Лабораторная работа

Слащинин Сергей, группа 17 МАГ-ИАД

18 января 2018 г.

В данной работе рассматриваются параллельные алгоритмы, реализующие различные методы для численного нахождения определенных интегралов.

Цель

Целью данной работы является анализ и реализация параллельных алгоритмов для численного интегрирования. Используя различные методы, необходимо вычислить:

$$\int_a^b f(x)\,\mathrm{d}x = \int_0^1 \frac{4}{x^2+1}\,\mathrm{d}x$$

Следует рассмотреть следующие методы численного интегрирования:

- метод прямоугольников
- метод трапеций
- метод Симпсона
- метод Гаусса 3-го порядка точности

Для каждого из них необходимо написать параллельную реализацию, используя программный интерфейс MPI, провести вычислительные эксперименты и проанализировать результаты.

Пункт 1

Технические характеристики используемого оборудования:

- ПК: Intel(R) Core(TM) i7-3632Q CPU @ 2.20GHz (4 физических ядра, 8 виртуальных), 8 Гб ОЗУ
- Кластер: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz (8 физических ядер, 16 виртуальных), 32 Гб ОЗУ

Далее представлены графики зависимости среднего времени работы алгоритма (за 40 запусков) от числа процессов при фиксированном значении шага интегрирования (и числа шагов n) $h = \frac{b-a}{n} = \frac{1-0}{10000000} = 0.0000001$ для персонального компьютера (Рис. 1) и $h = \frac{b-a}{n} = \frac{1-0}{1000000} = 0.000001$ для кластера (Рис. 2).

Как можно заметить из графиков, наименьшее среднее время работы программы достигается при количестве процессов p=8 и p=16 для персонального компьютера (Рис. 1) и кластера (Рис.2), соответственно. Это совпадает с количеством виртуальных ядер каждого процессора. Для метода прямоугольников и трапеций нужно вычислять значение функции $f(x)$ только в одной точке на каждом шаге интегрирования(для метода трапеций можно запоминать найденное значение с предыдущего шага), а для методов Симпсона (для него можно также запоминать значение с предыдущего шага) и Гаусса в двух точках, поэтому методы прямоугольников и трапеций работают заметно быстрее.

Пункт 2

Для каждого из методов ожидалось, что при постепенном увеличении числа процессов, когда их общее число превышает количество ядер процессора, среднее время работы программы будет расти. Это объясняется тем, что системе необходимо тратить дополнительное время на переключение ядер между одновременно выполняемыми процессами и синхронизацию (а также запуск) процессов. Как видно из результатов в п.1, эксперименты подтвердили эти ожидания. Наименьшее вермя работы на персональном компьютере достигается при количетсве потоков p=8, а на кластере при p=16, а затем начинает постепенно расти.

Рис. 1: Среднее время работы программы при $n = 10000000$ с разным количеством процессов p (на персональном компьютере)

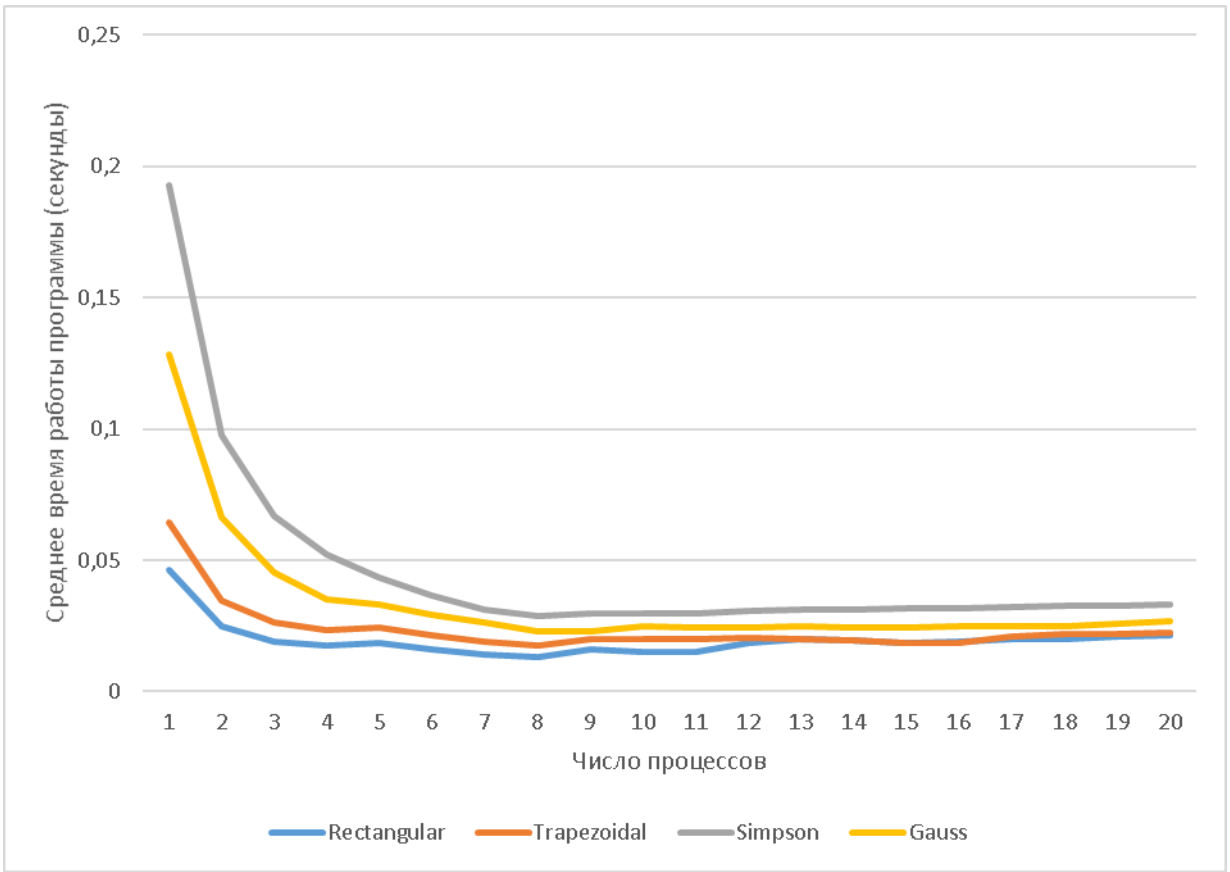
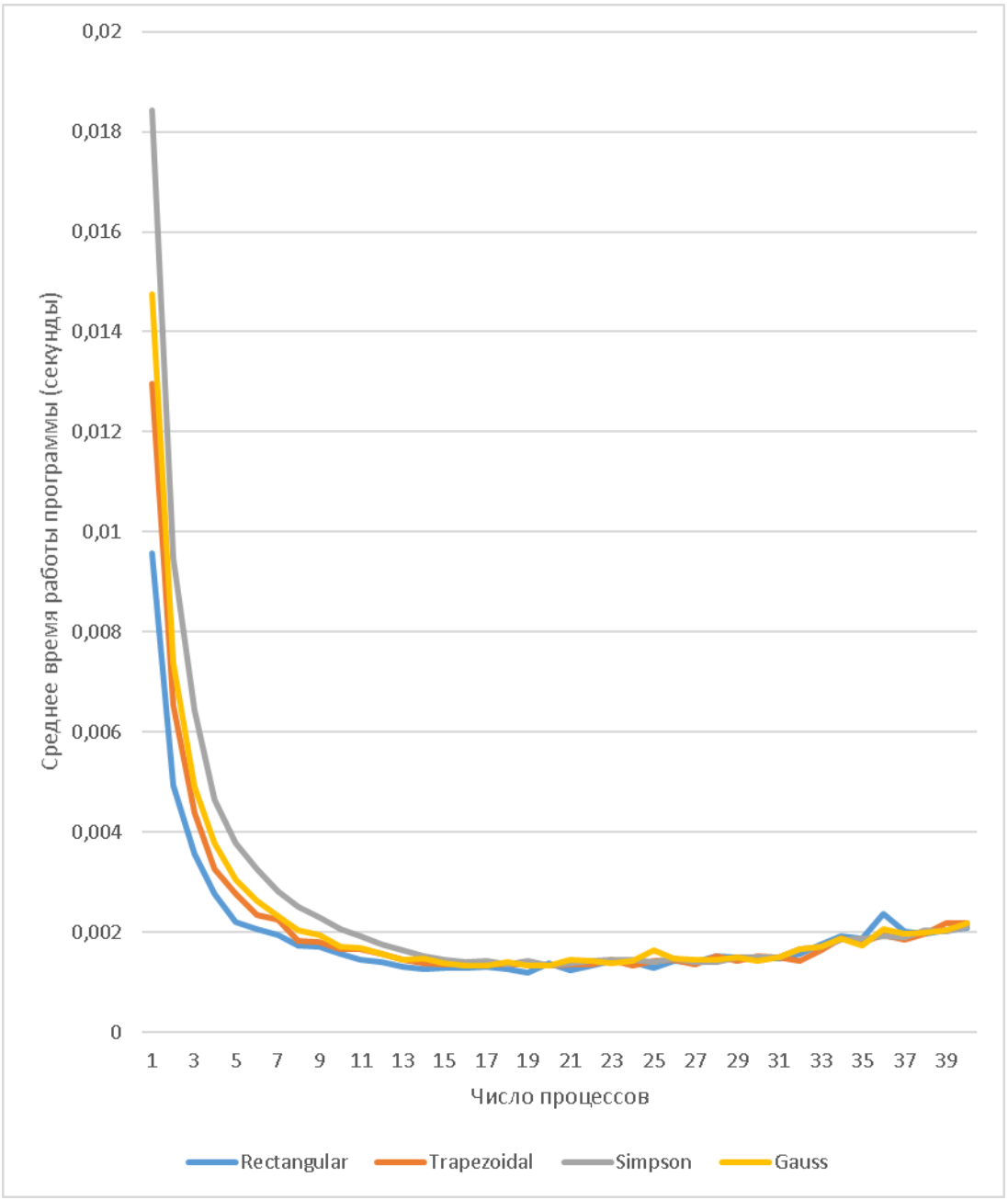


Рис. 2: Среднее время работы программы при $n = 1000000$ с разным количеством процессов p (на кластере)



Пункт 3

Пусть p - число параллельных процессов, n - число шагов интегрирования, $T(n,p)$ - общее время работы программы, $T_0(p)$ - время, необходимое на инициализацию, запуск и синхронизацию процессов, $C(n)$ - время, необходимое на вычислений на n отрезках интегрирования. Тогда:

$$T(n,p) = T_0(p) + C(\frac{n}{p})$$

Ускорением параллельного алгоритма называют отношение времени выполнения лучшего последовательного алгоритмам к времени выполнения параллельного алгоритма:

$$S(p) = \frac{T(n,1)}{T(n,p)}$$

$C(1)$ для каждого метода отличается, т.к. для вычисления приближения на отрезке для каждого метода функция $f(x)$ вычисляется в разном количестве точек и выполняется разное количество вспомогательных операций. Т.к. операции вычисления приближения интеграла на каждом из отрезков выполняются последовательно, а также включают в себя одинаковое число операций, можно считать, что $C(n) = p \cdot C(\frac{n}{p})$ и также $T_0(1) = 0$ при $n \gg p$. Получаем следующую формулу для ускорения:

$$S(p) = \frac{C(n)}{T_0(p) + C(\frac{n}{p})}$$

Эффективностью параллельного алгоритма называется величина:

$$E(p) = \frac{S(p)}{p}$$

Для $p = 8$ и $n = 10000000$ на практике были получены следующие результаты:

- Метод прямоугольников: $S(8) = 4.2987$, $E(8) = 0.537$
- метод трапеций: $S(8) = 4.2365$, $E(8) = 0.53$
- метод Симпсона: $S(8) = 6.7279$, $E(8) = 0.841$
- метод Гаусса 3-го порядка точности: $S(8) = 5.9721$, $E(8) = 0.747$

Пункт 4

Теоретические оценки для ускорения и эффективности:

$$S(p) = \frac{C(n)}{T_0(p) + C(\frac{n}{p})} \approx p = 8,$$

$$E(p) = \frac{S(p)}{p} \approx 1,$$

при условии, что $n \gg p$ и $T_0(p) \ll C(\frac{n}{p})$.

К сожалению, значения ускорения и эффективности, полученные на практике , довольно далеки от теоретических оценок. Это происходит потому, что условие $T_0(p) \ll C(\frac{n}{p})$ не выполняется для выбранного n. Наибольшее ускорение/эффективность показал параллельный алгоритм, использующий метод Симпсона, т.к. время $C(1)$ нахождения приближения на каждом шаге (а следовательно и $C(\frac{n}{p})$) для него наибольшее.

Пункт 5

Далее представлен график (Рис. 3) зависимости среднего времени работы алгоритма, применяемого для вычисления интеграла:

$$\int_0^N \frac{4}{x^2+1}\,\mathrm{d}x$$

от верхнего предела интегрирования N при $h = 0.000001$.

Ожидается, что при увеличении верхнего предела интегрирования, среднее время работы программы будет расти линейно (как и число шагов).

Как показали результаты эксперимента (Рис. 3), зависимость среднего времени работы от N линейная.

Рис. 3: Среднее время работы программы при $n = 1000000$ с при увеличении верхнего предела интегрирования N (на кластере)