Многопоточное интегрирование функций одной переменной методом Монте-Карло.

C++ Developer. Professional

Меня хорошо видно & слышно?

Защита проекта
Тема:Многопоточное интегрирование функций одной переменной методом Монте-Карло.

Дмитрий Ронжин

Математик и разработчик
https://ronzhin-dmitry.github.io/

План защиты

- «Легенда»
- Цель и задачи проекта
- Какие технологии использовались
- Что получилось
- Выводы
- Вопросы и рекомендации

«Легенда»

В наш замечательный НИИЧАВО завезли новый сервер с большим числом вычислительных ядер. Все сотрудники в восторге! Каждый мечтает на этом замечательном сервере интегрировать все подряд. Заодно пользователи просят добавить возможность использовать сервер в качестве онлайн калькулятора, поскольку бывает удобно в процессе работы отправлять вычисление значения функции в некоторой точке на удаленную машину.

Мудрое начальство поручило создать серверное ПО для обработки входящих TCP соединений от пользователей, содержащих запросы на вычисление, а также асинхронную обработку этих запросов.

Главная цель – возможность одновременно запускать много запросов на вычисление и возвращать пользователям ответ в виде строки (строка может быть либо результатом вычисления, либо сообщением об ошибке).

Цель и задачи проекта

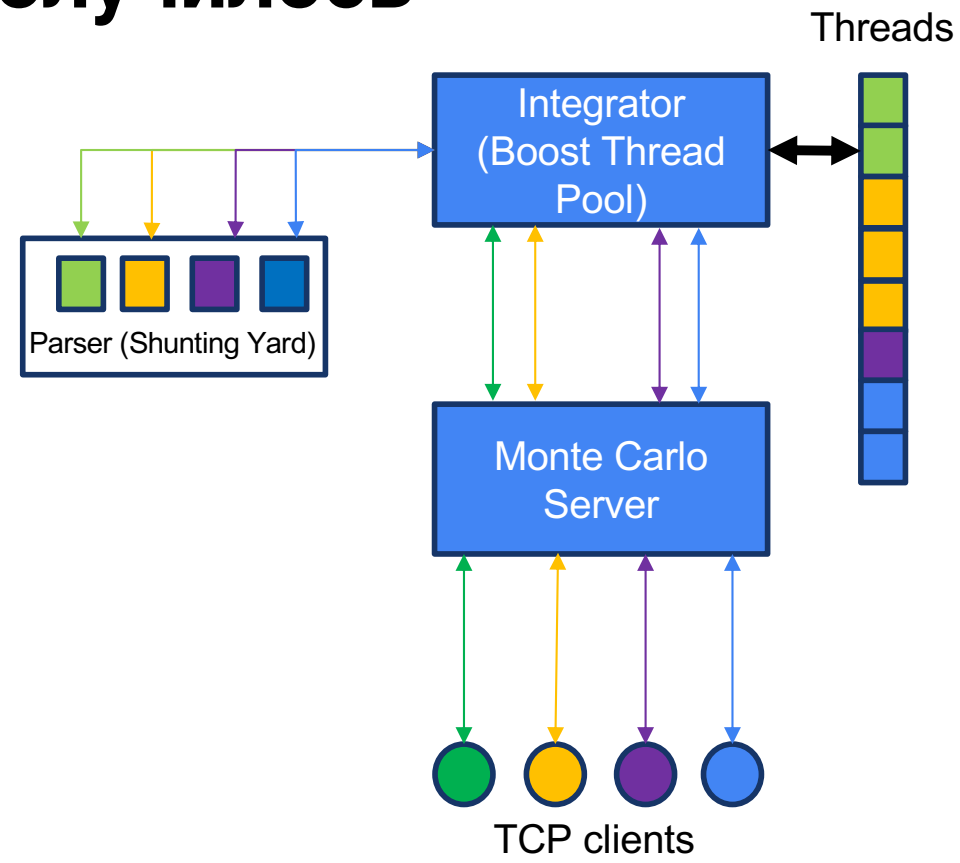
Цель проекта: создать библиотеку для многопоточного численного интегрирования методом Монте-Карло. Библиотека должна поддерживать возможность асинхронной обработки входящих запросов через TCP.

1. Разработать упрощенный модуль синтаксического анализа (Parser) для обработки текстового представления функций одной переменной;
2. Разработать модуль многопоточного интегрирования по заданному числу точек испытаний либо ожидаемой дисперсии ответа; модуль должен поддерживать вычисление собственных интегралов и сходящихся несобственных интегралов с одним бесконечным пределом (верхняя или нижняя граница интегрирования);
3. Разработать модуль асинхронной обработки входящих TCP соединений от пользователей и ответа на запросы клиентов. Пользователи могут отправлять запросы как на интегрирование, так и на вычисление значения функции в некоторой точке;
4. Разработать тесты для всех базовых модулей, продемонстрировать работоспособность приложения.

Какие технологии использовались

1. Boost Asio для асинхронной обработки входных соединений
2. Boost Thread Pool для многопоточного интегрирования методом Монте-Карло
3. Boost Test для покрытия базового функционала тестами
4. Немного алгоритмов и математики (Shunting Yard для синтаксического анализа, стандартная замена переменной для бесконечных границ интегрирования).

Что получилось



Для каждой сессии свой парсер.

Единый «интегратор» – точка входа для запуска вычислительных задач.

Каждая сессия (клиент) может определять необходимое число нитей для исполнения (динамическая балансировка нагрузки не предусмотрена).

Live demo 

Выводы

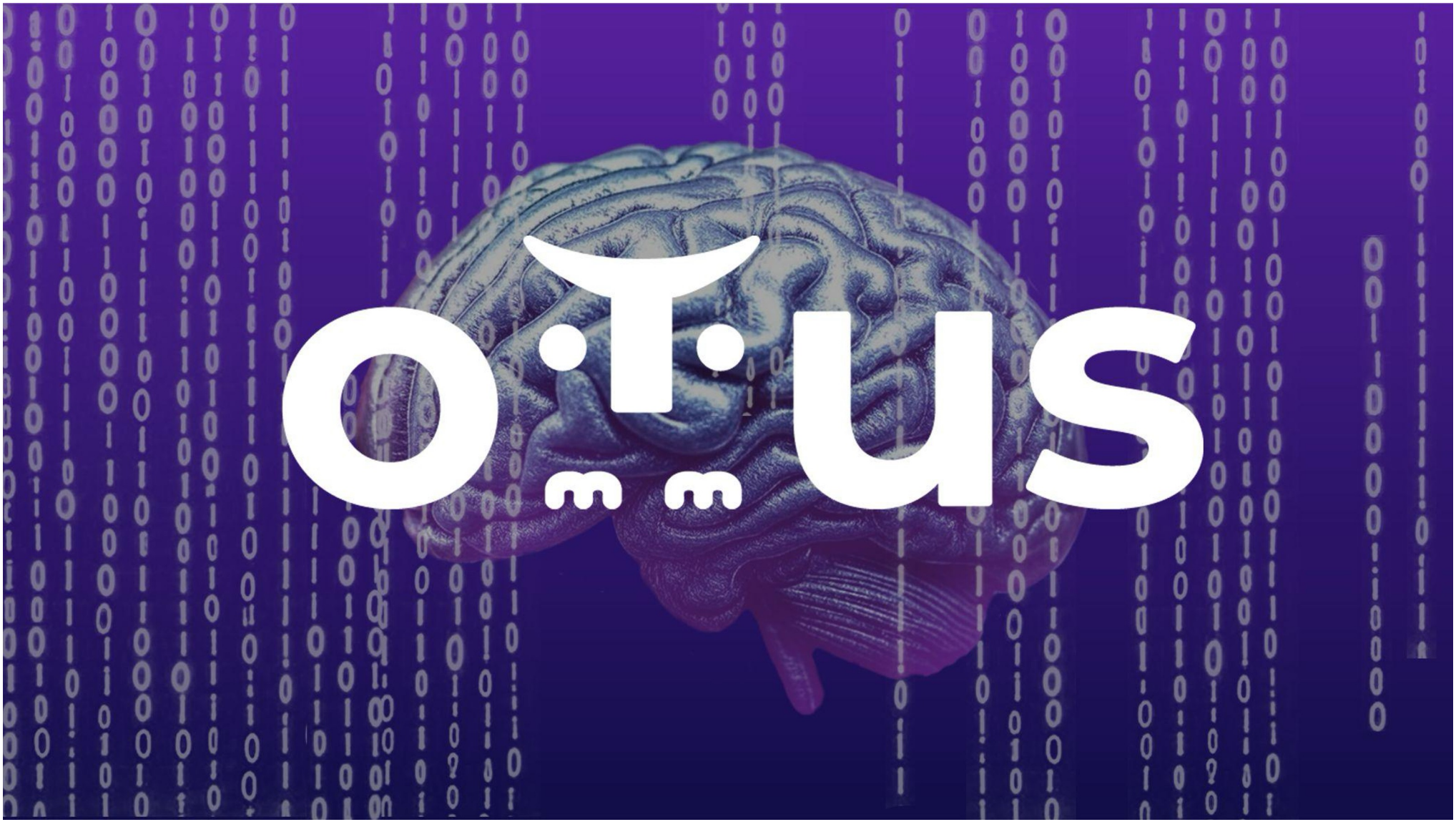
1. Boost thread pool вполне удобен, но нужно быть аккуратным с join. В данном проекте для синхронизации одного «пула» на вычислительную задачу использованы атомарные переменные, в рамках всего thread pool;
2. Получилось реализовать рабочую серверную часть, для клиентской части нужно делать отдельное приложение с поддержкой установленного API;
3. Полезно сразу написать пару тестов чтобы потом не наступать дважды на одни и те же грабли.

Вопросы и рекомендации

\+ если есть вопросы

— если вопросов нет

Спасибо за внимание!


OTIUS