**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

Лабораторная работа №3

**по дисциплине**

**«Распределенные программные системы и технологии»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3308 |  | Сысоев В.Б. |
| Преподаватель |  | Пазников А.А. |

Санкт-Петербург

2018

# 1. Задание

1. Разработать параллельную MPI-программу решения систем линейных алгебраических уравнений (СЛАУ).
2. Провести эксперименты и построить графики зависимости ускорения и коэффициента эффективности от количества потоков, оценить масштабируемость, построить оценки вычислительной и коммуникационной сложности.

Вариант задания:

2. Решение СЛАУ методом сопряжённых градиентов.

**Замечания:**

* На вход программы подаётся случайно сгенерированная СЛАУ Ax = b. Результатом выполнения программы является вектор решений x.
* Размер матрицы выбирается достаточно большим, чтобы время выполнения программ в лучшем случае было не меньше 10 секунд.
* В программе, реализующей метод сопряжённых градиентов, необходимо проверить, применим ли данный метод для исходной матрицы A.

# 2. Описание решения задачи

Реализованный метод сопряженных градиентов применим на симметричной, положительно определенной матрице СЛАУ размера n x n.

Для СЛАУ вида:

Процесс решения представляет минимизацию следующего функционала:

где в скобках обозначено скалярное произведение.

Подготовительный шаг решения (s=0) определяется формулами:

где x0 - произвольное начальное приближение, а коэффициент α0 вычисляется как:

Основные шаги (s = 1, 2, …, n-1) определяются формулами:

Здесь rs - невязка s - го приближения.

Анализ формул показывает, что самой затратной является операция умножения матрицы A на вектор hs, что делает вычислительную сложность алгоритма квадратичной.

Данный метод в теории является точным и позволяет решить СЛАУ не более чем за n шагов, однако, из-за погрешностей операций с плавающей точкой, на практике он применяется как итерационный.

Операция умножения матрицы на вектор, как и другие операции над векторами хорошо распараллеливаются, что и было сделано в данной реализации алгоритма. На каждой итерации процессы обмениваются двумя числами с плавающей точкой, для нахождения коэффициентов α и β, а также, каждый из процессов получает полный текущий вектор rs. Это делает коммуникационную сложность линейно зависимой от размера СЛАУ n.

# 3. Организация экспериментов

Эксперименты проводились на персональном компьютере с процессором Intel Core i7 4770, 16 Гб оперативной памяти и ОС Fedora 25.

Для компиляции использовался GCC версии 6.4.1, библиотека mpich версии 3.2. Основными ключами компиляции были -O3 -DNDEBUG и специфичные для mpich.

Для проведения экспериментов была взята СЛАУ с 10200 неизвестными. Каждый запуск программы включал в себя 500 итераций. Для удовлетворения условиям применимости метода сопряженных градиентов, генерируемая матрица являлась симметричной и содержала только положительные элементы. Результаты экспериментов представлены ниже.

# 4. Результаты экспериментов

На рисунке 2 представлен график ускорения в зависимости от количества процессов для описанной ранее СЛАУ.

Рисунок 2. График ускорения.

На рисунке 3 представлен график эффективности в зависимости от количества процессов для описанной ранее СЛАУ.

Рисунок 3. График эффективности.

Для выявления равномерности загрузки, в приложении предусмотрен подсчет времени выполнения каждого из процессов. Результаты подсчета, для 4 процессов представлены ниже:

Запускается /usr/lib64/mpich/bin/mpirun...

Process 1/4 execution time: 22.965529

Process 2/4 execution time: 22.950177

Process 3/4 execution time: 22.952523

Main process 0/4 execution time: 22.766096

RESULT:

Check delta sum: 266.12

/usr/lib64/mpich/bin/mpirun завершился с кодом 0

Как видно из представленных результатов, дисбаланса загрузки не наблюдается. Меньшее время работы главного процесса обусловлено созданием исходной матрицы СЛАУ, что не учитывается в счетчиках. Данные результаты вполне ожидаемы, учитывая синхронизацию процессов на каждом шаге итерации.

# 5. Анализ результатов экспериментов и выводы

Итоги экспериментов говорят о том, несмотря на хорошо поддающиеся распараллеливанию операции, с ростом числа процессов эффективность линейно снижается. Это можно объяснить возрастающей коммуникационной сложностью и накладными расходами на синхронизацию процессов на каждой итерации. Разумным решением данной проблемы видится увеличение вычислительной нагрузки на каждый из процессов в течении одной итерации. Другими словами, данная реализация будет более эффективна с ростом размера СЛАУ.

# 6. Список использованных литературных источников

1. Wikipedia, Метод сопряжённых градиентов (для решения СЛАУ) [Электронный ресурс]. URL:  https://ru.wikipedia.org/wiki/Метод\_сопряжённых\_градиентов\_(для\_решения\_СЛАУ) (дата обращения: 20.04.2018).
2. Пазников А.А. Параллельные вычислительные технологии: Практикум. – Новосибирск: Печатный Двор, 2016. – 59 с.
3. G. Rapin. Modelling and implementation of algorithms in applied mathematics using MPI Lecture 4: Conjugate Gradient (CG) method, Programming MPI [Электронный ресурс]. URL:  http://www.inf.ufpr.br/summerschool2011/PDF/rapin-MPI-Part04.pdf (дата обращения: 20.04.2018).
4. ИНТУИТ Академия Intel: Intel Parallel Programming Professional (Introduction) Лекция 7 Метод сопряженных градиентов [Электронный ресурс]. URL: https://www.intuit.ru/studies/courses/4447/983/lecture/14931?page=10 (дата обращения: 20.04.2018).