Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

|  |  |  |  |
| --- | --- | --- | --- |
| Институт: | *ИВТИ* | Кафедра: | *МКМ* |
| Направление подготовки: | | *01.03.02 Прикладная математика и информатика* | |

**ОТЧЕТ по практике**

|  |  |
| --- | --- |
| **Наименование практики:** | Производственная практика: технологическая (проектно-технологическая) практика |

**СТУДЕНТ**

|  |  |
| --- | --- |
| photo_2022-07-10_01-24-23 | / Кузьмин Ю.С. / |
| *(подпись )* | (*Фамилия и инициалы*) |

|  |  |
| --- | --- |
| Группа | А-14 -19 |
|  | *(номер учебной группы)* |

**ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ПРАКТИКЕ**

|  |
| --- |
|  |
| *(отлично, хорошо, удовлетворительно, неудовлетворительно)* |

|  |  |
| --- | --- |
|  | / Шевченко И.В. / |
| *(подпись )* | (*Фамилия и инициалы члена комиссии*) |

|  |  |
| --- | --- |
|  | / / |
| *(подпись )* | (*Фамилия и инициалы члена комиссии*) |

**Москва**

**2022**

ГРАФИК прохождения ПРАКТИКИ

|  |  |  |
| --- | --- | --- |
| Номер  п/п | Перечень работ в соответствии с заданием на практику | Отметка о выполнении работы  (выполнено /не выполнено) |
| 1 | Получение задания от руководителя. Установка и настройка на компьютере необходимых инструментальных средств. | выполнено |
| 2 | Изучение контроля версий git и создание удалённого репозитория на сайте github.com | выполнено |
| 3 | Изучение технологии распараллеливания OpenMP | выполнено |
| 4 | Изучение библиотеки GMP | выполнено |
| 5 | Реализация библиотеки обёртки над GMP, реализующей арифметические операции над массивами длинных чисел, с заданной степенью распараллеливания | выполнено |
| 6 | Проведение оценки распараллеливания. | выполнено |
| 7 | Оформление отчёта по практике. | выполнено |

|  |  |  |
| --- | --- | --- |
| Руководитель практики (от МЭИ) |  | / Шевченко И.В. / |
|  | *(подпись )* | (*Фамилия и инициалы*) |

## Результаты выполнения индивидуального задания

### Постановка задачи

Реализация библиотеки обёртки, реализующей арифметические операции над массивами длинных(целых) чисел, с заданной степенью распараллеливания. В процессе работы необходимо коммитить все наработки в репозиторий системы контроля версий.

### Описание объекта задания

**GNU Multiple Precision Arithmetic Library (GMP)** — бесплатная библиотека для арифметических операций произвольной точности, работающая с целыми числами со знаком, рациональными числами и числами с плавающей запятой. Нет никаких практических ограничений точности, кроме тех, которые подразумеваются доступной памятью (операнды могут иметь размер до 2^32-1 бит на 32-битных машинах и 2^37 бит на 64-битных машинах). GMP имеет богатый набор функций, а функции имеют обычный интерфейс. Базовый интерфейс предназначен для C, но существуют оболочки для других языков, включая Ada, C++, C#, Julia, .NET, OCaml, Perl, PHP, Python, R, Ruby и Rust.

Основными целевыми приложениями GMP являются криптографические приложения и исследования, приложения для обеспечения безопасности в Интернете и системы компьютерной алгебры.

GMP стремится быть быстрее любой другой библиотеки bignum для всех размеров операндов. Некоторые важные факторы для этого:

* Использование полных слов в качестве основного арифметического типа.
* Использование разных алгоритмов для разных размеров операндов; алгоритмы, которые работают быстрее для очень больших чисел, обычно медленнее для малых чисел.
* Высокооптимизированный код на языке ассемблера для наиболее важных внутренних циклов, специализированный для разных процессоров.

GMP используется для целочисленной арифметики во многих системах компьютерной алгебры, таких как Mathematica и Maple. Он также используется в библиотеке алгоритмов вычислительной геометрии (CGAL).

**Git** — это программное обеспечение для отслеживания изменений в любом наборе файлов, обычно используемое для координации работы программистов, совместно разрабатывающих исходный код во время разработки программного обеспечения. Его цели включают скорость, целостность данных и поддержку распределенных нелинейных рабочих процессов (тысячи параллельных ветвей, работающих в разных системах).

**OpenMP (Open Multi-Processing)** — открытый стандарт для распараллеливания программ на языках Си, Си++ и Фортран. Даёт описание совокупности директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью.

OpenMP реализует параллельные вычисления с помощью многопоточности, в которой ведущий (англ. master) поток создаёт набор ведомых потоков, и задача распределяется между ними. Предполагается, что потоки выполняются параллельно на машине с несколькими процессорами (количество процессоров не обязательно должно быть больше или равно количеству потоков).

Задачи, выполняемые потоками параллельно, так же, как и данные, требуемые для выполнения этих задач, описываются с помощью специальных директив препроцессора соответствующего языка — «прагм». Например, участок кода на Фортране, который должен исполняться несколькими потоками, каждый из которых имеет свою копию переменной N, предваряется следующей директивой: !$OMP PARALLEL PRIVATE(N)

Количество создаваемых потоков может регулироваться как самой программой при помощи вызова библиотечных процедур, так и извне, при помощи переменных окружения.

Ключевые элементы стандарта:

* конструкции для создания потоков (директива parallel),
* конструкции распределения работы между потоками (директивы DO/for и section),
* конструкции для управления работой с данными (выражения shared и private для определения класса памяти переменных),
* конструкции для синхронизации потоков (директивы critical, atomic и barrier),
* процедуры библиотеки поддержки времени выполнения (например, omp\_get\_thread\_num),
* переменные окружения (например, OMP\_NUM\_THREADS).

### Описание метода решения задачи

Была создана библиотека, в которой реализован класс gmpArray. В нём определены поля n, хранящий количество элементов массива, и values, хранящий значения элементов. Также реализован оператор [], позволяющий обратиться к необходимому элементу по номеру, и методы sumOMP и mulOMP, реализующие сумму и умножение массивов поэлементно, используя многопоточность, с заданной степенью распараллеливания.

### Экспериментальная часть

Были созданы массивы с количеством элементов 10 000, 25 000, 50 000 и 100 000. Для каждого размера 75 раз было проведены поэлементные суммирование и умножение и найдено среднее время работы для каждого эксперимента. Количество потоков при этом равно от 1 до 4 для каждого вычислительного эксперимента. Ниже приведено полученное время работы.

Number of threads= 1, arrays' size= 10000, average time of sumOMP= 1.133333ms

Number of threads= 2, arrays' size= 10000, average time of sumOMP= 0.226667ms

Number of threads= 3, arrays' size= 10000, average time of sumOMP= 0.120000ms

Number of threads= 4, arrays' size= 10000, average time of sumOMP= 0.093333ms

Number of threads= 1, arrays' size= 10000, average time of mulOMP= 139.306667ms

Number of threads= 2, arrays' size= 10000, average time of mulOMP= 77.266667ms

Number of threads= 3, arrays' size= 10000, average time of mulOMP= 62.066667ms

Number of threads= 4, arrays' size= 10000, average time of mulOMP= 54.133333ms

Number of threads= 1, arrays' size= 25000, average time of sumOMP= 7.120000ms

Number of threads= 2, arrays' size= 25000, average time of sumOMP= 2.066667ms

Number of threads= 3, arrays' size= 25000, average time of sumOMP= 1.506667ms

Number of threads= 4, arrays' size= 25000, average time of sumOMP= 1.093333ms

Number of threads= 1, arrays' size= 25000, average time of mulOMP= 356.320000ms

Number of threads= 2, arrays' size= 25000, average time of mulOMP= 199.280000ms

Number of threads= 3, arrays' size= 25000, average time of mulOMP= 172.866667ms

Number of threads= 4, arrays' size= 25000, average time of mulOMP= 139.053333ms

Number of threads= 1, arrays' size= 50000, average time of sumOMP= 8.426667ms

Number of threads= 2, arrays' size= 50000, average time of sumOMP= 5.920000ms

Number of threads= 3, arrays' size= 50000, average time of sumOMP= 4.306667ms

Number of threads= 4, arrays' size= 50000, average time of sumOMP= 4.693333ms

Number of threads= 1, arrays' size= 50000, average time of mulOMP= 723.626667ms

Number of threads= 2, arrays' size= 50000, average time of mulOMP= 431.120000ms

Number of threads= 3, arrays' size= 50000, average time of mulOMP= 328.053333ms

Number of threads= 4, arrays' size= 50000, average time of mulOMP= 273.600000ms

Number of threads= 1, arrays' size= 100000, average time of sumOMP= 20.200000ms

Number of threads= 2, arrays' size= 100000, average time of sumOMP= 12.306667ms

Number of threads= 3, arrays' size= 100000, average time of sumOMP= 9.546667ms

Number of threads= 4, arrays' size= 100000, average time of sumOMP= 8.493333ms

Number of threads= 1, arrays' size= 100000, average time of mulOMP= 1722.413333ms

Number of threads= 2, arrays' size= 100000, average time of mulOMP= 1019.853333ms

Number of threads= 3, arrays' size= 100000, average time of mulOMP= 666.266667ms

Number of threads= 4, arrays' size= 100000, average time of mulOMP= 602.093333ms

### Вывод

При использовании библиотек GMP, которая даёт возможность работы с числами большой разрядности, и OpenMP, которая даёт возможность распараллеливания программы, была разработана библиотека обёртка, которая позволяет производить арифметические операции над массивами длинных целых чисел с заданной степенью распараллеливания. Сервис git позволил контролировать версии программы.

По полученным экспериментальным данным, можно сделать вывод, что увеличение количества потоков даёт заметный прирост в скорости работы программы, но чем количество потоков больше, тем прирост в скорости работы меньше.

### Список литературы

Документация OpenMP <https://www.openmp.org/specifications/>

Документация GMP <https://mpir.org/mpir-1.3.1.pdf>

Документация Git https://git-scm.com/doc