МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тверской государственный технический университет»

(ТвГТУ)

Кафедра “Программного обеспечения”

**Курсовая работа**

по дисциплине «Параллельные вычисления»

Тема: «Сравнение средств языков С++, C# и Python для обеспечения параллельных вычислений»

Выполнили: студенты группы

ПИН 17.06

Иванов Р.В  
Чугунов А.А  
Завгороднев Е.Ю

Середавкин А.С

Проверил:

Биллиг В.А

Тверь 2021

# Введение

В данной работе производится сравнение трех инструментов для параллельного программирования: библиотека OpenMP - C++, System.Threading.Tasks – C#, модуль multiprocessing – Python. В качестве алгоритмов для распараллеливания выступают алгоритмы: вычисления интегралов, сортировки пузырьком, быстрой сортировки, быстрой рекурсивной сортировки, решения СЛАУ методом Гаусса. В качестве параметра для сравнения выступает время выполнения алгоритмов на различных данных. В ходе данной работы выяснится, какой инструмент реализует параллельные вычисления наилучшим образом.

# Алгоритмы

## Вычисление интеграла

## Для распараллеливания используется стратегия сегментного алгоритма вычисления суммы. Интервал интегрирования разбивается на k сегментов, на каждом из которых для вычисления интеграла используется обычный последовательный алгоритм. Вычисление интегралов на отдельных сегментах ведется параллельно.

В таблицу заносится среднее значение времени 5ти вычислений интеграла.

|  |  |  |  |
| --- | --- | --- | --- |
| Интеграл – вычисление числа пи | | | |
| Кол-во секторов | Точность вычисления eps | Кол-во потоков | С++ |
| 100 | 1.00E-16 | 1 | 0.71 |
| 100 | 1.00E-16 | 2 | 0.49 |
| 100 | 1.00E-16 | 3 | 0.39 |
| 100 | 1.00E-16 | 4 | 0.31 |
| 100 | 1.00E-16 | 5 | 0.27 |
| 100 | 1.00E-16 | optimal | 0.13 |

|  |  |  |  |
| --- | --- | --- | --- |
| Интеграл – вычисление числа пи | | | |
| Кол-во секторов | Точность вычисления eps | MaxDegreeOfParallelism | С# |
| 100 | 1.00E-16 | 1 | 0.32 |
| 100 | 1.00E-16 | 2 | 0.2 |
| 100 | 1.00E-16 | 3 | 0.15 |
| 100 | 1.00E-16 | 4 | 0.12 |
| 100 | 1.00E-16 | 5 | 0.11 |
| 100 | 1.00E-16 | optimal | 0.1 |

В среднем C# версия алгоритма оказалась быстрее C++ версии в два раза из за оптимизаций JIT-компилятора, который оптимизирует часто исполняемые участки кода.

На одном вычислении интеграла можно увидеть следующие результаты:

<<Рома блять вставь результаты>>

## Решение СЛАУ методом Гаусса

## Сортировка пузырьком

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются {\displaystyle N-1}N - 1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма).

|  |  |  |  |
| --- | --- | --- | --- |
| Сортировка пузырьком | | | |
| Кол-во секторов | Точность вычисления eps | Кол-во потоков | C++ |
| 200 | 1.00E+06 | 1 | 15.01 |
| 200 | 1.00E+06 | 2 | 8.03 |
| 200 | 1.00E+06 | 3 | 5.55 |
| 200 | 1.00E+06 | 4 | 4.42 |
| 200 | 1.00E+06 | 5 | 3.83 |
| 200 | 1.00E+06 | optimal | 3.01 |

## Быстрая сортировка

|  |  |  |  |
| --- | --- | --- | --- |
| Быстрая сортировка | | | |
| Кол-во секторов | Точность вычисления eps | Кол-во потоков | С++ |
| 1000 | 1.00E+07 | 1 | 3.75 |
| 1000 | 1.00E+07 | 2 | 3.16 |
| 1000 | 1.00E+07 | 3 | 2.97 |
| 1000 | 1.00E+07 | 4 | 2.89 |
| 1000 | 1.00E+07 | 5 | 2.84 |
| 1000 | 1.00E+07 | optimal | 2.78 |

## Быстрая сортировка(рекурсия)

Общая идея алгоритма состоит в следующем:

* Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность (см. ниже).
* Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».
* Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

|  |  |  |  |
| --- | --- | --- | --- |
| Быстрая сортировка | | | |
| Глубина | Размер | Кол-во потоков | С++ |
| 1.00E+06 | 1.00E+07 | 1 | 1.32 |
| 1.00E+06 | 1.00E+07 | 2 | 1.03 |
| 1.00E+06 | 1.00E+07 | 3 | 0.96 |
| 1.00E+06 | 1.00E+07 | 4 | 0.91 |
| 1.00E+06 | 1.00E+07 | 5 | 0.87 |
| 1.00E+06 | 1.00E+07 | optimal | 0.81 |

# Вывод

Рома пидорас