**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ...................................................................................................................... 4

1. АНАЛИЗ ЗАДАЧ ПРОЕКТИРОВАНИЯ………………………….......................... 6

2. РАЗРАБОТКА И ОПИСАНИЕ ПОСЛЕДОВАТЕЛЬНОГО АЛГОРИТМА ПРОГРАММНОЙ СИСТЕМЫ....................................................................................... 7

3. РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ, РЕАЛИЗУЮЩЕЙ ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ ОБРАБОТКИ………………………………8

4. РАЗРАБОТКА И ОБОСНОВАНИЕ ВАРИАНТА СХЕМЫ ПАРАЛЛЕЛИЗАЦИИ АЛГОРИТМА...…......……………………………………….10

5. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ СИСТЕМЫ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ……………………………………………………………...11

6. ТЕСТИРОВАНИЕ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ПРОГРАММНЫХ СИСТЕМ С ПОСЛЕДОВАТЕЛЬНОЙ И ПАРАЛЛЕЛЬНОЙ ОБРАБОТКОЙ…………………...................................................................................14

ЗАКЛЮЧЕНИЕ..............................................................................................................17СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……............................................18

ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ

ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ

# ВведениЕ

Довольно часто самые сложные алгоритмы требуют огромного количества вычислительных ресурсов в реальных задачах, когда программист пишет код в стандартном его понимании процедурного или объектно-ориентированного Программирования (ООП), то для особо требовательных алгоритмических задач, которые работают с большим количеством данных и требуют минимизировать время выполнения задачи, необходимо производить оптимизацию. Задача оптимизации алгоритма сводится к тому, чтобы правильно выстроить последовательность операций и оптимально разместить данные в Кэше, минимизировав количество возможных пересылок данных из памяти. Параллелизация позволяет пользоваться не только вычислительными возможностями одного процессора, а множеством процессоров, в том числе и в системах с распределенной памятью, и т.п. В подобных системах как правило намного больше вычислительных мощностей, но этим всем нужно уметь управлять и правильно организовывать параллельную работу на конкретных потоках.

В настоящее время существуют два основных подхода к распараллеливанию вычислений. Это параллелизм данных и параллелизм задач. В основе обоих подходов лежит распределение вычислительной работы по доступным пользователю процессорам параллельного компьютера. При этом приходится решать разнообразные проблемы. Прежде всего, это достаточно равномерная загрузка процессоров, так как если основная вычислительная работа будет ложиться на один из процессоров, мы приходим к случаю обычных последовательных вычислений, и в этом случае никакого выигрыша за счет распараллеливания задачи не будет. Сбалансированная работа процессоров - это первая проблема, которую следует решить при организации параллельных вычислений. Другая и не менее важная проблема - скорость обмена информацией между процессорами. Если вычисления выполняются на высокопроизводительных процессорах, загрузка которых достаточно равномерная, но скорость обмена данными низкая, основная часть времени будет тратиться впустую на ожидание информации, необходимой для дальнейшей работы данного процессора.

В настоящее время существует много технологий которые позволяют наиболее легко организовать параллельные вычисления, одна из самых популярных на текущий момент это MPI. Используемый в данном курсовом проекте протокол MPI использует стиль программирования, основанный на параллелизме задач. Параллелизм задач подразумевает, что вычислительная задача разбивается на несколько относительно самостоятельных подзадач и каждый процессор загружается своей собственной подзадачей.

MPI - Message Passing Interface (интерфейс передачи сообщений). Название носит смысловой характер, что будет ярко выражено в последующих статьях, а пока все что требуется знать - основной способ взаимодействия параллельных процессов в такой системе - передача сообщений между ними. Это означает что наши потоки будут между собой общаться при обработке данных, а как это уже вопрос реализации.

# 1. АНАЛИЗ ЗАДАЧ ПРОЕКТИРОВАНИЯ

В данном курсовом проекте нам предлагается возможность спроектировать многокомпонентную вычислительную систему, состоящую из вычислительных модулей (ВМ), каждый из которых имеет память для хранения обрабатываемых данных и результатов (в рамках курсового проектирования это набор рабочих станций в составе локальной вычислительной сети).

Исходные данные для выполнения задания:

1. Количество используемых вычислительных узлов (процессов): 4 процесса
2. Тип графического формата (ч/б изображение): формат bmp
3. Размерность исходных данных: N\*N (N = 3000 пикселей)
4. Описание задачи обработки данных: уменьшение графического изображения с заданным масштабом; Выделение границ элементов чёрно-белого графического изображения на основе дифференциального оператора Робертса (дифференциальный фильтр I)
5. Язык программирования – C/С++, библиотека MPI (Message Passing Interface)
6. Алгоритмы обработки изображений [1]
7. Аппаратная база – рабочие станции в составе ЛВС, кластерные, мультипроцессорные, многоядерные архитектуры.

Для декодирования и кодирования изображений .bmp формата использовалась библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом OpenCV2. Версия mpich – 4.0.2.

В ходе выполнения курсового проекта необходимо разработать и реализовать последовательный и параллельный алгоритмы обработки изображения согласно начальным условиям и входным данным.

**2. РАЗРАБОТКА И ОПИСАНИЕ ПОСЛЕДОВАТЕЛЬНОГО АЛГОРИТМА ПРОГРАММНОЙ СИСТЕМЫ**

В данном курсовом проекте был реализован следующий последовательный алгоритм программной системы:

1. Чтение изображения в виде двумерной матрицы пикселей, принимающих значение от 0 до 255 (ч\б изображение)
2. Применение дифференциального оператора Робертса для выделения границ элементов чёрно-белого графического изображения
3. Ввод масштаба нового изображения с клавиатуры и изменение его размеров
4. Создание нового изображения на основе полученного после всех изменений и вывод его на экран.

Данный алгоритм действий позволяет достичь результат, описанный в задании курсового проекта.

**3. РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ, РЕАЛИЗУЮЩЕЙ ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ ОБРАБОТКИ**

Перед реализацией программной системы необходимо установить и подключить библиотеку компьютерного зрения OpenCV2.

В ходе разработки программной системы, реализующей последовательный алгоритм обработки, были реализованы следующие алгоритмы:

1. Алгоритм выделения границ объектов графического изображения с применением дифференциального оператора Робертса
2. Алгоритм изменения размера картинки при помощи ввода масштаба с клавиатуры.
3. Алгоритм чтения изображения.

**Алгоритм выделения границ объектов графического изображения с применением дифференциального оператора Робертса:**

Входные данные: двумерная матрица чёрно-белого изображения.

Выходные данные: двумерная матрица изменённого чёрно-белого изображения.

Алгоритм:

1. Начинается цикл, проходящий по каждому пикселю изображения, в котором:

А) p1 = пиксель(х, у), p2 = пиксель (х+1, у+1), р3 = пиксель (х+1, у), р4 = пиксель (х, у+1)

Б) переменная А = р1 – р2, Б = р3 – р4

В) Результирующий пиксель: R = (А^2 + Б^2)^1/2

Г) R записывается как значение текущего пикселя

1. После прохождения данного цикла, изображение было обработано, возвращается матрица видоизменённого графического изображения.

**Алгоритм изменения размера картинки при помощи ввода масштаба с клавиатуры:**

Входные данные: матрица с изображением изначальных размеров, значение масштаба картинки.

Выходные данные: матрица изображения с изменённым масштабом.

Алгоритм:

1. Вывод сообщения о необходимости ввода масштаба
2. Ввод значения масштаба картинки с клавиатуры
3. Изменения масштаба картинки с применением функции Resize из библиотеки OpenCV2.

**Алгоритм чтения изображения:**

Входные данные: строка пути к файлу.

Выходные данные: матрица изображения.

Алгоритм:

1. Путь к файлу содержится в специальной строке image\_path.
2. При помощи функции imread библиотеки OpenCV2 происходит чтение данных файла, указанного в строке image\_path, в режиме чёрно-белого изображения. Данные о изображении записываются в двумерную матрицу, где хранятся в виде пикселей которые могут принимать значение от 0 до 255.

**4.** **РАЗРАБОТКА И ОБОСНОВАНИЕ ВАРИАНТА СХЕМЫ ПАРАЛЛЕЛИЗАЦИИ АЛГОРИТМА**

Данный курсовой проект подразумевает в себе выполнения огромного множества однотипных и времязатратных операций, которые не зависят от прошлой операции.

Именно поэтому в данном проекте было принято решение об использовании нескольких процессов для совместного решения одной и той же задачи, то есть разложить решенную задачу на несколько частей, и каждая часть вычисляется параллельно независимым процессором.

Параллельная вычислительная система может быть специально разработанным суперкомпьютером с несколькими процессорами или кластером из нескольких независимых компьютеров, соединенных определенным образом.

В данном курсовом проекте был разработан следующий вариант схемы параллелизации:

1. Инициализация n процессов MPI.
2. Если процесс == 0:
   1. Чтение ч\б изображения из файла raw.bmp
   2. Вычисление количества каналов изображения, полный размер матрицы изображения
   3. Вычисление размера блока изображения (полный размер делится на количество процессов – 4)
3. Отправка отдельного блока изображения каждому процессу
4. Выделение границ объектов своего блока каждым процессом
5. Сборка блоков изображений в единое целое
6. Изменение размеров изображения согласно масштабу
7. Запись изображения в файл computed.bmp.

**5. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ СИСТЕМЫ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ**

Перед реализацией программной системы необходимо установить и подключить библиотеку компьютерного зрения OpenCV2, а также заголовочный файл mpi.h.

В ходе разработки программных модулей системы параллельной обработки данных были реализованы следующие алгоритмы:

1. Алгоритм чтения изображения
2. Алгоритм распределения блоков изображения по процессам
3. Алгоритм обработки блока изображения каждым отдельным процессом
4. Алгоритм сборки целого изображения из нескольких блоков
5. Алгоритм изменения масштаба изображения
6. Алгоритм записи изображения в файл.

**Алгоритм чтения изображения:**

Входные данные: строка пути к файлу.

Выходные данные: матрица изображения.

Алгоритм:

1. Путь к файлу содержится в специальной строке image\_path.
2. При помощи функции imread библиотеки OpenCV2 происходит чтение данных файла, указанного в строке image\_path, в режиме чёрно-белого изображения. Данные о изображении записываются в двумерную матрицу, где хранятся в виде пикселей которые могут принимать значение от 0 до 255.

**Алгоритм распределения блоков изображения по процессам:**

Входные данные: матрица с изображением, кол-во процессов.

Выходные данные: блоки изображения для обработки каждым процессом.

Алгоритм:

1. Вычисление размера одного блока – размер всей матрицы в байтах делится на общее количество процессов, которые будут обрабатывать изображение.
2. Выделение памяти типа unsigned char (массив) для хранения блока изображения на обработку одному процессу.
3. Распределение блоков изображения при помощи использования команды MPI\_SCATTER.

**Алгоритм обработки блока изображения каждым отдельным процессом:**

Входные данные: необработанный блок изображения.

Выходные данные: обработанный блок изображения.

Алгоритм:

1. Процесс получил блок данных unsigned char чёрно-белого изображения.
2. Начинается цикл, проходящий по каждому пикселю блока изображения, в котором:

А) p1 = пиксель(х, у), p2 = пиксель (х+1, у+1), р3 = пиксель (х+1, у), р4 = пиксель (х, у+1)

Б) переменная А = р1 – р2, Б = р3 – р4

В) Результирующий пиксель: R = (А^2 + Б^2)^1/2

Г) R записывается как значение текущего пикселя

1. Установка барьера для всех процессов MPI\_BARRIER.

**Алгоритм сборки целого изображения из нескольких блоков:**

Входные данные: обработанные блоки изображения.

Выходные данные: матрица обработанного изображения.

Алгоритм:

1. Если процесс == 0: инициализация матрицы для хранения обработанного изображения таких же размеров и типа, как исходное.
2. Сборка ранее разбросанной по процессам информации при помощи команды MPI\_GATHER в ранее созданную переменную для хранения обработанного изображения.

**Алгоритм изменения масштаба изображения:**

Входные данные: матрица с изображением изначальных размеров, значение масштаба картинки.

Выходные данные: матрица изображения с изменённым масштабом.

Алгоритм:

1. Вывод сообщения о необходимости ввода масштаба
2. Ввод значения масштаба картинки с клавиатуры
3. Изменения масштаба картинки с применением функции Resize из библиотеки OpenCV2.

**Алгоритм записи изображения в файл:**

Входные данные: матрица с обработанным изображением.

Выходные данные: файл содержащий обработанное изображение.

Алгоритм:

1. Если процесс == 0:
   1. При помощи функции imwrite библиотеки OpenCV2 записываем обработанное изображение в файл computed.bmp.

**6. ТЕСТИРОВАНИЕ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ПРОГРАММНЫХ СИСТЕМ С ПОСЛЕДОВАТЕЛЬНОЙ И ПАРАЛЛЕЛЬНОЙ ОБРАБОТКОЙ**

Тестирование программы осуществлялось в интегрированной среде разработки Microsoft Visual Studio, C++ стандарт – 17.

Операционная система – Windows 10 21H2.

Аппаратное обеспечение – процессор Intel(R) Core(TM) i3-1005G1 1.20GHz, оперативная память 8 GB LPDDR4.

Для измерения времени работы использовалась библиотека chrono, которая позволяет засчитывать временные промежутки с приемлемой точностью.

Для тестирования использовалось grayscale изображение (см. рисунок 6.1).

****

Рисунок 6.1 – Исходное изображение для тестирования

После последовательной или параллельной обработки получается следующее выходное изображение (см. рисунок 6.2).



Рисунок 6.2 – Выходное изображение

Выполним тестирование и рассчитаем среднюю производительность разработанных программ с последовательной (=1/T1;) и параллельной (=1/Тр)обработкой на исходных данных заданной размерности (согласно варианту задания), Т1 Тр - фактическое среднее время последовательной и параллельной программы. Для этого рассчитаем средние значения времен выполнения процессов по результатам нескольких (не менее 10) экспериментов; при этом зафиксируем как наблюдаемые, так и средние значения (см. таблицу 6.1).

|  |  |  |
| --- | --- | --- |
| № | Время выполнения параллельного алгоритма Tp, секунд | Время выполнения последовательного алгоритма T1, секунд |
| 1 | 0.263 | 0.674 |
| 2 | 0.251 | 0.522 |
| 3 | 0.281 | 0.484 |
| 4 | 0.255 | 0.356 |
| 5 | 0.251 | 0.565 |
| 6 | 0.274 | 0.560 |
| 7 | 0.265 | 0.609 |
| 8 | 0.286 | 0.498 |
| 9 | 0.215 | 0.580 |
| 10 | 0.256 | 0.458 |
| Среднее значение | 0.26 | 0.531 |

На основе полученных результатов рассчитаем коэффициент увеличения производительности (ускорения) за счет параллелизации:

Таким образом, параллельный алгоритм в два раза быстрее обрабатывает исходное графическое изображение, чем то же самое изображение обрабатывает последовательный, что значит следущее: за счёт разделения программы на несколько процессов, получилось достичь прироста производительности чуть более чем в два раза, что показывает высокую эффективность применения параллельных систем в сфере компьютерных вычислений. Также во время разработки данного курсового проекта было подмечено, что при увеличении исходных данных, необходимость в параллельной обработке возрастает, так как с каждым увеличением размера картинки поднималась и эффективность параллельного алгоритма, по сравнению с последовательным.

**ЗАКЛЮЧЕНИЕ**

Согласно условиям задания были разработаны и описаны последовательные алгоритмы программной системы, разработаны и обоснованы варианты схемы параллелизации алгоритма, созданы программные модули системы параллельной обработки данных, а также произведены тестирование и сравнительный анализ производительности программных систем с последовательной и параллельной обработкой.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов.
2. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
3. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.
4. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.
5. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
6. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.
7. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.
8. ГОСТ 19.402-78. ЕСПД. Описание программы.
9. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
10. ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.