Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра суперкомпьютеров и квантовой информатики

Отчёт по заданию 3:

Разработка параллельной версии решения СЛАУ итерационным методом сопряжённых градиентов с помощью OpenMP

Аксой Тевфик Огузхан
323 группа

Постановка задачи

- Генерировать куб и соответстующую разреженную матрицу (матрица A) для куба по формулам из занятия.
- Взять в качестве предусловителя диагоналную матрицу с диагональю из матрицы A
- Написать алгоритм метода сопряжённых градиентов
- Распараллелить вычисления SpMV, axpby и ddot с помощью OpenMP.
- Измерить средние времена выполнений итераций, SpMV, axpby, ddot.
- Запустить программу на вычислительном комплексе IBM Polus с разными размерами вводных данных (стороны куба, количество максимальных итераций и количествами OpenMP нитей). Сделать выводы о времени выполнения.

Кратко алгоритм решения

Выбираем начальное приближение x_0

$r_0 = b - Ax_0$
$convergence = false$
$k = 1$
$\boldsymbol{repeat}$
  $z_k = M^{-1} r_{k-1}$
  $\rho_k = (r_{k-1}, z_k)$
  $\boldsymbol{if}\ k = 1\ \boldsymbol{then}$
    $p_k = z_k$
  $\boldsymbol{else}$
    $\beta_k = \dfrac{\rho_k}{\rho_{k-1}}$
    $p_k = z_k + \beta_k p_{k-1}$
  $\boldsymbol{endif}$
  $q_k = Ap_k$
  $\alpha_k = \dfrac{\rho_k}{(\rho_k, q_k)}$
  $x_k = x_{k-1} + \alpha_k p_k$
  $r_k = r_{k-1} - \alpha_k q_k$
  $\boldsymbol{if}\ (\rho_k < \epsilon)\ \boldsymbol{or}\ (k \geq maxiter)\ \boldsymbol{then}$
    $convergence = true$
  $\boldsymbol{else}$
    $k = k + 1$
  $\boldsymbol{endif}$
$\boldsymbol{until}\ convergence$

В случае предобуславливателя Якоби матрица M – диагональная матрица, с диагональю из матрицы A. Начальное приближение – нулевое.

Компиляция и запуск программы

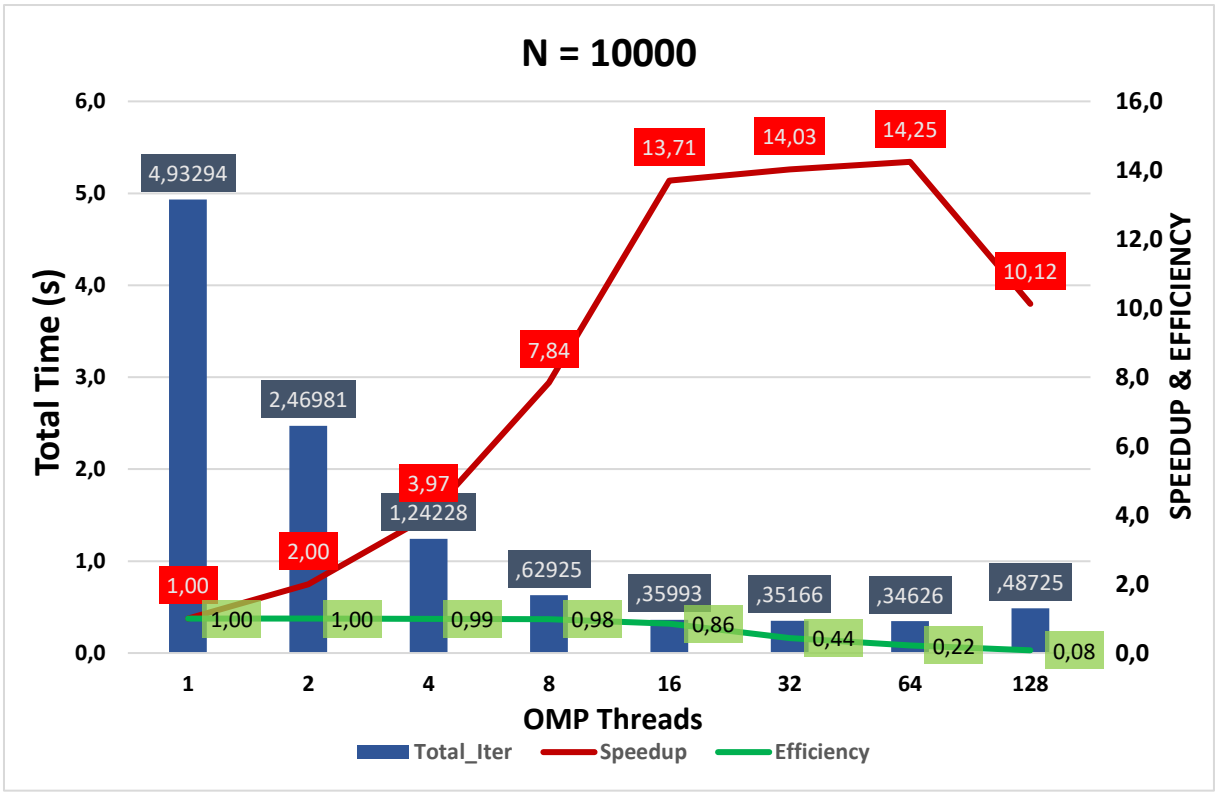
Программа скомпилировалась с помощью команды **xlc++_r main.cpp -o main -std=c++17 -fopenmp -qsmp=omp -qmaxmem=-1 -O2(для оптимизации)** или с помощью **Makefile**, написав команду **“make”** в командной строке. Для размещения программы в очередь, скопировал планировщик **ompjob.lsf** из своего решения первой задачи и ручно поменял нужные параметры. Вызывается команда **bsub < ompjob.lsf** для размещения в очередь**.**

Результаты и выводы

После запуска и обработки программы в Полюсе, файл с результатами ***«result.out»*** на свой компьютер скопировал, дальше для составления графиков и таблицы по результатам, написал **скрипт** на **“Python3”**. Дальше, в Excel составил графики ускорения в зависимости от количества нитей по таблицам и результатам запуска программы.

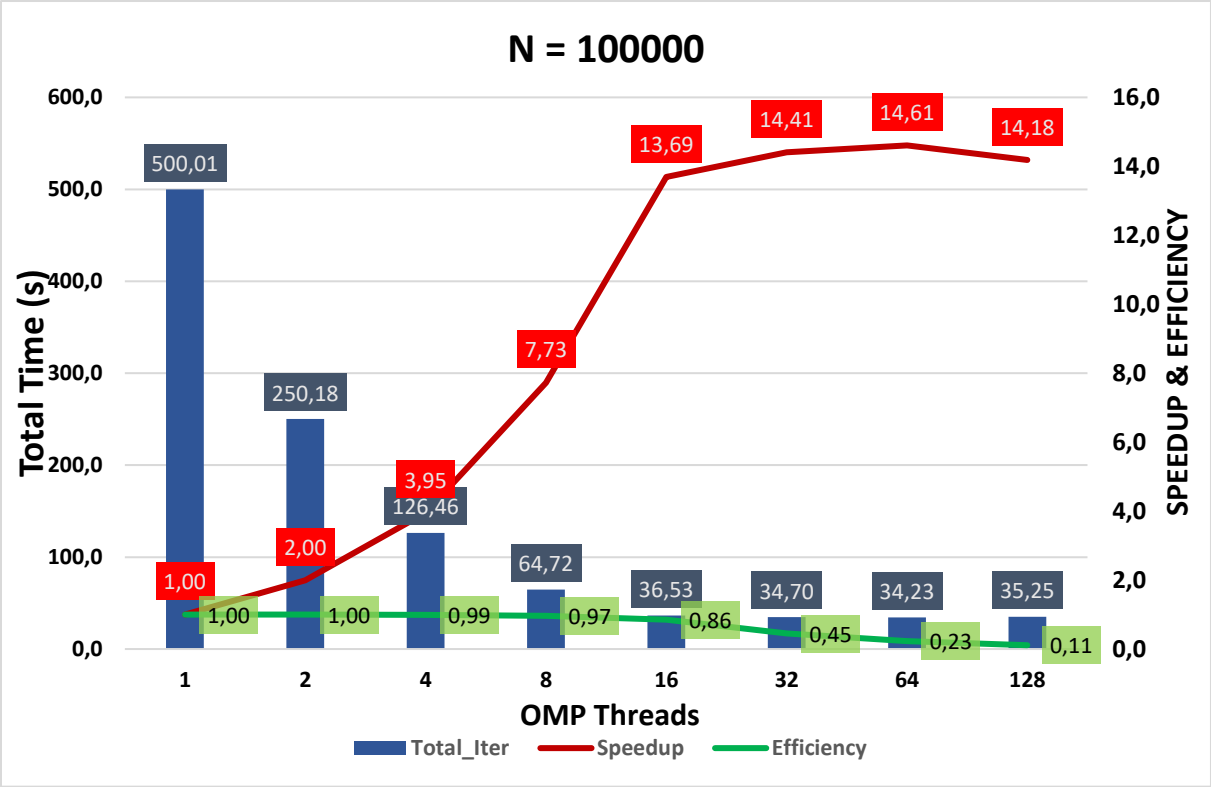
N_Size	Threads	Cube_Gen	Total_Iter	Avg_Iter	Avg_SpMV	Avg_axpby	Avg_ddot
10000	1	0,108	4,933	0,164	0,164	0,000035	0,000036
10000	2	0,108	2,470	0,082	0,082	0,000040	0,000040
10000	4	0,108	1,242	0,041	0,041	0,000047	0,000038
10000	8	0,108	0,629	0,021	0,021	0,000051	0,000041
10000	16	0,110	0,360	0,012	0,012	0,000055	0,000059
10000	32	0,111	0,352	0,012	0,011	0,000068	0,000076
10000	64	0,112	0,346	0,012	0,011	0,000088	0,000160
10000	128	0,112	0,487	0,016	0,014	0,000235	0,001968

N_Size	Threads	Total_Iter	Speedup	Efficiency
10000	1	4,933	1,00	1,00
10000	2	2,470	2,00	1,00
10000	4	1,242	3,97	0,99
10000	8	0,629	7,84	0,98
10000	16	0,360	13,71	0,86
10000	32	0,352	14,03	0,44
10000	64	0,346	14,25	0,22
10000	128	0,487	10,12	0,08



N_Size	Threads	Cube_Gen	Total_Iter	Avg_Iter	Avg_SpMV	Avg_axpby	Avg_ddot
100000	1	10,268	500,012	16,667	16,666	0,00033	0,00031
100000	2	10,158	250,178	8,339	8,338	0,00034	0,00022
100000	4	10,184	126,464	4,215	4,214	0,00038	0,00010
100000	8	10,160	64,720	2,157	2,156	0,00043	0,00009
100000	16	10,182	36,526	1,218	1,216	0,00043	0,00009
100000	32	10,260	34,701	1,157	1,155	0,00051	0,00014
100000	64	10,160	34,231	1,141	1,138	0,00069	0,00022
100000	128	10,118	35,251	1,175	1,164	0,00098	0,00680

N_Size	Threads	Total_Iter	Speedup	Efficiency
100000	1	500,012	1,00	1,00
100000	2	250,178	2,00	1,00
100000	4	126,464	3,95	0,99
100000	8	64,720	7,73	0,97
100000	16	36,526	13,69	0,86
100000	32	34,701	14,41	0,45
100000	64	34,231	14,61	0,23
100000	128	35,251	14,18	0,11



В результатах запуска программ можно сделать выводы, что ускорение ограничивается с использованием 16 нитей, при использовании больше 16 нитей в программе, ускорение в программе не ощущается, а для данных маленького размера программа даже может замедляться.

Соответственно эффективность, для запусков, где больше 16 нитей, намного медленнее становится с каждым повышением количества нитей.

В таблицах видим, что основное полученное ускорение в основном благодаря базовой операции умножение разреженной матрицы на вектор (SpMV).