ФГБОУ ВПО «Национальный исследовательский университет «МЭИ»

Институт информатики и вычислительной техники

Кафедра прикладной математики и искусственного интеллекта

**Отчет по комплекту лабораторных работ**

по курсу

«Параллельное программирование и параллельные системы»

**Выполнила:**

Кравченко К.С.

Группа: А-05м-21

**Проверил:**

Кутепов В.П.

Москва, 2022

1. **Постановка задачи**

Требуется найти решение системы m линейных уравнений, которая записывается в общем виде как

,

Эту систему уравнений можно записать также в матричном виде:

где , , .

A – матрица системы, – вектор правых частей, – вектор неизвестных.

При известных A и требуется найти такие , при подстановке которых в систему уравнений она превращается в тождество.

1. **Описание метода.**

Для решения поставленной задачи будем использовать метод верхней релаксации.

Приведем систему к виду

Обозначим

В векторно-матричном виде система запишется в виде:

*x=Bx+c*

Для метода верхней релаксации введем параметр ω > 1. Тогда итерационная формула метода верхней релаксации будет иметь следующий вид:

1. **Описание инструментария**

**Лабораторная работа № 1. Императивный язык.**

Для выполнения первой лабораторной работы был выбран язык C#. Для решения поставленной задачи использовались методы Parallel.For и Parallel.ForEach библиотеки System.Threading.Tasks.

**Лабораторная работа № 2. Императивный язык с использованием программного интерфейса MPI.**

MPI – это программный инструментарий для обеспечения связи между ветвями параллельного приложения. MPI предоставляет программисту единый механизм взаимодействия ветвей внутри параллельного приложения независимо от машинной архитектуры (однопроцессорные / многопроцессорные с общей/раздельной памятью), взаимного расположения ветвей (на одном процессоре / на разных). Минимально в состав MPI входят: библиотека программирования и загрузчик приложений.

Базовым механизмом связи между MPI процессами является передача и приём сообщений. Сообщение несёт в себе передаваемые данные и информацию, позволяющую принимающей стороне осуществлять их выборочный приём:

● отправитель — ранг (номер в группе) отправителя сообщения;

● получатель — ранг получателя;

● признак — может использоваться для разделения различных видов сообщений;

● коммуникатор — код группы процессов.

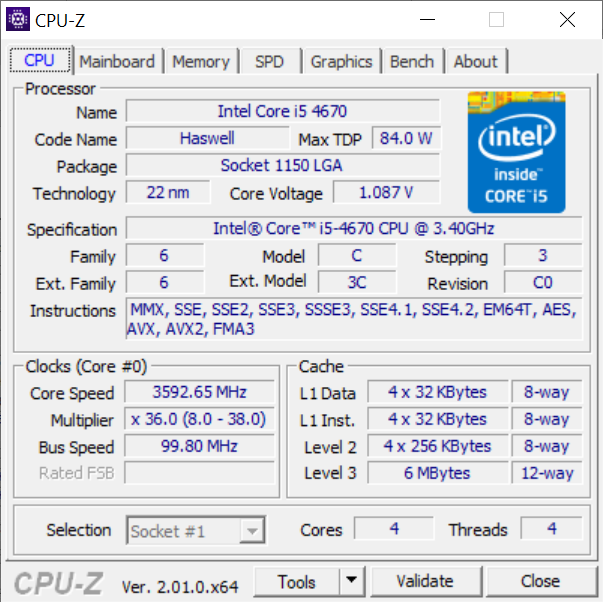
Операции приёма и передачи могут быть блокирующимися и неблокирующимися. Для неблокирующихся операций определены функции проверки готовности и ожидания выполнения операции.

Другим способом связи является удалённый доступ к памяти (RMA), позволяющий читать и изменять область памяти удалённого процесса. Локальный процесс может переносить область памяти удалённого процесса (внутри указанного процессами окна) в свою память и обратно, а также комбинировать данные, передаваемые в удалённый процесс с имеющимися в его памяти данными (например, путём суммирования). Все операции удалённого доступа к памяти не блокирующиеся, однако, до и после их выполнения необходимо вызывать блокирующиеся функции синхронизации.

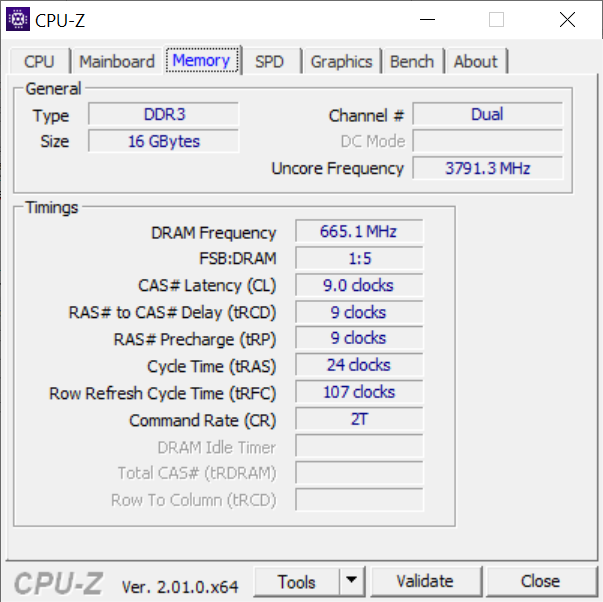
При выполнении второй лабораторной работы была использована реализация MPI для языка Python.

1. **Характеристики компьютера**

**Процессор:**



**Оперативная память:**



**Операционная система:**

Название: Windows 10 Pro

Разрядность: 64

Версия: 21H2

Сборка: 19044.1466

1. **Описание набора экспериментальных данных.**

С целью формирования симметричной положительно определенной матрицы элементы ниже главной диагонали генерировались в диапазоне от 0 до 1, значения элементов выше главной диагонали получались симметрией элементов ниже главной диагонали, а элементы на главной диагонали генерировались в диапазоне от n до 2n, где n – размер матрицы.

В качестве набора данных использовались:

* СЛАУ с 1000 неизвестными и 1000 уравнениями.
* СЛАУ с 5000 неизвестными и 5000 уравнениями.

1. **Результаты экспериментов**

**СЛАУ с 1000 неизвестными и 1000 уравнениями.**

Императивный язык:

Последовательное выполнение:

* время: 00:00:00.06
* количество тиков: 648030

Параллельное выполнение:

* время: 00:00:00.06
* количество тиков: 1357808

Количество проделанных итераций в обоих случаях одинаково и равно 7.

Как видно из результатов, параллельная реализация не дала выигрыша во времени. Это произошло, т.к. накладные расходы на создание, синхронизацию и передачу задачи данных в потоки вызывают большие потери производительности, чем выигрыш от распараллеливания малого количества простых операций (в методе верхней релаксации мы можем распараллелить только вычисление сумм).

MPI:

Последовательное выполнение:

количество тиков: 75039991

Параллельное выполнение:

количество тиков: 45589786

Ускорение: 1,65

**СЛАУ с 5000 неизвестными и 5000 уравнениями.**

Императивный язык:

Последовательное выполнение:

* время: 00:00:01.54
* количество тиков: 15495845

Параллельное выполнение:

* время: 00:00:01.54
* количество тиков: 12264679

Количество проделанных итераций в обоих случаях одинаково и равно 6.

Как видно из результатов, в случае с большей размерностью исходных данных ускорение составило 1,26.

MPI:

Последовательное выполнение:

количество тиков: 1702727793

Параллельное выполнение:

количество тиков: 1058146489

Ускорение: 1,61

В отличие от реализации на императивном языке, реализация с MPI затрачивает большее время на выполнение программы, однако она дает ускорение даже на исходных данных малых размеров.