Параллельные вычисления

Параллельная реализация операций с сеточными данными на неструктурированной смешанной сетке

Лазарев Владимир Александрович

528 группа

17.10.2020

Описание задачи и программной реализации

Краткое описание задания

Целью задания является построения портера разреженной матрицы на основе представления неструктурированной сетки, построение СЛАУ на основе полученного портрета и решения СЛАУ.

Краткое описание программной реализации

В ходе разработки программы были реализованы следующие методы:

- *void createNodesOfGraph* – метод генерации портрета матрицы. Присутствует распараллеливание. Аргументами метода являются:
  - *int Nx* – ширина сетки;
  - *int Ny* – высота сетки;
  - *int k1* – количество несоединенных вершин графа;
  - *int k2* – количество соединенных вершин графа;
  - *map<int, vector<int>> resultGraph* – представление графа в виде справочника. Для каждого элемента по его индексу можно получить список его соседей;
  - *vector<int> IA, JA* – портрет матрицы;
  - *int sizeOfResultGraph* – количество вершин.
- *string vectorToString* – метод выводит на печать в консоль структуру типа vector<int>;
- *vector<string> createAdjacencyList* – метод выводит в консоль список смежности для каждой вершины;
- *void printResult* – печать результатов полученного портрета;
- *void makeSLAE* – метод построения СЛАУ на основе ранее созданного портера матрицы. Выполняется расчет ненулевых коэффициентов и вектора правой части. Присутствует распараллеливание. Аргументы:
  - *vector<int> IA, vector<int> JA* – портрет матрицы;
  - *vector<double> A* – ненулевые коэффициенты матрицы;
  - *vector<double> b* – вектор правой части;
  - *int countOfNodes* – количество вершин графа.
- *void printSLAE* – печать ненулевых коэффициентов каждой строки и значения вектора правой части;
- *double scalar* – принимает на вход две структуры типа vector<double> и рассчитывает их скалярное произведение (скалярное произведение векторов). Присутствует распараллеливание. Аргументы:
  - *vector<double> x1, x2* – вектора для перемножения;
  - *double allTime* – общее время, потраченное на текущий метод;
  - *double countOfCalls* – количество раз, сколько вызывался текущий метод.
- *double normalizeVector* – принимает на вход один вектор и возвращает его норму. Присутствует распараллеливание.
- *vector<double> spMV* – реализация матрично-векторного произведения. Присутствует распараллеливание. Список аргументов:
  - *int countOfNodes* – количество вершин графа;
  - *vector<int> IA, JA* – портрет матрицы;

    - *vector<double> A* – ненулевые коэффициенты матрицы;
    - *vector<double> x* – вектор, на который умножать;
    - *double allTime* – общее время, потраченное на текущий метод;
    - *double countOfCalls* – количество раз, сколько вызывался текущий метод.
- *vector<double> linearCombination* – реализация линейной комбинации двух векторов. Присутствует распараллеливание. Содержит следующие аргументы:
    - *vector<double> x1, x2* – вектора для кобинации;
    - *double a1, a2* – коэффициенты линейной комбинации;
    - *double allTime* – общее время, потраченное на текущий метод;
    - *double countOfCalls* – количество раз, сколько вызывался текущий метод.
- *void createMatrixFromA* – создание матрицы *M*, состоящей только из элементов главной диагонали матрицы *A*. Присутствует распараллеливание.
- *void reverseMatrix* – возведение матрицы в степень (-1). Получение обратной матрицы иными словами.
- *void solveSLAE* – реализация алгоритма решения СЛАУ итерационным методом с использованием метода сопряженных градиентов с предобуславливателем Якоби. Имеет следующие агрументы:
    - *vector<int> IA, JA* – портрет матрицы;
    - *vector<double> A* – ненулевые коэффициенты матрицы;
    - *vector<double> b* – вектор правой части;
    - *int countOfNodes* – количество вершин графа;
    - *double tol* – требуемая точность решения;
    - *vector< vector<double>> x* – вектор векторов решения СЛАУ на каждой итерации;
    - *vector< vector<double>> r*;
    - *int n* – количество итераций до получения решения;
    - *double res* – L2 норма невязки;
    - *vector<double> xRes* – вектор *x*, при котором удалось получить решение необходимой точности.
- *void printSolveVector* – печать вектора-решения СЛАУ *x*;
- *void doTask* – метод, выполняющий этап за этапом. Предполагалось, что можно вызвать несколько таких друг за другом с разным количеством потоков для наглядного представления преимуществ распараллеливания;
- *int main* – входная точка программы с инициализацией всех необходимых переменных.

Параметры запуска программы

Программа запускается со следующими параметрами:

1. *Nx* – ширина сетки. Целое число.
2. *Ny* – высота сетки. Целое число.
3. *k1* – количество несоединенных вершин графа. Целое число.
4. *k2* – количество соединенных вершин графа. Целое число.
5. *T* – количество нитей (поток), которые участвуют в распараллеленных участках программы. Целое число.
6. *tol* – необходимая точность решение СЛАУ. Число с плавающей точкой.

7. *isPrint* – необходимость печати детальной информации (таблица смежности, портрет матрицы, вектор решения СЛАУ). 0 – печать отключена, 1 – печать включена.

Опробованные методы оптимизации

В ходе выполнения практической работы возникали трудности с потреблением памяти и времени работы. Возникали проблемы, т. к. изначально вершина графа являлась объектов, хранящей вектор своих соседей (также объекты). В итоге программа отказывалась работать на размерах сетки более 15*15. Было принято решение реорганизовать вектор соседей вершины. Вместо объекта стал хранить всего лишь его индекс. Это позволило запускать программу на большем размере сетки. Заметив такой прирост производительности, убрал совсем объекты из программы, заменив на *map<int, vector<int>>*. Это также улучшило потребление памяти и скорость работы приложения. Детальнее рефакторинг кода можете наблюдать в репозитории https://github.com/vlazarew/Parallel_Lab_1_Graph.

Исследование производительности

Характеристики вычислительной системы

ЭВМ 1 (ПК):

- Процессор – Intel Core i7-7700K;
- Количество ядер – 4;
- Базовая тактовая частота – 4,2 GHz;
- Максимальная тактовая частота – 4,5 GHz;
- GFLOPS – 249,6;
- Кеш-память – 8 MB;
- Частота системной шины – 8 GT/S;
- BW – 2666 * 1* 8 = 21 Гб/с.

ЭВМ 2 (Ноутбук):

- Процессор – Intel Core i5-8520U;
- Количество ядер – 4;
- Базовая тактовая частота – 1,6 GHz;
- Максимальная тактовая частота – 3,4 GHz;
- GFLOPS – 163,2;
- Кеш-память – 6 MB;
- Частота системной шины – 4 GT/S;
- BW – 1866 * 1* 8 = 15 Гб/с.

Описание параметров компиляции:

Используется компилятор *g++*, проект собирается под стандарт *C++ 17* с флагами *-O3*, *-fopenmp* и *-g*

Результаты измерения производительности

Последовательная производительность

Таблица для ЭВМ 1

Статистика ПО/Количество вершин N	**1000**	**10000**	**100000**	**10000000**
Первый этап (генерирование портрета)				
Время выполнения, с	0,241211	0,396918	4,39825	1324,14
Потребляемая память, МВ	1,2	14	156	1400
Второй этап (построение СЛАУ)				
Время выполнения, с	0,000392	0,005259	0,04949	0,602284
Третий этап (решение СЛАУ)				
Время выполнения, с	0,060829	0,313397	2,57477	31,5652
Скалярное произведение векторов				
Среднее время выполнения, с	$3,4 * 10^{-5}$	0,0004	0,00347	0,0342
TBP (AI = 1/8)	2,625 GLOPS, 1,4% TPP	2,625 GLOPS, 1,4% TPP	2,625 GLOPS, 1,4% TPP	2,625 GLOPS, 1,4% TPP
Линейная комбинация				
Среднее время выполнения, с	$5,9 * 10^{-6}$	0,00061	0,00576	0,05517
TBP (AI = 1/6)	3,5 GLOPS, 1,05% TPP	14	156	1400
Матрично-векторное произведение				
Среднее время выполнения, с	0,000291	0,003657	0,02956	0,330362
TBP (AI = 1/12)	1,75 GLOPS, 0,7% TPP	1,75 GLOPS, 0,7% TPP	1,75 GLOPS, 0,7% TPP	1,75 GLOPS, 0,7% TPP

Таблица для ЭВМ 2

Статистика ПО/Количество вершин N	**1000**	**10000**	**100000**	**10000000**
Первый этап (генерирование портрета)				
Время выполнения, с	0,03614	0,545809	6,56015	2052,51
Потребляемая память, МВ	1,2	14	156	1400
Второй этап (построение СЛАУ)				
Время выполнения, с	0,001256	0,008035	0,081523	0,866924
Третий этап (решение СЛАУ)				
Время выполнения, с	0,175656	0,533167	4,56304	49,8717
Скалярное произведение векторов				
Среднее время выполнения, с	$7{,}58\ 10^{-5}$	0,000552	0,005974	0,055931
TBP (AI = 1/8)	1,875 GLOPS, 1,14% TPP	1,875 GLOPS, 1,14% TPP	1,875 GLOPS, 1,14% TPP	1,875 GLOPS, 1,14% TPP
Линейная комбинация				
Среднее время выполнения, с	0,0001188	0,000826	0,007924	0,07819
TBP (AI = 1/6)	2,5 GLOPS, 1,53% TPP	2,5 GLOPS, 1,53% TPP	2,5 GLOPS, 1,53% TPP	2,5 GLOPS, 1,53% TPP
Матрично-векторное произведение				
Среднее время выполнения, с	0,000632	0,00475	0,043093	0,474737
TBP (AI = 1/12)	1,25 GLOPS, 0,76% TPP	1,25 GLOPS, 0,76% TPP	1,25 GLOPS, 0,76% TPP	1,25 GLOPS, 0,76% TPP

Параллельное ускорение

Таблица для ЭВМ 1

Статистика ПО/Количество вершин N	**1000**	**10000**	**100000**	**10000000**
Первый этап (генерирование портрета)				
T = 2, Время выполнения, с	0,0213872	0,280413	3,03506	811,157
T = 4. Время выполнения, с	0,023012	0,253521	2,44427	419,272
T = 8. Время выполнения, с	0,0224873	0,224348	2,24453	293,176
Второй этап (построение СЛАУ)				
T = 2, Время выполнения, с	0,0003167	0,003707	0,0277	0,291634
T = 4. Время выполнения, с	0,0002901	0,00218	0,0214	0,200149
T = 8. Время выполнения, с	0,00021	0,008823	0,01828	0,11911
Третий этап (решение СЛАУ)				
T = 2, Время выполнения, с	0,0687075	0,327522	1,666	26,7714
T = 4. Время выполнения, с	0,097133	0,267817	1,23658	12,795
T = 8. Время выполнения, с	0,094914	0,208724	1,29874	11,8201
Скалярное произведение векторов				
T = 2, Время выполнения, с	$3{,}39 * 10^{-5}$	0,000402	0,0031	0,029667
T = 4. Время выполнения, с	$3{,}8 * 10^{-5}$	0,00031	0,00234	0,02383
T = 8. Время выполнения, с	$3{,}42 * 10^{-5}$	0,00044	0,00229	0,02089
Линейная комбинация				
T = 2, Время выполнения, с	$3{,}93 * 10^{-5}$	0,000467	0,0032	0,03187
T = 4. Время выполнения, с	$4{,}17 * 10^{-5}$	0,000455	0,0024	0,02431
T = 8. Время выполнения, с	$5{,}57 * 10^{-5}$	0,000351	0,0024	0,02341
Матрично-векторное произведение				
T = 2, Время выполнения, с	0,000193	0,002478	0,01704	0,173614
T = 4. Время выполнения, с	0,000193	0,00173	0,00113	0,125424
T = 8. Время выполнения, с	0,000154	0,00165	0,00111	0,09969

Таблица для ЭВМ 2

Статистика ПО/Количество вершин N	**1000**	**10000**	**100000**	**10000000**
Первый этап (генерирование портрета)				
T = 2, Время выполнения, с	0,0029218	0,028688	0,388148	218,914
T = 4. Время выполнения, с	0,003304	0,0384948	0,344301	157,751
T = 8. Время выполнения, с	0,011463	0,0354938	0,271978	117,057
Второй этап (построение СЛАУ)				
T = 2, Время выполнения, с	0,0005282	0,000941	0,016486	0,111489
T = 4. Время выполнения, с	$6{,}1 * 10^{-5}$	0,0016843	0,00764	0,06805
T = 8. Время выполнения, с	$6{,}4 * 10^{-5}$	0,0008323	0,008806	0,08115
Третий этап (решение СЛАУ)				
T = 2, Время выполнения, с	0,0977914	0,111851	0,49237	6,23203
T = 4. Время выполнения, с	0,08889	0,165213	0,47957	2,89067
T = 8. Время выполнения, с	0,07857	0,127696	0,509148	2,5564
Скалярное произведение векторов				
T = 2, Время выполнения, с	$3{,}37 * 10^{-6}$	$2{,}43*10^{-5}$	0,000327	0,002715
T = 4. Время выполнения, с	$4{,}9 * 10^{-6}$	$3{,}24*10^{-5}$	0,000272	0,002377
T = 8. Время выполнения, с	$1{,}77 * 10^{-5}$	$3{,}87*10^{-5}$	0,000271	0,002196
Линейная комбинация				
T = 2, Время выполнения, с	$9{,}53 * 10^{-6}$	$7{,}3 * 10^{-5}$	0,00105	0,01597
T = 4. Время выполнения, с	$1{,}08 * 10^{-5}$	$9{,}69*10^{-5}$	0,000969	0,00818
T = 8. Время выполнения, с	$2{,}59 * 10^{-5}$	0,000172	0,001045	0,00695
Матрично-векторное произведение				
T = 2, Время выполнения, с	$1{,}93 * 10^{-5}$	0,000173	0,00218	0,04037
T = 4. Время выполнения, с	$1{,}9 * 10^{-5}$	0,000172	0,001672	0,01561
T = 8. Время выполнения, с	$4{,}38 * 10^{-5}$	0,000223	0,001775	0,014479

Анализ производительности

ЭВМ 1

Статистика ПО/Количество вершин N	**1000**	**10000**	**100000**	**10000000**
Скалярное произведение векторов				
T = 1, GFLOPS (N)	0,0588234	0,05	0,057636	0,058476
T = 2, GFLOPS (N)	0,058997	0,04975	0,064516	0,068576
T = 4, GFLOPS (N)	0,05263	0,064516	0,085468	0,083924
T = 8, GFLOPS (N)	0,0584784	0,045454	0,087336	0,095738
Линейная комбинация				
T = 1, GFLOPS (N)	0,169491	0,016393	0,017361	0,018125
T = 2, GFLOPS (N)	0,025445	0,021413	0,03125	0,031377
T = 4, GFLOPS (N)	0,023980	0,021978	0,04166	0,041135
T = 8, GFLOPS (N)	0,019342	0,02849	0,04166	0,0417161
Матрично-векторное произведение				
T = 1, GFLOPS (N)	0,055092	0,437604	5,412828	48,431817
T = 2, GFLOPS (N)	0,083067	0,645811	9,389859	92,15865
T = 4, GFLOPS (N)	0,08306735	0,925040	14,15957	12,75675
T = 8, GFLOPS (N)	0,1041038	0,969890	14,4147	16,04978
3-ий этап (алгоритм решения)				
T = 1, GFLOPS (N)	0,102420	5,106462	62,14283	320,823
T = 2, GFLOPS (N)	0,233380	4,886236	96,04051	597,654
T = 4, GFLOPS (N)	0,233380	5,975535	129,391	1250,491
T = 8, GFLOPS (N)	0,168942	7,667302	123,199	1352,629

ЭВМ 2

Статистика ПО/Количество вершин N	**1000**	**10000**	**100000**	**10000000**
Скалярное произведение векторов				
T = 1, GFLOPS (N)	1,51515	0,03846	0,033476	0,84388
T = 2, GFLOPS (N)	0,59347	0,823044	0,61162	0,736648
T = 4, GFLOPS (N)	0,408162	0,584792	0,735292	0,84388
T = 8, GFLOPS (N)	0,112394	0,516792	0,738004	0,910746
Линейная комбинация				
T = 1, GFLOPS (N)	0,008417	0,012106	0,012619	0,012789
T = 2, GFLOPS (N)	0,104931	0,136986	0,095238	0,062617
T = 4, GFLOPS (N)	0,092592	0,103199	0,103199	0,122249
T = 8, GFLOPS (N)	0,03861	0,058139	0,095693	0,143884
Матрично-векторное произведение				
T = 1, GFLOPS (N)	0,02536708	0,336909	3,712974	33,702938
T = 2, GFLOPS (N)	0,8306735	9,250404	7,339596	39,63347
T = 4, GFLOPS (N)	0,8437894	9,304186	9,569569	1024,986
T = 8, GFLOPS (N)	0,3660273	7,176322	90,14264	1105,050
3-ий этап (алгоритм решения)				
T = 1, GFLOPS (N)	0,0908189	3,001592	35,0651	506,888
T = 2, GFLOPS (N)	0,1639714	14,30787	324,965	2567,387
T = 4, GFLOPS (N)	0,1803914	9,686586	333,639	5535,061
T = 8, GFLOPS (N)	0,204085	12,532499	314,257	6258,815

Анализ полученных результатов

Исходя из данных, приведенных выше видно, что на 2-ух рабочих станциях ТВР операций dot, axpby, spMV находится в диапазоне от 0,7% до 1,5% ТРР. То есть достигается производительность в районе 1% от пиковой производительности системы. При этом, поскольку обе станции имеют 4 физических ядра и 8 потоков, то наилучшие показатели по скорости выполнения программы достигаются только когда программа запущена с T=8 количеством нитей. Исходя из приведенных выше сведений о производительности программы от разного количество N и T (количества потоков) можно сделать вывод, что при 8-ми потоках достигается максимальная производительность, которая порой упирается в максимальную пропускную способность системы.