирование

Разработанная архитектура представлена на рисунке 2.

На уровне платформозависимой реализации находятся библиотеки, необходимые для работы компонентов системы на различных платформах.

Выше расположен уровень компонентов, включающий два слоя:

- слой базовых компонентов: компоненты данного слоя полностью независимы друг от друга ни по интерфейсам, ни по реализации. Они могут распространяться отдельно;

- слой составных компонентов: модули данного слоя построены на основе модулей базового слоя и имеют от них зависимости по интерфейсам. Повторное использование возможно при наличии в системе модулей, реализующих необходимые интерфейсы зависимостей.

Самый верхний уровень - уровень приложений. На данном уровне расположены конечные приложения, которые используют функциональность компонентов, объединяя их в единую систему.

Заключение

Применение компонентного подхода может значительно упростить проектирование расширяемой, легко поддерживаемой архитектуры системы со слабой связностью компонентов, и является хорошей альтернативой простому объектно-ориентированному проектированию.

Разработанная архитектура клиентского ПО может успешно применяться в проектах различного уровня, где требуется эффективное взаимодействие клиент-сервер, и быстрая адаптация ПО к введению нового функционала.

На данный момент большая часть функциональности реализована и успешно апробирована в работе рендерфермы СФУ.

Список литературы

1 Панасюк А.И., Астриков Д.Ю., Кузьмин Д.А. Эффективность использования вычислительных ресурсов в процессах распределенной визуализации трехмерных моделей // Сборник трудов Second International Conference «Cluster Computing» CC. - 2013. - С. 166–174.

2 Clemens Szyperski Component Software: Beyond Object-Oriented Programming. Second Edition. London: Pearson Education Ltd., 2002.

3 Кузьмин Д.А. Сибирский Федеральный Университет (СФУ) // Суперкомпьютеры. - № 10. - С. 35.

4 Нильссон Д. Применение DDD и шаблонов проектирования: проблемно-ориентированное проектирование приложений с примерами на C# и .NET. М.: Вильямс, 2008. 560 с.

5 Фаулер М. Архитектура корпоративных программных приложений. М.: Вильямс, 2006. 544 с.