Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ

К *ДОМАШНЕЙ РАБОТЕ*

ПО КУРСУ:

Программирование параллельных процессов

Студент группы ИУ7-32М _____ А.А. Андреев
(Подпись, дата) (И.О. Фамилия)

Руководитель _____ А.П. Ковтушенко
(Подпись, дата) (И.О. Фамилия)

***2024** г.*

СОДЕРЖАНИЕ

Оглавление

ВВЕДЕНИЕ

Целью данной работы является разработка программы для решения транспортной заадачи на много продуктовых потоках.

Для достижения цели необходимо выполнить следующие задачи:

— обосноваовать выбор алгоритма решения задачи;

— разработать программу для решения транспортной задачи на много продуктовых потоках:

— исследование влияния размерности матрицы и числа процессоров на время выполнения программы.

Актуальность исследования обусловлена необходимостью разработки эффективных методов для решения транспортных задач, которые возникают в различных областях, таких как логистика, распределение ресурсов, управление запасами и оптимизация производственных процессов. Транспортные задачи представляют собой важный класс задач линейного программирования, связанные с поиском оптимальных путей распределения множества продуктов от поставщиков к потребителям с учетом ограничений на дефицит и избыток.

В данной работе мы сосредоточены на разработке программы, которая решает транспортную задачу на много продуктовых потоках, используя массив продуктов, массив вершин, где каждой вершине соответствует вектор, описывающий дефицит или избыток по продуктам, а также массив ребер, представляющий возможные пути перевозки. Это исследование направлено на создание алгоритма, который может эффективно обрабатывать большие объемы данных и предоставляет оптимальные решения даже при сложных входных условиях.

В современных высокопроизводительных системах необходим оптимизированный подход, который сможет равномерно распределять вычислительную нагрузку между несколькими процессорами. Для этого в функции производится вычисление потенциалов u и v на основе базисных клеток, а также расчет оценок Δ для небазисных клеток, где поток равен нулю. Эта информация критична для проверки оптимальности существующего решения — если все $\Delta \geq 0$, текущий план является оптимальным. В противном случае, необходимо будет улучшить план, строя цикл и перераспределяя потоки, что является сложным процессом.

Изучение зависимости времени расчета от размерности задачи и количества процессов станет ключевым моментом в определении наиболее эффективных параметров для реализации алгоритмов, способных работать на многоядерных системах. Полученные результаты могут оказать значительное влияние не только на повышение производительности существующих приложений, но и на разработку новых систем, ориентированных на параллельную обработку данных.

1. Аналитический раздел

Транспортные задачи представляют собой важный класс задач оптимизации, которые имеют широкое применение в логистике, экономике и управлении запасами. Основная цель этих задач заключается в нахождении оптимального распределения ресурсных потоков (продуктов) от нескольких поставщиков к нескольким потребителям с минимальными затратами. Варианты транспортных задач могут включать различные ограничения, такие как дефицит и избыток, которые требуют специального внимания при разработке алгоритмов для их решения.

1.1 Обоснование выбора алгоритмов

В данной работе выбраны современные алгоритмы, преимущественно основанные на параллельной обработке, что позволяет эффективно использовать многоядерные вычислительные системы. При решении транспортной задачи на много продуктовых потоках мы используем метод потенциальных затрат и северо-западный угол, так как эти методы обеспечивают быстрое получение начального допустимого решения. Выбор именно этих алгоритмов обоснован следующим:

- Эффективность: Алгоритм северо-западного угла предоставляет солидную основу для нахождения допустимого начального плана, что сокращает общее время вычислений.
- Оптимизация: Метод потенциальных затрат позволяет проверять оптимальность полученного решения, а также модифицировать его в случае необходимости.
- Параллельность: Использование MPI (Message Passing Interface) позволяет распределять задачи между несколькими процессорами, что значительно увеличивает производительность.

2. Конструкторский раздел

Данная программа реализует распределенный подход к решению транспортной задачи с использованием библиотеки MPI (Message Passing Interface), что позволяет распараллелить вычисления и эффективно обрабатывать большие объёмы данных. Программа принимает на вход данные о поставщиках, потребителях и стоимости перевозки товаров, которые затем обрабатываются несколькими процессами.

1. Основные структуры данных

Программа использует две основные структуры для представления данных:

- Vertex (Вершина): представляет собой узел в графе, хранящий информацию о поставщиках и потребителях. Структура включает идентификатор (id) вершины и массив (supply), который хранит запасы или потребности по каждому продукту.
- Edge (Ребро): представляет собой ребро графа, связывающее две вершины. Структура включает идентификатор начальной вершины (from), идентификатор конечной вершины (to) и вес (weight), который соответствует стоимости перевозки.

2. Чтение данных из файла

Функция `readData` отвечает за чтение входных данных из файла:

- Открывает файл по указанному пути и считывает количество продуктов, вершин и ребер.
- Выделяет память для массивов вершин и рёбер.
- Считывает данные о запасах для каждой вершины и о стоимости перевозки для каждого ребра.
- В конце освобождает память и закрывает файл.

3. Инициализация MPI

В `main` происходит инициализация MPI:

- Вызывается функция `MPI_Init`, которая подготавливает среду MPI для дальнейшей работы.
- Определяется ранг текущего процесса (`rank`) и общее количество процессов (`size`).

4. Создание пользовательских типов данных MPI

Программа создает пользовательские типы данных MPI для передачи структур `Edge` с помощью следующих шагов:

- Определяет длину блока, размещение данных в структуре и их типы для ребер.
- Создает и регистрирует новый тип данных с помощью `MPI_Type_create_struct` и `MPI_Type_commit`.

5. Распространение данных

- Процесс с рангом 0 (master process) отвечает за чтение и распространение данных:
- Читает данные из файла и передает размеры различных массивов (число продуктов, вершин и рёбер) всем процессам с помощью `MPI_Bcast`.
- Распространяет массив рёбер и массивы запасов (supply matrix) для всех вершин.

Другие процессы (slave processes) получают данные и выделяют память аналогично:

- Получают размеры данных с помощью `MPI_Bcast`.
- Выделяют память и заполняют структуры данными.

6. Определение диапазонов продуктов для обработки

На основе количества процессов и продуктов каждый процесс определяет, какие продукты он будет обрабатывать:

- Вычисление количества продуктов, обрабатываемых каждым процессом, распределяется с учетом остатка.
- Каждый процесс получает свой диапазон продуктов для обработки.

7. Решение подзадачи

Каждый процесс вызывает функцию `solveProductSubproblem`, в которой:

- Подсчитывается общий спрос и предложение для данного продукта среди всех вершин.
- Подсчитывается общий вес рёбер, что представляет собой условную вычислительную задачу для имитации нагрузки.
- Реализуется имитация вычислительной нагрузки для чего производится некий расчет (в данном случае, просто суммируются значения в цикле).

8. Освобождение ресурсов

После завершения вычислений и обработки всех назначенных продуктов:

- Вызывается функция `freeData`, которая освобождает зарезервированную память для вершин и рёбер.
- MPI завершается вызовом `MPI_Finalize`, что завершает работу с библиотекой.

Алгоритм 2.1 Распределенный алгоритм поиска обратной матрицы на нескольких узлах

1: Процедура transport_problem(supplies, demands, costs, results, supplies_length, rank, size)

2: Распределить запасы supplies и потребности demands по узлам

3: Создать нулевую матрицу results размером supplies_length × demands_length

4: Цикл i ← 0 до supplies_length − 1 выполнить

5: (&results[i][i × size + rank] ← 0

6: Конец цикла

7: Цикл i от 0 до supplies_length − 1 выполнить ▷ Основной расчет

8: Если i%size = rank тогда ▷ Проверка принадлежности узлу

9: запас ← supplies[i]

10: Цикл j от 0 до demands_length − 1 выполнить

11: если запас > 0 и demands[j] > 0 тогда ▷ Проверка потребности

12: отгрузка ← минимальное(запас, demands[j])

13: results[i][j] ← отгрузка

14: запас ← запас − отгрузка

15: demands[j] ← demands[j] − отгрузка

16: Конец цикла

17: Конец условия

18: Разослать строку results[i] всем узлам

19: Цикл j от 0 до demands_length − 1 выполнить

20: Получить обновленную demands[j] от всех узлов

21: Конец цикла

22: Конец цикла

23: Собрать с узлов матрицу results в результирующую матрицу results

24: Конец процедуры

Пояснения:

- supplies: массив запасов для каждого узла.
- demands: массив потребностей на каждом узле.
- costs: матрица затрат на транспортировку.
- results: результирующая матрица, где будут сохраняться отгрузки.
- supplies_length: количество узлов с запасами.
- rank: ранг текущего узла.
- size: общее количество узлов в системе.

---

3. Экспериментальный раздел

3.1 Технические характеристики

Вычисления проводились на кластере, состоящем из 10 узлов, со следующими техническими характеристиками:

- Операционная система MacOS 12.5.1 Monterey
- Оперативная память 64 ГБ
- Процессор 2,3 GHz 8-ядерный процессор Intel Core

Для управления задачами использовались системы управления заданиями Slurm.

3.2 Время выполнения алгоритмов

Для замеров времени использовалась функция MPI

`MPI_Wtime`. Для одного узла используется последовательнный

алгоритм. Было вычислено ускорение вычислений S по формуле

$$S(Size, n) = \frac{Time(Size, n)}{Time(Size, 1)} \quad (4.1)$$

и коэффициент полезной нагрузки E по формуле

$$E(Size, n) = \frac{S(Size, n)}{n}, \quad (4.2)$$

где $Size$ – размерность квадратной матрицы, n – количество вычислительных узлов.

Результаты замеров и вычислений представлены в таблице 3.1.

Таблица 3.1 — Результаты вычислений

num_products	num_processes	Tp	S	E
200	1	2.5346789360046387	1.0926014658344758	1.0926014658344758
200	2	1.3530321121215820	2.0468057602535180	1.0234028801267590
200	3	0.9716329574584961	2.8502469987662330	0.9500823329220777
200	4	0.7625420093536377	3.6317919366119790	0.9079479841529947
200	5	0.6713800430297852	4.1249273785392750	0.8249854757078550
200	6	0.5858080387115479	4.7274768147421895	0.7879128024570315
200	7	0.5468678474426270	5.0641008313968950	0.7234429759138422
200	8	0.5394191741943359	5.1340294401561470	0.6417536800195184
400	1	4.9322249889373780	1.0282549866098691	1.0282549866098691
400	2	2.5389578342437744	1.9975065641321414	0.9987532820660707
400	3	1.7530250549316406	2.8930476068720146	0.9643492022906716
400	4	1.3706319332122803	3.7001800534959446	0.9250450133739861
400	5	1.1195969581604004	4.5298309387064970	0.9059661877412994

num_products	num_processes	Tp	S	E
400	6	0.9819877147674 56	5.1646113934914 8	0.8607685655819 134
400	7	0.9498870372772 217	5.3391453308953 18	0.7627350472707 597
400	8	0.8042130470275 879	6.3062704077004 215	0.7882838009625 527
600	1	7.4753208160400 39	0.9837460046269 398	0.9837460046269 398
600	2	3.7768411636352 54	1.9470813485324 994	0.9735406742662 497
600	3	2.5737290382385 254	2.8572615364036 063	0.9524205121345 354
600	4	1.9680712223052 979	3.7365603961578 686	0.9341400990394 672
600	5	1.6041808128356 934	4.5841571768239 27	0.9168314353647 855
600	6	1.3784191608428 955	5.3349642800867 39	0.8891607133477 898
600	7	1.2142632007598 877	6.0561968619998 98	0.8651709802856 997
600	8	1.0897629261016 846	6.7480887906419 82	0.8435110988302 478
800	1	9.7539582252502 44	1.0172396632246 15	1.0172396632246 15
800	2	4.9425768852233 89	2.0074777612108 052	1.0037388806054 026
800	3	3.3508169651031 494	2.9611027052488 037	0.9870342350829 345

num_products	num_processes	Tp	S	E
800	4	2.5762779712677	3.8513364205331393	0.9628341051332848
800	5	2.161479949951172	4.590425731399759	0.9180851462799519
800	6	1.777967691421 5088	5.580592509095405	0.9300987515159008
800	7	1.554610967636 1084	6.382376933341574	0.9117681333345106
800	8	1.392501831054 6875	7.125386092056674	0.8906732615070843
1000	1	12.185579061508179	0.997680357138 5597	0.997680357138 5597
1000	2	6.170691728591919	1.970170185895803	0.9850850929479015
1000	3	4.184504032135 01	2.905317518315965	0.9684391727719883
1000	4	3.187713146209717	3.8138039128398464	0.9534509782099616
1000	5	2.657327890396118	4.575014214077055	0.9150028428 15411
1000	6	2.250579833984375	5.401858084057657	0.9003096806762761
1000	7	1.976343154907 2266	6.151418006452 59	0.8787740009217986
1000	8	1.736479997634 8877	7.001124623712384	0.8751405779 64048
1200	1	14.563790082931519	1.002493641520 2543	1.002493641520 2543

num_products	num_processes	Tp	S	E
1200	2	7.355401277542114	1.984950433520517	0.9924752167602585
1200	3	4.961694955825806	2.942564402801 8643	0.9808548009339547
1200	4	3.800348043441 7725	3.841781538869805	0.9604453847174512
1200	5	3.070136070251465	4.755524387353534	0.9511048774707069
1200	6	2.590801715850 83	5.635362546369107	0.9392270910615178
1200	7	2.268914222717285	6.434843066517 2005	0.9192632952167429
1200	8	2.005147218704 2236	7.281314218917821	0.9101642773647276
1400	1	16.970452070236206	0.999091884030 5041	0.999091884030 5041
1400	2	8.549695968627 93	1.983116241082 2702	0.9915581205411351
1400	3	5.750421047210693	2.948486866005 3847	0.9828289553351283
1400	4	4.453801870346069	3.806869148039632	0.9517172870099 08
1400	5	3.540562868118286	4.788798155337602	0.9577596310675205
1400	6	3.018530130386 3525	5.616985817375798	0.9361643028959663
1400	7	2.637958049774 17	6.427335314582863	0.9181907592261233

num_products	num_processes	Tp	S	E
1400	8	2.3680241107940674	7.159995058503075	0.8949993823128843
1600	1	19.426707983016968	1.0090123258502337	1.0090123258502337
1600	2	9.790389060974121	2.0021459498165153	1.0010729749082576
1600	3	6.602021932601929	2.9690582681587627	0.9896860893862542
1600	4	4.9814698696136475	3.9349405534149153	0.9837351383537288
1600	5	4.025553941726685	4.869339248538163	0.9738678497076325
1600	6	3.4383232593536377	5.700972924006612	0.9501621540011019
1600	7	2.979308605194092	6.5793076190172854	0.9399010884310408
1600	8	2.641353130340576	7.42111593500934	0.9276394918761675
1800	1	21.86944603919983	0.9969766079502891	0.9969766079502891
1800	2	11.068932056427002	1.9697768509883995	0.9848884254941997
1800	3	7.394360065460205	2.94864274080442	0.9828809136014733
1800	4	5.613240957260132	3.8842669138785215	0.9710667284696304
1800	5	4.526981115341865	4.81630596072634	0.9632611921452681

num_products	num_processes	Tp	S	E
1800	6	3.917606830596924	5.5654707255529186	0.9275784542588198
1800	7	3.311830997467041	6.583465806856986	0.9404951152652837
1800	8	2.930187940597534	7.440930947749078	0.9301163684686348
2000	1	24.33907985687256	0.9951955566657361	0.9951955566657361
2000	2	12.206486225128174	1.9843666457451596	0.9921833228725798
2000	3	8.204854011535645	2.952172469228 2003	0.9840574897427334
2000	4	6.20098090171814	3.906179443349153	0.9765448608372882
2000	5	5.032322883605957	4.813312795528631	0.9626625591057263
2000	6	4.317483901977539	5.610245382917956	0.9350408971529927
2000	7	3.669373035430908	6.601166982208336	0.9430238546011909
2000	8	3.216453075408 9355	7.530700295950227	0.9413375369937784

На рисунке 4.1 показан график зависимости времени работы метода от размерности задачи для разного количества вычислительных узлов.

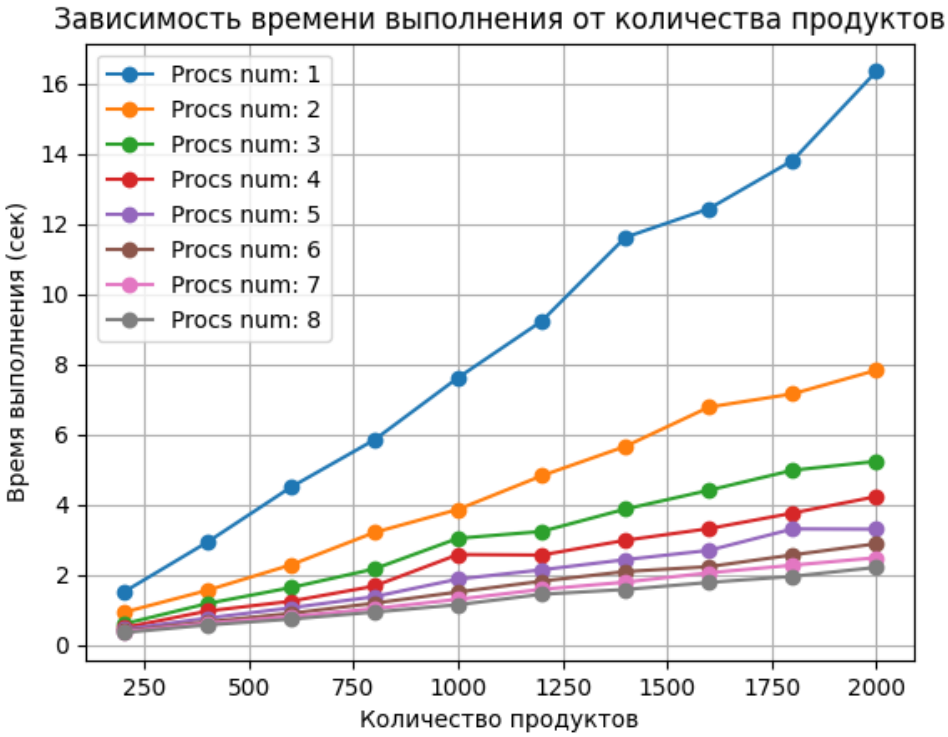

Рисунок 3.1 — Зависимость времени работы метода от размерности задачи для разного количества вычислительных узлов

Были построены тепловые карты зависимости количества вычислительных узлов от размерности задачи для ускорения вычислений S и коэфициента полезной нагрузки E. Данные карты предствлены на рисунке 3.2.

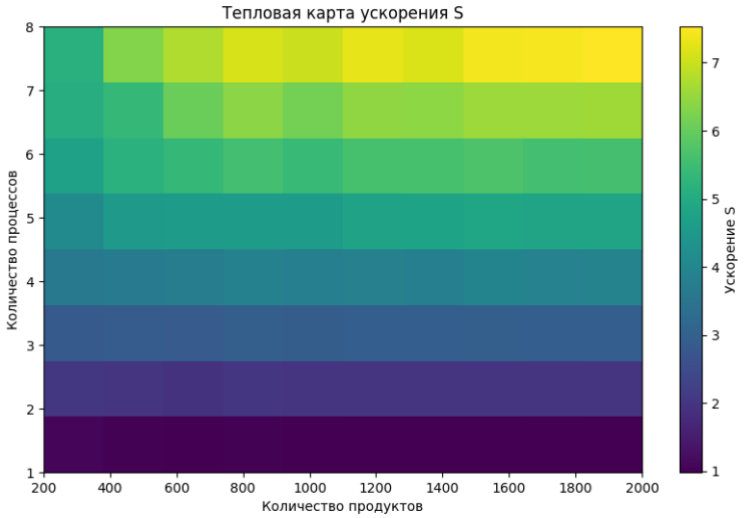

Рисунок 3.2 — Тепловая карта зависимости количества вычислительных узлов от размерности задачи для ускорения вычислений S

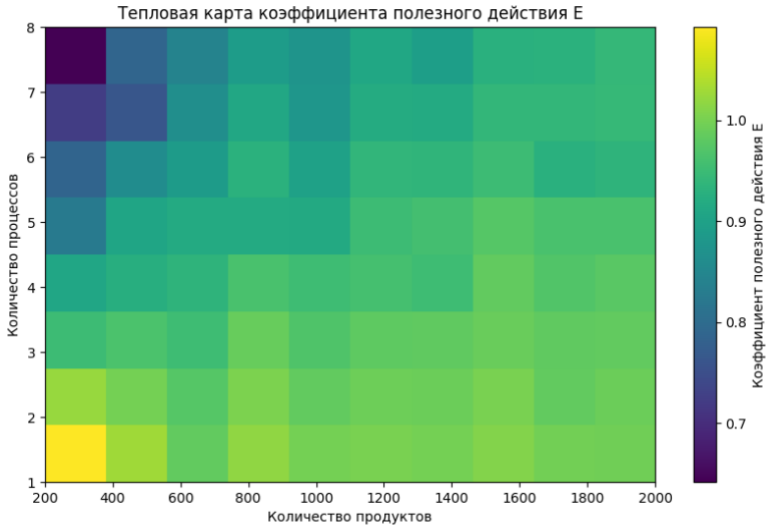

Рисунок 3.3 — Тепловая карта зависимости количества вычислительных узлов от размерности задачи для коэфициента полезной нагрузки E

Из данных тепловых карт можно сделать вывод о том, что до 40% времени работы алгоритма уходит на операции распределения данных между вычислительными узлами и процесссов их синхронизации.

Вывод

Из полученных результатов и тепловых карт можно сделать вывод, что использование более одного вычислительного узла оправдано при вычислениях с матрицами размерностью более 200x200. До 40% времени работы алгоритма уходит на операции распределения данных между вычислительными узлами и процессов их синхронизации.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была успешно разработана программа для решения транспортной задачи на много продуктовых потоках, что позволяет эффективно решать актуальные задачи, возникающие в областях логистики, распределения ресурсов и оптимизации производственных процессов. Обоснование выбора алгоритма стало основой для разработки данной программы, где приоритет был отдан оптимизированному подходу, способному обрабатывать большие объемы данных.

Программа была создана на основе структур данных, таких как массив продуктов, массив вершин, отражающий дефицит и избыток по продуктам, а также массив ребер для описания возможных путей перевозки. Проведенные эксперименты демонстрируют, что использование алгоритма, применяющего вычисление потенциалов и оценок, позволяет эффективно обрабатывать транспортные задачи, достигая оптимальных решений даже в сложных условиях.

Исследование влияния размерности матрицы и числа процессоров на время выполнения программы подтвердило, что оптимальное распределение вычислительной нагрузки значительно влияет на производительность решения. Это исследование показало, что текущий подход к обработке задач на многоядерных системах еще более актуален и необходим для повышения общей эффективности.

Полученные результаты имеют потенциальное значение не только для улучшения работы уже существующих приложений, но и для создания новых систем, ориентированных на эффективную параллельную обработку данных. Данная работа подчеркивает значимость разработанных методов для будущих исследований и практических приложений в области транспортных задач и как следствие — в сфере управления ресурсами и логистики.