Архитектура библиотеки параллельных геометрических вычислений с расширяемой функциональностью

С.И. Ротков, П.Ю. Лазарев
Нижегородский государственный архитектурно строительный университет
Нижний Новгород, Россия
rotkov@nngasu.ru, pavellazarev@yandex.ru

Аннотация

В данной статье описана архитектура библиотеки для параллельных и распределенных геометрических и математических вычислений с расширяемой функциональностью.

В статье рассматриваются методы расширения функциональности библиотеки и способы использования уже реализованных алгоритмов.

Ключевые слова: *параллельное программирование, распределенное программирование, клиент, сервер, плагин, plug-in*

1. ВВЕДЕНИЕ

В компьютерной графике существует множество вычислительно-емких задач, таких как автоматическое определение видов проецируемого объекта с целью выбора оптимального аппарата проецирования [6], удаление невидимых линий в сложноструктурированных моделях [5, 7], различные булевские операции над составными моделями, рендеринг и т.д. При работе со сложными структурированными моделями возникает потребность в вычислительной мощности, значительно превосходящими возможности даже современных компьютеров. Одним из способов решения проблемы является распараллеливание алгоритмов этих задач.

В работах [5-7] предложены алгоритмы удаления невидимых линий, допускающие распараллеливание вычислительного процесса, основанного на принципах диакоптики, т.е. обработки информации по частям. При этом предполагается, что все составные части геометрической модели пространственного объекта проецируются на картинную плоскость независимо друг от друга при одном и том же аппарате проецирования. Полученные при этом линии очерков составных частей модели могут перекрывать друг друга и анализ их перекрытия дает возможность решить задачу удаления невидимых линий без анализа всей модели, как это делается практически во всех системах геометрического моделирования и компьютерной графики пространственных объектов.

При создании параллельных версий алгоритмов широко используются различные библиотеки, такие как MPI, PThreads, OpenMP и т.д. [3] Однако одна и та же задача может иметь множество алгоритмов решения, каждый из которых лучше подходит для конкретных случаев. Таким образом, при создании приложения работающего со сложными моделями, разработчики будут вынуждены использовать различные библиотеки от различных производителей, в которых уже реализованы те или иные алгоритмы, что значительно затрудняет разработку, т.к. различные библиотеки имеют различные программные интерфейсы.

Таким образом, возникает необходимость в создании библиотеки параллельных и распределенных вычислений, функциональность которой может быть легко расширена, а использование уже реализованных на ее базе алгоритмов было бы простым.

2. ТРЕБОВАНИЯ К СИСТЕМЕ

1. **Легкость использования библиотеки.** Библиотека должна предоставлять унифицированный программный интерфейс для использования всех реализованных алгоритмов.

2. **Возможность расширения функциональности.** Библиотека должна предоставлять возможность простого расширения функциональности как за счет разработки новых модулей, так за счет подключения уже разработанных сторонними производителями.

3. **Возможность использования собственных форматов данных.** Библиотека должна быть независима от используемых разработчиками форматов данных.

3. АРХИТЕКТУРА БИБЛИОТЕКИ

Архитектура рассматриваемой библиотеки основана на технологии клиент-сервер и состоит из трех элементов: клиентской части, сервера и удаленных агентов. [1,2]

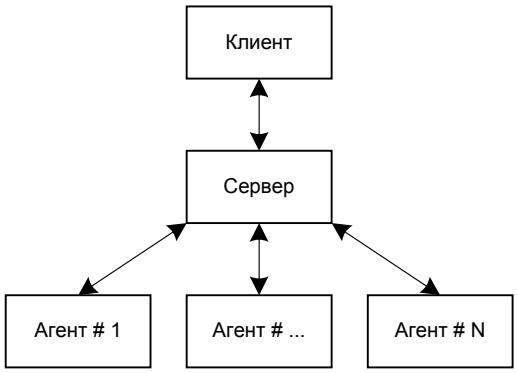

Рис. 1. Общая архитектура

Библиотека реализует форматно-независимую передачу данных между клиентом, сервером и агентами. Расширяемость функциональности библиотеки достигается использованием технологии *плагинов* – подключаемых модулей, реализующих некоторый заданный интерфейс.

3.1 Клиентская часть

Клиент состоит из трех модулей: клиентского приложения (КП), реализующего пользовательский интерфейс, Модуля данных (МД), реализующего необходимые

пользовательские форматы данных и работу с ними, и Модуль пересылки данных (МПД) – стандартного модуля предоставляемого библиотекой.

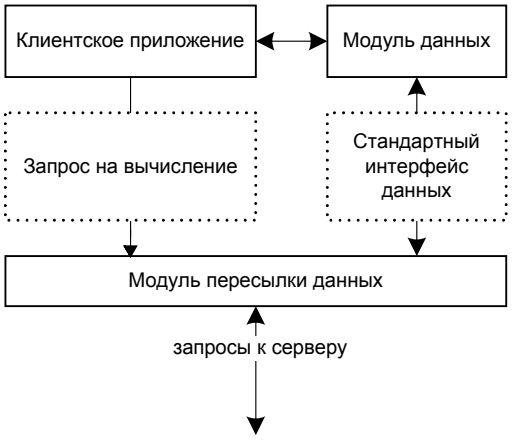

Рис. 2. Архитектура клиента

Клиентское приложение, делая запрос в МПД на обработку пакета данных, указывает идентификаторы алгоритма обработки и типа данных. Эти идентификаторы используются для поиска модуля на сервере, способного обработать задание.

Следующий пример показывает код запроса на удаление невидимых линий у полигональной модели построенной из набора треугольников.

Пример (на C++): функция удаления невидимых (скрытых) линий у полигональной модели.

```
void function DeleteHiddenLines (UserData * pData, SRDModul * pModul)
{
   pModul->SendRequest( pData,
                        HIDDEN_LINE_REMOVE,
                        TYPE_TRIANGLES_3D);
}
```

В функцию подаются указатели на пользовательские данные и модуль пересылки данных.

UserData – класс реализующий стандартный интерфейс данных.

SRDModul – стандартный класс библиотеки, ответственный за передачу данных.

HIDDEN_LINE_REMOVE – идентификатор алгоритма удаления невидимых линий

TYPE_TRIANGLES_3D – идентификатор типа, показывающий что данные представляют собой набор треугольников.

Все функции модуля пересылки данных асинхронны, т.е. после вызова функции программа продолжает работу, не ожидая окончания обработки данных. Это позволяет программе взаимодействовать с пользователем и не «подвисать» во время ожидания. [4] После окончания обработки задания МПД, приняв пакет от сервера, помещает результат в Модуль данных, подавая идентификатор типа данных результата (в данном примере будет возвращен набор отрезков, а тип данных будет TYPE_LINE_2D) и подает сигнал Клиентскому приложению. Клиентское приложение, получив сигнал, уже может продолжить обработку полученных данных, например, отобразить их.

3.2 Серверная часть

Сервер содержит четыре основных модуля: модуль приема/пересылки данных (МППД), модуля подготовки входящих задач (МПВЗ), модуля подготовки исходящих результатов (МПИР) и модуля управления агентами (МУА).

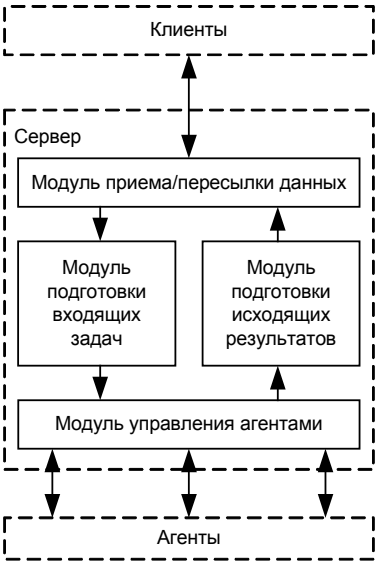

Рис. 3. Архитектура сервера

Все модули работают независимо друг от друга, что позволяет легко обновлять какую-либо компоненту сервера независимо от других компонент. Более того, это позволяет создать *распределенный* сервер – сервер, компоненты которого работают на различных машинах. Распределенные сервера бывают полезны, когда существует очень мощная вычислительная нагрузка, с которой не может справиться одна машина.[2]

Модуль Приема/Пересылки Данных (рис.4) отвечает за прием данных от клиентов и отправку полученных результатов вычислений обратно. При этом МППД проверяет каждый пришедший пакет на возможность обработки – если плагинов, способных обработать пакет, нет, модуль уведомляет клиента об этом и удаляет пакет. Чтобы операции приема и отправки данных могли выполняться независимо, МППД имеет два основных рабочих потока – один поток для приема входящих пакетов от клиентов, а другой – для отправки исходящих пакетов с результатами вычислений.

Поток приема данных, приняв пакет от клиента, помещает его в Очередь Входящих Пакетов. Для защиты от переполнения, очередь должна иметь ограничение по общему объему размещенных данных. Второй поток – поток отправки – берет исходящие данные из Очереди Исходящих Пакетов и отправляет соответствующему клиенту. Адрес клиента хранится в заголовке исходящего пакета. При отправке данных сервер не проверяет клиента на доступность, и не ожидает его ответа для избегания возможных задержек. Обе очереди взаимосвязаны – если общий объем данных в обеих очередях достигает или превышает лимит, то прием данных прекращается до тех пор, пока очередь исходящих сообщений не уменьшится так, чтобы общий объем данных стал меньше лимита.

Таким образом, очередь исходящих пакетов имеет преимущество перед очередью входящих пакетов – даже если очередь переполнена, очередь исходящих пакетов может пополняться, а очередь входящих – нет. Очередь входящих пакетов имеет фиксированный лимит общего размера данных. Очередь исходящих сообщений – гибкий, т.е. ее лимит может превышать свое первоначальное значение за счет уменьшения лимита входящих сообщений. Очередь входящих пакетов будет заблокирована для входящих сообщений до тех пор, пока общий размер обеих очередей не станет меньше лимита или размер самой очереди входящих пакетов не уменьшится

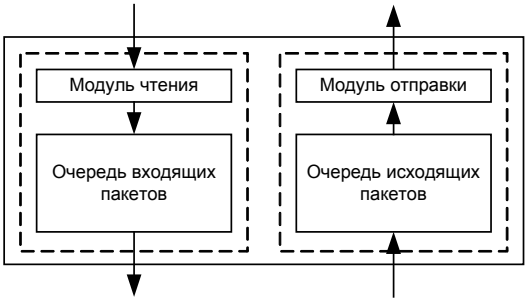

Рис. 4. Модуль приема/пересылки данных

Модуль Подготовки Входящих Задач (МПВЗ) (рис.4) отвечает за обработку входящих пакетов и преобразование в наборы подзадач. МПВЗ состоит из двух управляющих модулей (менеджера входящих пакетов и менеджера плагинов) и набора плагинов подготовки подзадач.

Подзадача – это элементарное задание. Каждая входящая задача может быть разбита на набор элементарных заданий. Например, если взять предыдущий пример удаления невидимых линий, то такими элементарными заданиями могут быть сортировка объектов сцены, вычисление невидимых граней объектов и т.д.

Менеджер Входящих Пакетов (МВП) отслеживает появление новых заданий в очереди и запрашивает у менеджера плагинов наличие незанятого плагина, способного обработать пакет. Если соответствующий свободный плагин был найден, то ему передается пакет данных, а сам пакет удаляется из очереди. Если свободного плагина, способного обработать пакет, нет, то пакет остается в очереди.

Менеджер плагинов (МП) отслеживает освобождение плагинов. Как только появляется какой-либо освободившийся плагин МП извещает Менеджера входящих пакетов. Тот, в свою очередь просматривает очередь на предмет наличия ожидающего пакета, которому необходим данный плагин. Если таких пакетов несколько, то выбирается пакет, пришедший раньше всех.

Плагин подготовки подзадач выбирает алгоритм обработки, соответствующий идентификаторам пакета, и, обработав данные, выдает набор пакетов, соответствующих элементарным заданиям, которые будут отправлены агентам. Этот набор пакетов помещается в очередь входящих подзадач, но если она переполнена, то данные остаются в плагине, и плагин ожидает освобождения очереди. Отдав свои данные, плагин вновь становится свободным. Размер очереди входящих подзадач зависит от размера очереди входящих решений (см. Модуль Подготовки Исходящих Результатов) аналогично зависимости очередей входящих и исходящих пакетов в МППД.

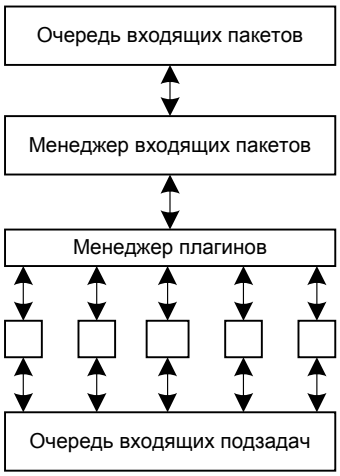

Рис. 5. Модуль Подготовки Входящих Задач

Модуль Подготовки Исходящих Результатов (Рис.5)

Данный модуль отвечает за компоновку результатов вычислений в единый пакет данных, который будет отправлен клиенту.

Аналогично МПВЗ, МПИР состоит из двух управляющих модулей и набора плагинов. В целом работа модуля аналогична вышеописанному модулю подготовки входящих задач. Отличается только работа плагинов.

Плагин компоновки решений, получив идентификатор базового алгоритма обработки, посылаемого клиентом вместе с пакетом данных, формирует результирующий пакет данных из результатов вычислений, приходящих от агентов. Когда результирующий пакет данных полностью сформирован, он помещается в очередь исходящих пакетов, откуда будет отослан обратно клиенту.

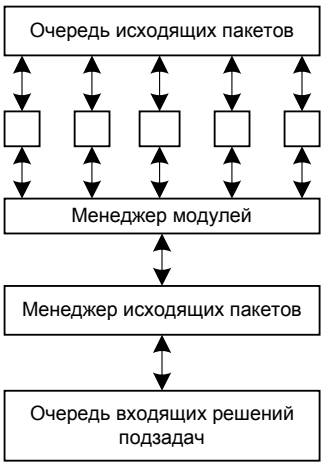

Рис. 6. Модуль Подготовки Исходящих Результатов

Модуль Управления Агентами (Рис.6) отвечает за управление агентами и распределением по ним пакетов подзадач для обработки. МУА работает с очередью входящих заданий аналогично вышеописанным модулям, т.е. для каждого нового пакета или их набора ищется соответствующий свободный агент, а для каждого освобождающегося агента, в очереди ищется пакет, который может быть им обработан.

Если свободных агентов, способных обработать набор пакетов, несколько, то количество и размер пакетов,

отсылаемых конкретным агентам, вычисляется исходя из следующего алгоритма.

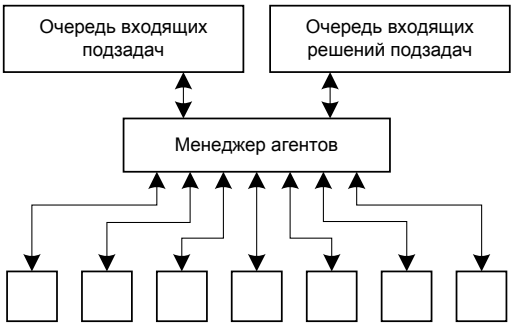

Рис. 7. Модуль Управления Агентами

Пусть имеется N свободных агентов, способных обработать данную подзадачу. Пусть имеем всего M однородных пакетов, содержащих данные одного типа и предназначенных для обработки одним типом алгоритма. Пусть $t_i = t_i^1 + t_i^2$ время обработки единицы данных i-м агентом, где t_i^1 - время передачи единицы данных от сервера агенту и обратно, а t_i^2 - время обработки единицы данных агентом.

Нам необходимо достичь следующего условия:

$$\begin{cases} t_i m_i \rightarrow \min; \\ \sum_{i=1}^{N} m_i = M; \end{cases}$$

Составим следующую систему уравнений относительно mi, предварительно отсортировав по коэффициентам t_i:

$$\begin{cases} t_1 m_1 = t_2 m_2 = \ldots = t_N m_N \\ t_1 \geq t_2 \geq \ldots \geq t_N \\ \sum_{i=1}^{N} m_i = M; \end{cases} \rightarrow \begin{cases} m_1 = M - \sum_{i=2}^{N} m_i \\ t_2 m_2 - t_1 m_1 = 0 \\ \ldots \\ t_N m_N - t_1 m_1 = 0 \end{cases}$$

Поиск значений m_i происходит следующим образом: получив ответ для $\boldsymbol{m_N}$, определяем – если ответ целочисленный, то продолжаем вычисление $\boldsymbol{m_{N-1}}$, если же дробный, то округляем до ближайшего большего целого.

Так как коэффициент $\boldsymbol{t_N}$ у $\boldsymbol{m_N}$ наименьший, это означает, что соответствующий этой паре агент имеет наибольшую скорость вычислений, по сравнению с остальными, так что мы просто увеличиваем его нагрузку.

Аналогично определяем остальные $\boldsymbol{m_i}$. Таким образом, пакеты распределяются по агентам так, чтобы увеличить нагрузку на машины с большой вычислительной мощностью и уменьшить на более медленных машинах, что позволяет уменьшить общее время ожидания всех решений.

3.3 Агентская часть

Агент имеет в своем составе модуль чтения/отправки данных, менеджер плагинов, а также один или несколько плагинов обработки подзадач.

Получив набор пакетов данных, агент определяет соответствующий плагин и обрабатывает полученные данные. Результат вычислений отправляется обратно серверу.

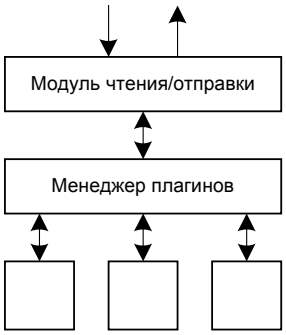

Рис. 8. Архитектура агента

4. РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ

Для расширения функциональности библиотеки, разработчику необходимо реализовать три вида плагинов: плагин подготовки подзадач, плагин компоновки решений и плагин обработки данных. Если плагин подготовки подзадач создает набор подзадач, которые могут быть обработаны уже имеющимися плагинами, то третий плагин может не реализовываться. Каждый из этих плагинов должен реализовывать соответствующий интерфейс.

Если разработчику необходимо работать с собственным форматом данных, то он должен реализовать Модуль данных. Этот модуль должен предоставлять стандартный интерфейс данных, позволяющий как читать эти данные в виде однородного массива байт, так и писать в этот модуль. Модуль должен сам упаковывать свои данные в байтовый массив и распаковывать приходящий результат.

5. ЗАКЛЮЧЕНИЕ

Предлагаемая библиотека позволяет значительно упростить разработку новых приложений параллельных и распределенных вычислений, помогая разработчику сфокусироваться на разработке и реализации алгоритмов, не отвлекаясь на организацию передачи и синхронизации данных.

Более того, так как все плагины имеют унифицированный программный интерфейс, то использование уже реализованных алгоритмов становится простым и прозрачным.

Основные возможности библиотеки:

- Возможность работы как на машине клиента, так и на другой машине
- Простота наращивания мощности
- Легкость расширения функциональности.
- Поддержка большинства основных типов данных компьютерной графики.

6. БИБЛИОГРАФИЯ

[1] Фейт С. TCP/IP: Архитектура, протоколы, реализация (включая IP версии 6 и IP Security) Изд. 2-е (пер. Кузьмина М.) - 424 с.; М: «Лори», 2004

[2] Барановская Т.П. Архитектура компьютерных систем и сетей. Барановская Т.П., Лойко В.И.,

Семенов М.И. и др., М: «Финансы и статистика» - 2003, 256 стр.

[3] Эндрюс, Г.Р. Основы многопоточного, параллельного и распределенного программирования /Пер. с англ./ – М.: Издательский дом «Вильямс», 2003. – 512 с.

[4] Microsoft Development Network – http://msdn1.microsoft.com/en-us/default.aspx

[5] Ротков С.И. «Средства геометрического моделирования и компьютерной графики пространственных объектов для CALS-технологий». Диссертация на соискание ученой степени доктора технических наук – Нижний Новгород, 1999 – 280 с.

[6] Буеракова Л.В. «Формирование и оптимизация геометрической информации об объекте для получения технического чертежа». Диссертация на соискание ученой степени кандидата технических наук – Нижний Новгород, 1991 – 190 с.

[7] Митин С.В. «Исследование и разработка методов и средств визуализации трехмерных объектов». Диссертация на соискание ученой степени кандидата технических наук – Нижний Новгород, 1994 – 144 с.

Об авторах

Ротков Сергей Игоревич– профессор, доктор технических наук, заведующий кафедрой начертательной геометрии, машинной графики и САПР Нижегородского государственного архитектурно-строительного университета.

Адрес: Нижегородский государственный архитектурно-строительного университет, 603950, г. Нижний Новгород, ул. Ильинская, д.65.

Телефон: 34-10-34

e-mail: rotkov@nngasu.ru

Лазарев Павел Юрьевич – аспирант кафедры начертательной геометрии, машинной графики и САПР Нижегородского государственного архитектурно-строительного университета.

Адрес: Нижегородский государственный архитектурно-строительного университет, 603950, г. Нижний Новгород, ул. Ильинская, д.65.

e-mail: pavellazarev@yandex.ru

Architecture of the library of parallel and distributed geometric calculations with scalable functionality

Abstract

The paper is devoted to architecture of the library of parallel and distributed geometric calculations with scalable functionality

The paper contains information about library functionality enhancement and existing functionality using.

Keywords*: programming of a parallel applications, programming of a distributed applications, client, server, plug-in*

About the authors

Sergey Rotkov is Dr, prof., a chief of Descriptive Geometry and CAD/CAM Graphical Systems Department. His contact e-mail is rotkov@nngasu.ru

Pavel Lazarev is a Ph.D. student at Descriptive Geometry and CAD/CAM Graphical Systems Department. His contact e-mail is pavellazarev@yandex.ru