Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

**Сетевая приложение «Многопоточная сортировка слиянием»**

Курсовая работа

по дисциплине:

«Операционные системы»

Разработал:

Студент группы 9091

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_».\_\_\_\_\_.2021г.

Проверил:

Ананьев В.В.\_\_\_\_\_\_

«\_\_».\_\_\_\_\_.2021г.

**Великий Новгород**

**2021**

ВВЕДЕНИЕ

**Цель.**

Предоставить пользователю средства для очень быстрой сортировки массива без вычислений на компьютере пользователя для экономии ресурсов.

**Решение.**

Создание программы, запущенной на сервере, которая будет принимать входящие значения от подключённых клиентов, обрабатывать и отправлять обратно. В качестве сортировки, для высокого быстродействия, выбрана многопоточная сортировка слиянием.

**Перечень основных функций, предъявляемых к разрабатываемому приложению**:

1. Подключение нескольких клиентов по протоколу TCP/IP
2. Обеспечение высокой производительности сортировки
3. Сортировка должна выполняться многопоточно
4. Сортировка должна выдавать верные результаты
5. Обеспечение высокой надёжности работы

ГЛАВА 1. РАЗРАБОТКА ПРОЕКТА.

1.1 Постановка задачи.

Задачей данной курсовой работы является разработка сетевого приложения «Многопоточная сортировка слиянием», может обрабатывать сразу несколько клиентов. Приложение должно работать под управлением операционной системы семейства Unix/Linux. Сетевая часть приложения должна быть выполнена на языке C++ с помощью сокетов предоставляемых системной библиотекой <sys/socket.h>. Сортировка производится алгоритмом “сортировка слиянием” в многопоточном режиме.

1.2 Выбор инструментальных средств.

Основным дистрибутивом линукс, на котором будет работать игра выбран Ubuntu 24.04 на архитектуре i386, так как только этот дистрибутив установлен на учебных компьютерах в НовГУ.

Сетевая часть программы будет реализована с помощью библиотек сокетов Беркли на C++. Протокол передачи данных: TCP.

1.3 Выбор модели.

Модель выглядит следующим образом: клиент должен подключиться к серверу, отправить необходимый массив для сортировки, сервер обрабатывает запрос, сортируя массив, и отправляет клиенту обратно. При этом клиент может запрашивать сортировки массива параллельно, пока сортируется другой (если, например запрос состоит из очень большого количества чисел)

1.4 Взаимодействие клиента и сервера

Как уже было сказано, клиент и сервер обмениваются информацией по протоколу TCP/IP. Передача массива от клиента к серверу происходит в два этапа. Сначала серверу отсылается размер массива для того, чтобы сервер выделил память под массив на своей машине. После чего клиент отсылает сам массив указанного размера. Сервер выполняет сортировку и отправляет сортированный массив клиенту. Передача массива происходит с помощью передачи указателя.

1.5 Вывод по главе 1.

Таким образом, для реализации приложения “Многопоточная сортировка слиянием” основной ОС выбрана Ubuntu, языком программирования C++. Определены средства передачи информации, модель приложения, взаимодействие сервера и клиента.

ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА.

**2.1 Реализация сервера**

Для успешного запуска сервера ему необходим аргумент командной строки, который будет обозначать порт, на котором будет работать сервер. Программа пробует создать сокет с помощью системного вызова socket(), в случае неудачи выводит сообщение и завершает работу. Затем, в функции create\_connection() по полученному сокету осуществляется создание подключения – так же с проверкой на неудачу. Неудача обычно может возникнуть, если порт уже занят. Дальше программа разделяется на два потока – поток обработки подключения клиентов (client\_connect\_handle) и поток обработки серверных команд (handle\_commands\_thr), подключаясь к последней. Сервер имеет несколько команд: -disconnect, которая закрывает соединение с предложенным клиентом, -exit для выхода, -help для вывода помощи. Поток подключения работает так – имеем бесконечный цикл, в котором с помощью блокирующей функции accept() будет принимать входящих клиентов. Для каждого подключённого клиента создаём новый запрос обработки запросов клиента. Поток обработки запросов состоит в следующем – имеем бесконечный цикл, вызываем блокирующую функцию recv() с получением размера массива от клиента. Когда сервер получит эту информацию, код пойдёт дальше. С помощью recv() получаем от клиента уже сам массив нужного размера. Вызываем функцию сортировки merge\_sort() модуля Sorter.cpp, до и после которой запоминаем время программы с помощью системной функции clock(). Далее отправляем отсортированный массив клиенту.

**2.2 Реализация сортировки**

Рассмотрим файл Sorter.cpp. Основная функция вызова сортировки – merge\_sort(), которая в качестве аргументов принимает указатель (массив) double для сортировки и размер данного массива. Данная функция вызывает функцию merge\_sort\_mt, в качестве аргументов передаёт массив, размер, и 4 – число потоков для обработки сортировки. Число 4 выбрано, потому что слишком большое число потоков будет трудно-контролируемым, а малое число не приведёт к существенному приросту производительности. Функция merge\_sort\_mt должна разделять существующий массив на 2 части и вызывать саму себя в новом потоке, если размер подмассива существенен. Разделение происходит с помощью вызова той же функции merge\_sort\_mt, перемещая указатель массива на начало массива и середину массива. Для проверки размера подмассива существуют условия – если переданный размер < 2, то закончить работу, если меньше 4, или глубина (потоков не осталось) равна нулю, то разделяем массив и заканчиваем разделение, а если размер 4 и больше, то начинаем разделение в новом потоке. Рассмотрим детальнее что происходит если размер 4 и более. Код имеет структуру Params, которая имеет данные – double для хранения массива, size\_t для размера массива, int для хранения глубины (кол-во потоков). Данная структура нужна для передачи потоку в качестве аргумента, так как поток принимает единственный аргумент void\*. Как раз в случае если размер 4 и более, то создаётся переменная типа Params в которой указаны аргументы для разделения левой части массива и это будет выполнено в новом потоке с помощью pthread\_create(…,…, merge\_sort\_thread, &params). Разделение правой части вызовется в текущем потоке. Так же здесь присутствуют функции pthread\_mutex\_lock и pthread\_mutex\_unlock, после вызова, который осуществляется вывод информации в консоль – мьютексы необходимы для разблокировки потока и вывода информации на экран. При любом условии размера массива, кроме случая размера меньше 2 в конце функции вызывается функция merge – функция слияния. Таким образом, так как функция merge\_sort\_mt вызывается рекурсивно, сначала исходный массив разделится на части размера около 2, и эти маленькие подмассивы будут слиты. Затем слитые массивы сольются с соответствующий подмассивом побольше и так пока не получим отсортированный массив исходного размера. Рассмотрим функцию merge. Функция выделяет указатель на новый массив – в нём будем заполнять слитый подмассив. Слитие происходит следующим образом – инициализируем указатели lhs начала подмассива и середины rhs подмассива. Сравниваем – если lhs < rhs по значению, то кладём в новый массив lhs, сдвигаем lhs на 1 вперёд, сдвигаем указатель на новый массив на 1 вперёд. Иначе всё то же самое проделывается с rhs. Этот цикл происходит до тех пор, пока lhs не достигнет середины, либо rhs не достигнет конца. В случае если rhs достиг конца раньше, то заполняем массив оставшимися lhs, так как они гарантированно будут больше всех (так как остальные элементы мы все сравнили). В случае если rhs не достиг конца мы ничего не делаем, так как эти элементы лежат справа (а чем правее, тем больше элементы). После этого копируем новый массив в основной, удаляем этот переменный массив.

**2.3 Реализация клиента**

Клиент принимает в качестве аргумента командной строки строку вида ip:port, где ip – адрес подключения, port – порт, на котором работает сервер. Программа разбивает эту строку на подстроки ip и port, приводит их в численный вид, создаёт свой сокет с помощью системного вызова socket(), и подключается к серверу по полученному из ip хосту и порту. Далее создаётся поток обработки клиентских команд handle\_commands\_thr, и подключается к нему.

Клиент так же получает команды из командной строки и обрабатывает в цикле while (true). Клиент имеет следующие команды. Команда -file \*filename\* открывает файл, читает из него числа типа double и отправляет на сервер в случае успешного прочтения. Функция открытия файла file\_to\_numbers пробует открыть файл с помощью fopen() в режиме чтения, если файл открылся вычисляет его размер с помощью перемещения по файлу до конца (fseek), получения размера (ftell), перемещения обратно (fseek), по вычисленному размеру прочитывает файл, убирает лишние ‘\n’, вызывает функция string\_to\_numbers, которая по строке возвращает массив. Команда -rand \*count\* \*min\* \*max\* переводит текстовые аргументы в числовые, создаёт массив по размеру count, и заполняет его случайными числами с помощью системной функции rand(), в передлах min и max (min + rand() % max). Функция -nums nums1|nums2|nums3 переводит текстовые числа nums1,2,3… в численные с помощью функции string\_to\_numbers и отправляет их на сервер. Команда -exit завершает программу, команда -help выводит помощь.

Отправление массива на сервер и его получение происходит в отдельном потоке, что позволяет отправлять на сортировку несколько массивов одновременно (нужно в случае, если массивы очень большие). Передача массива и размера массива в поток осуществляется с помощью структуры ArrayParams. Поток отправляет на сервер размер массива, сам массив, ожидает массив от сервера, выводит отсортированный массив.

**Заключение.**

Таким образом, в рамках образовательного курса “Операционные системы” мной было создано клиент-серверное приложение “Многопоточная сортировка слиянием”.

Данное приложение выполняет поставленный перед ним функционал, имеет хорошее быстродействие, подключение реализовано с помощью протокола TCP/IP и использованием системных функций. Приложение выполнено на языке C++.