МГУ им. М. В. Ломоносова, факультет ВМК

Задание 3 Многопоточная реализация солвера CG для СЛАУ с разреженной матрицей, заданной в формате ELLPACK.

Арбузов Николай Романович
группа 323

Постановка задачи

Требуется написать параллельную программу с использованием технологии OpenMP для решения системы линейных уравнений $\boldsymbol{Ax} = \boldsymbol{b}$ методом сопряженных градиентов.

Алгоритм

Алгоритм является итерационным и выполняется до тех пор, пока не будет достигнута необходимая точность или не будет превышено максимально допустимое число итераций.

Алгоритм предобусловленного метода CG имеет следующий вид:

Choose an initial guess x_0;
$r_0 = b - Ax_0$;
convergence = false;
k = 1;
repeat
  $z_k = M^{-1} r_{k-1}$;
  $\rho_k = (r_{k-1}, z_k)$;
  if $k = 1$ **then**
    $p_k = z_k$;
  else
    $\beta_k = \rho_k / \rho_{k-1}$;
    $p_k = z_k + \beta_k p_{k-1}$;
  end if
  $q_k = Ap_k$;
  $\alpha_k = \rho_k / (p_k, q_k)$;
  $x_k = x_{k-1} + \alpha_k p_k$;
  $r_k = r_{k-1} - \alpha_k q_k$;
  if $(\rho_k < \varepsilon)$ **or** $(k \geq maxiter)$ **then**
    convergence = true;
  else
    k = k + 1;
  end if
until convergence

В случае предобуславливателя Якоби матрица M – диагональная матрица,

с диагональю из матрицы A. Начальное приближение – нулевое.

Так как наша матрица по алгоритму построения является матрицей с диагональным преобладанием, было принято решение использовать алгоритм без предобуславливателя.

Компиляция и запуск

Все вычисления производились на машине Polus.

Сама программа написана на языке C++ и состоит из файлов:

- main.cpp
- CG.cpp
- CG.h
- matrix.cpp
- matrix.h

Компилировалась с использованием Makefile:

```
all: main

main: *.cpp *.h
    g++ *.cpp -o prog -std=c++17 -fopenmp

omp_polus: *.cpp *.h
    xlc++ *.cpp -o prog -Wall -std=c++11 -qsmp=omp -fopenmp

clean:
    rm -rf ./prog
```

Запуск производился постановкой в очередь с помощью lsf-файлов вида:

Для 1 и 2 потоков:

```
#BSUB -n 1
#BSUB -W 00:30
#BSUB -o \"./out_files/j/i.out\"
#BSUB -e \"./err_files/j/i.err\"
#BSUB -R \"span[hosts=1]\"
OMP_NUM_THREADS=i ./prog j
```

Для 4, 8, 16 и 32 потоков:

```
#BSUB -W 00:15
#BSUB -o "./out_files/j/i.out"
#BSUB -e "./err_files/j/i.err"
#BSUB -R "affinity[core(M)]"
OMP_NUM_THREADS=i
/polusfs/lsf/openmp/launchOpenMP.py ./prog j
```

Где **i** – количество потоков, на которых будет запускаться программа, **M** – количество ядер (**M = i / 2**), **j** – размер матрицы, на которой будут производиться вычисления.

Результаты

Тесты проводились для кубических матриц с размерами 100x100x100, 200x200x200 и 250x250x250.

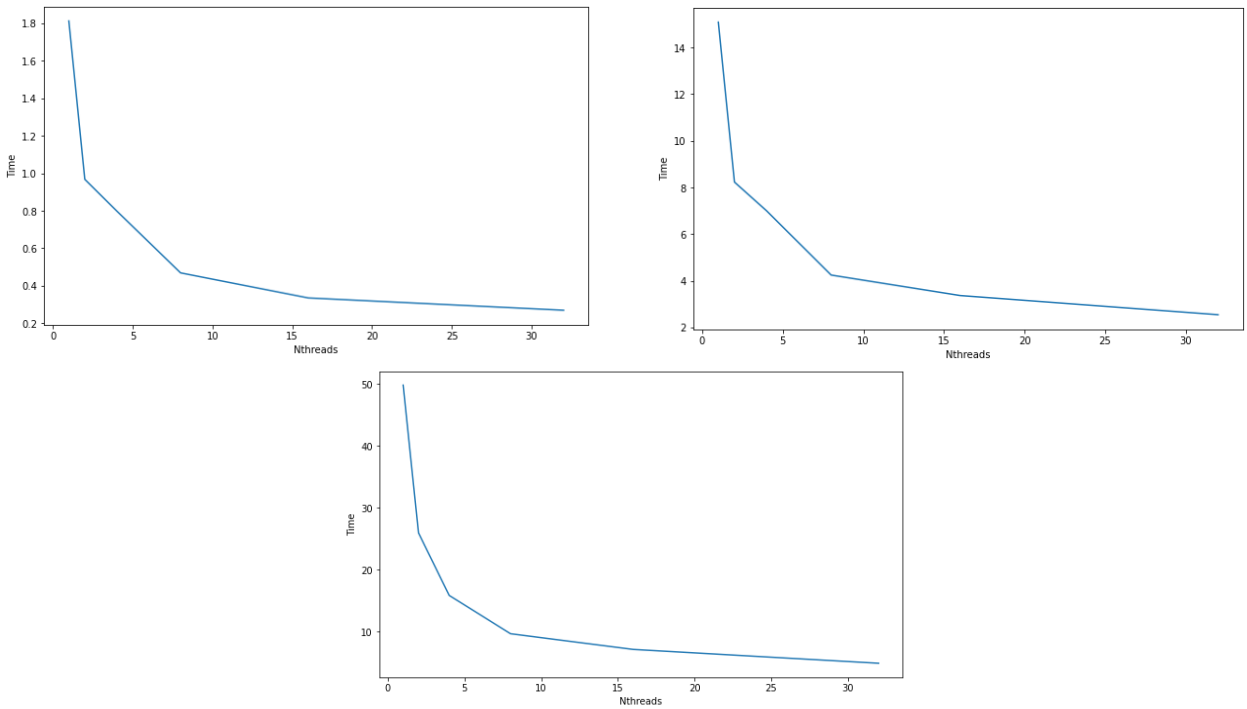

Рисунок 1. На первом графике время для матрицы размерности 100x100x100, на втором – 200x200x200, на третьем – 250x250x250

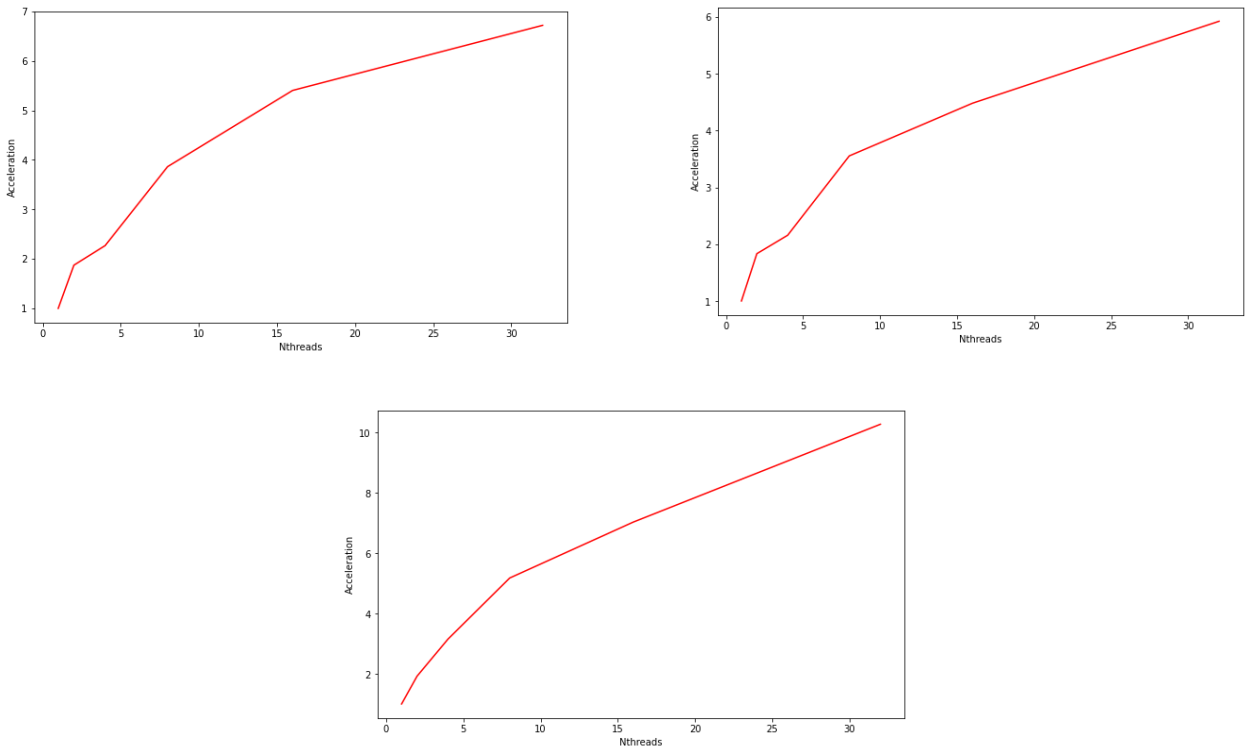

Рисунок 2. Зависимость ускорения от количества потоков. На первом графике для матрицы размерности 100x100x100, на втором – 200x200x200, на третьем – 250x250x250

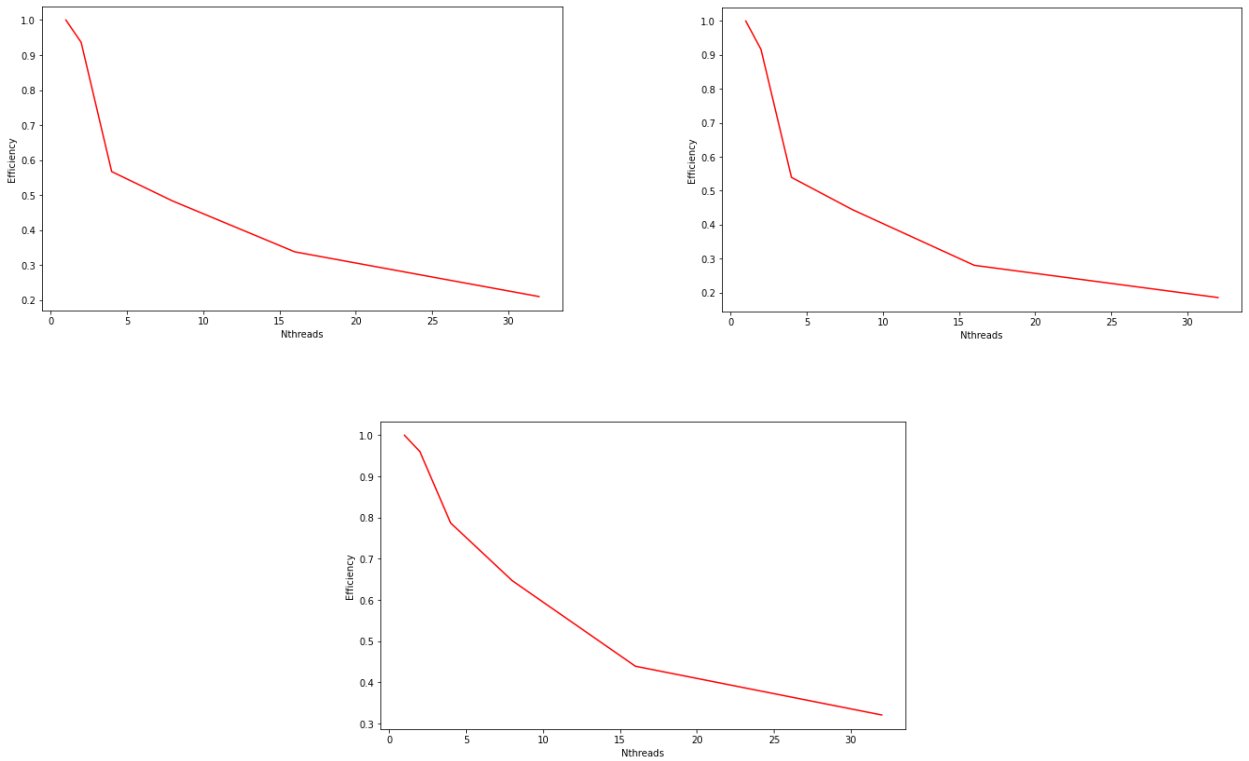

Рисунок 3. Зависимость эффективности от количества потоков. На первом графике для матрицы размерности 100x100x100, на втором – 200x200x200, на третьем – 250x250x2503

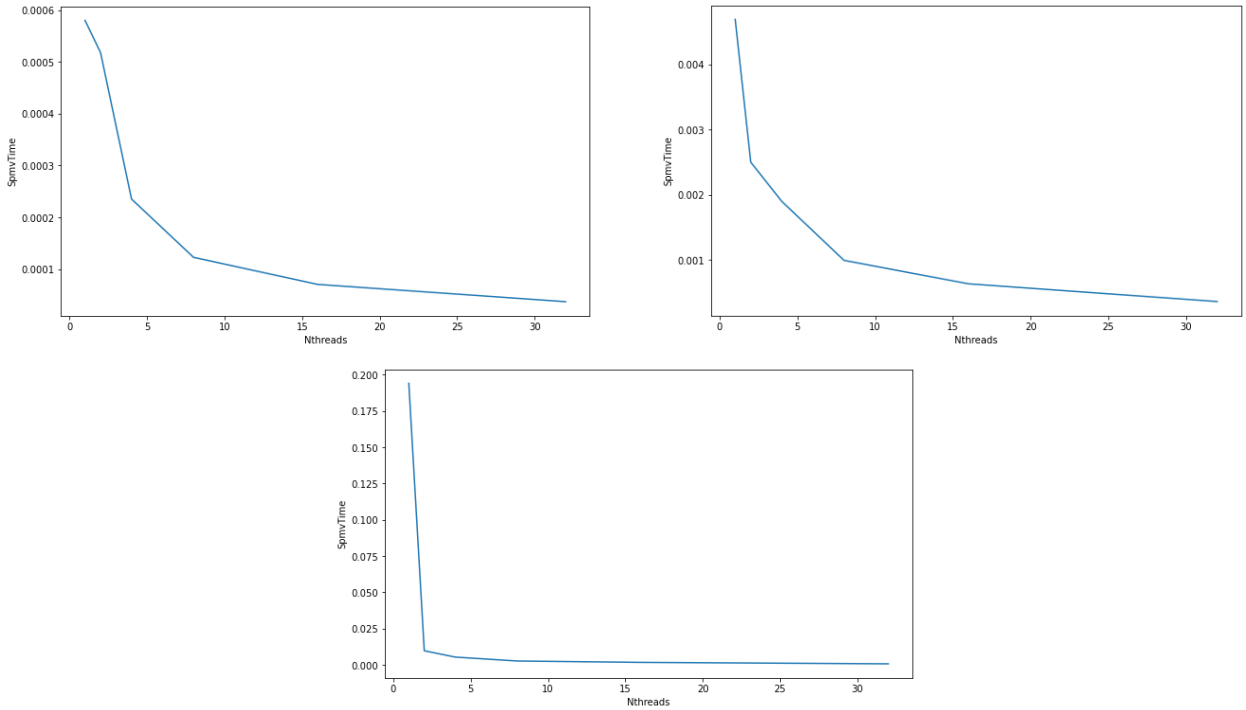

Рисунок 4. Зависимость времени умножении матрицы на вектор от количества потоков.

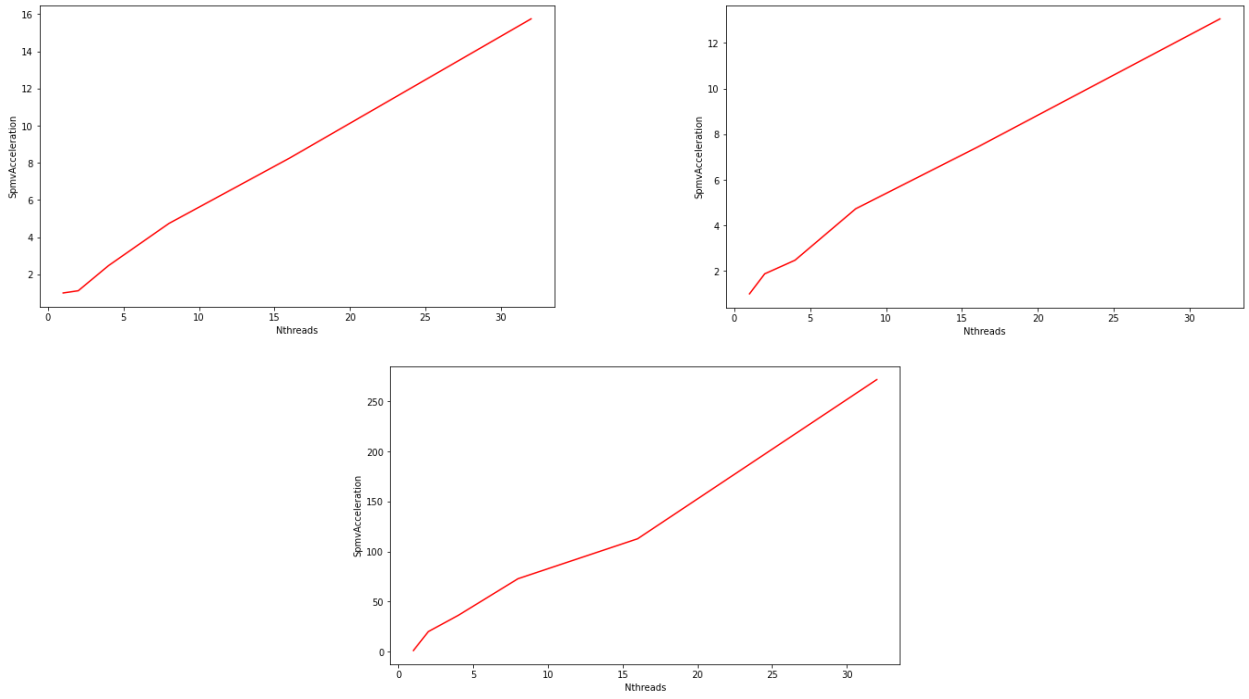

Рисунок 5. Ускорение операции умножения матрицы на вектор.

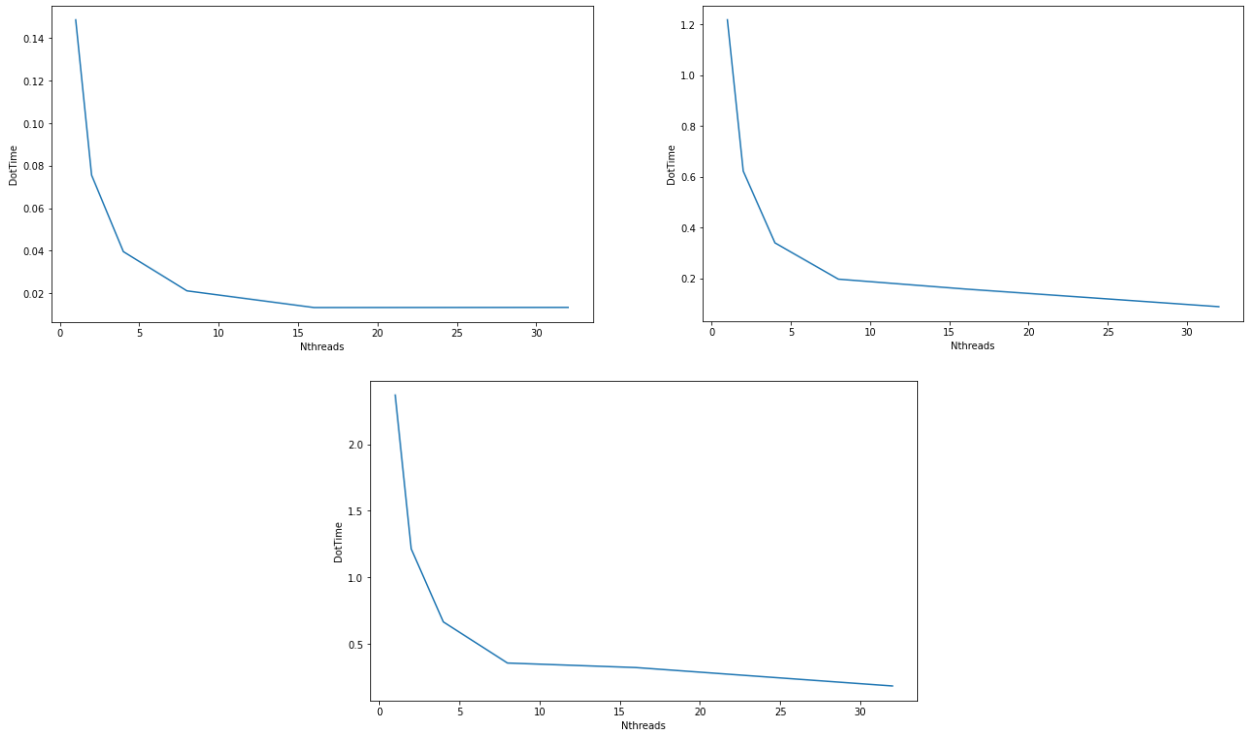

Рисунок 6. Зависимость времени скалярного умножения векторов от количества потоков.

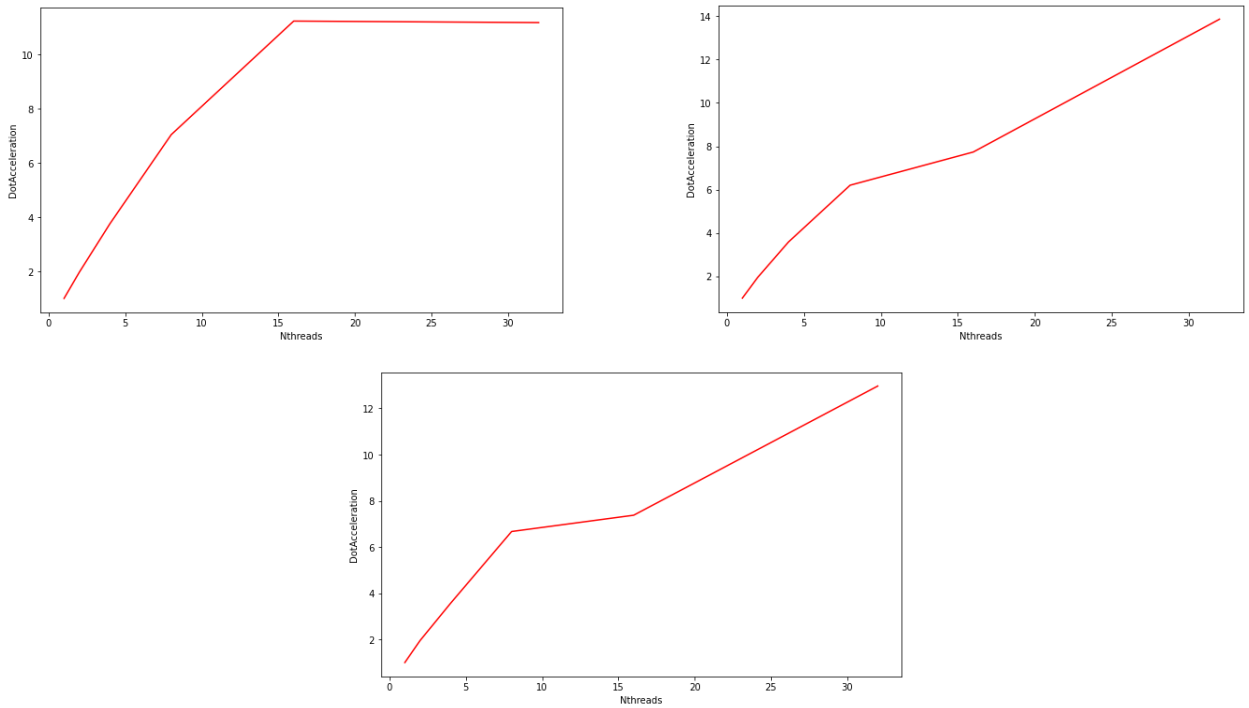

Рисунок 7. Ускорение операции скалярного умножения векторов

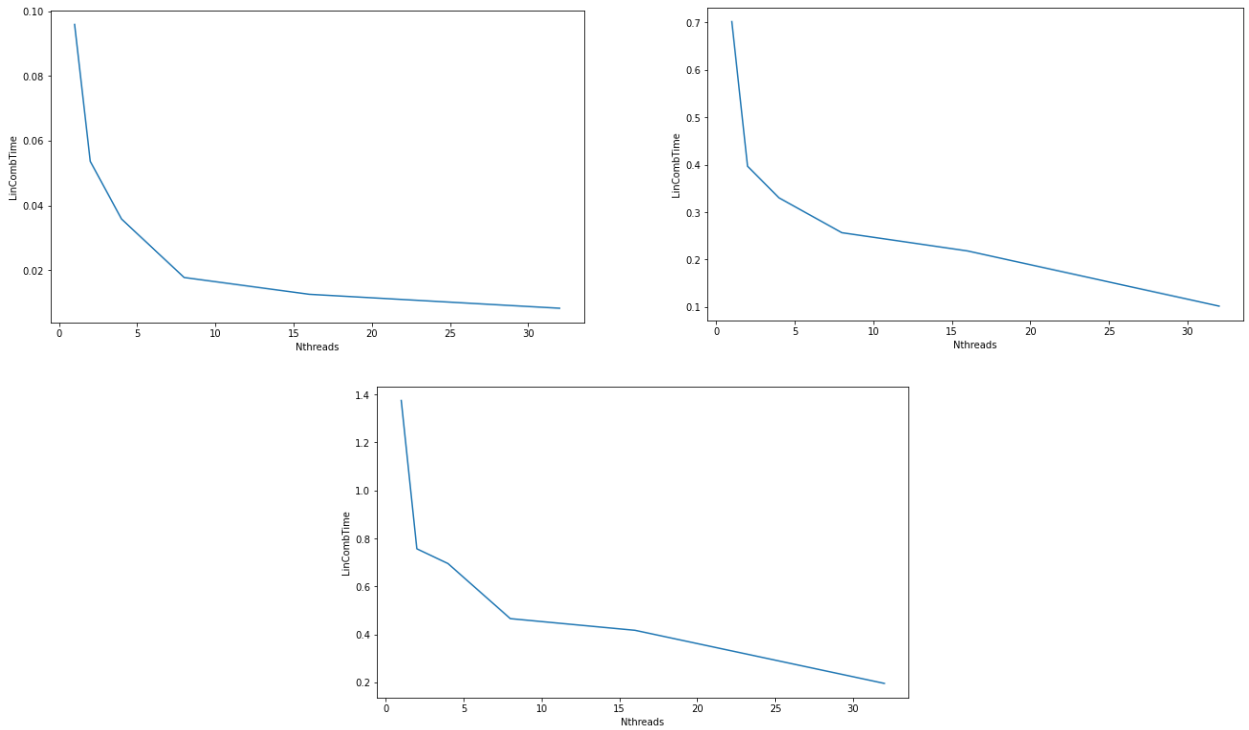

Рисунок 8. Зависимость времени операции линейной комбинации векторов от количества потоков.

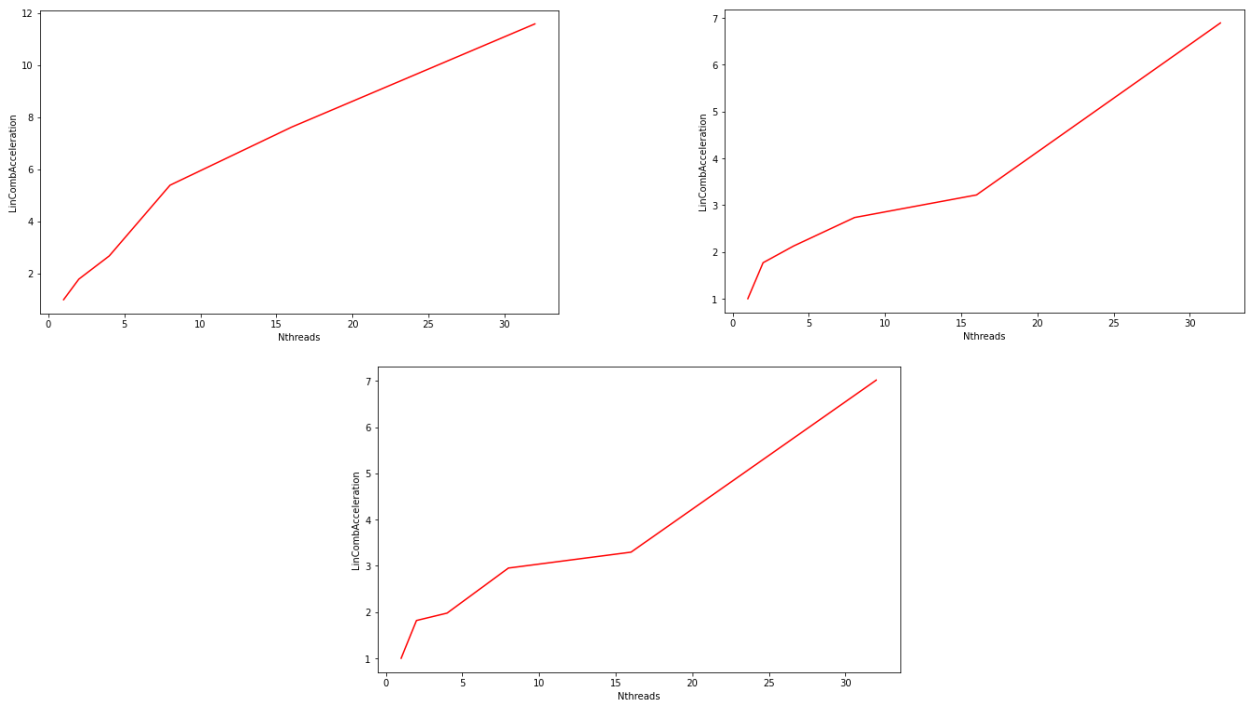

Рисунок 9. Ускорение операции линейной комбинации векторов.

Полная таблица результатов:

MatrixSize	Nthreads	Time	SpmvTime	DotTime	LinCombTime	Residual	Error	Acceleration	Efficiency
1000000	1	1.871196	0.000580	0.148666	0.095963	3.162160e-12	8.182540e-13	1.000000	1.000000
1000000	2	1.129063	0.000518	0.075536	0.053673	3.160670e-12	8.182860e-13	1.657300	0.828650
1000000	4	0.780024	0.000235	0.039547	0.035839	3.160730e-12	8.182650e-13	2.398895	0.599724
1000000	8	0.473429	0.000123	0.021108	0.017792	3.162710e-12	8.185870e-13	3.952430	0.494054
1000000	16	0.350440	0.000070	0.013231	0.012587	3.161430e-12	8.182680e-13	5.339562	0.333723
1000000	32	0.286686	0.000037	0.013298	0.008284	3.161440e-12	8.185060e-13	6.526996	0.203969
8000000	1	15.289183	0.004689	1.217290	0.702066	3.583650e-09	5.125880e-09	1.000000	1.000000
8000000	2	8.818105	0.002498	0.621518	0.396990	3.583650e-09	5.125880e-09	1.733840	0.866920
8000000	4	6.729860	0.001895	0.339078	0.330313	3.583650e-09	5.125880e-09	2.271843	0.567961
8000000	8	4.363742	0.000991	0.196210	0.256755	3.583650e-09	5.125880e-09	3.503686	0.437961
8000000	16	3.516711	0.000631	0.157396	0.218222	3.583650e-09	5.125880e-09	4.347580	0.271724
8000000	32	2.547058	0.000359	0.087848	0.101856	3.583650e-09	5.125880e-09	6.002685	0.187584
15625000	1	50.072063	0.194099	2.369631	1.374479	2.133200e-09	3.037050e-09	1.000000	1.000000
15625000	2	26.721637	0.009727	1.213332	0.756588	2.133200e-09	3.037050e-09	1.873840	0.936920
15625000	4	16.508400	0.005367	0.664562	0.695158	2.133200e-09	3.037050e-09	3.033126	0.758282
15625000	8	9.864239	0.002666	0.355214	0.465688	2.133200e-09	3.037050e-09	5.076120	0.634515
15625000	16	7.323573	0.001722	0.321284	0.417074	2.133200e-09	3.037050e-09	6.837109	0.427319
15625000	32	4.921089	0.000714	0.182746	0.195875	2.133200e-09	3.037050e-09	10.174997	0.317969

MatrixSize	Nthreads	SpmvAcceleration	SpmvEfficiency	DotAcceleration	DotEfficiency	LinCombAcceleration	LinCombEfficiency
1000000	1	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1000000	2	1.119882	0.559941	1.968143	0.984071	1.787911	0.893956
1000000	4	2.468076	0.617019	3.759193	0.939798	2.677589	0.669397
1000000	8	4.736137	0.592017	7.043246	0.880406	5.393549	0.674194
1000000	16	8.255784	0.515987	11.235916	0.702245	7.623821	0.476489
1000000	32	15.745705	0.492053	11.179812	0.349369	11.584451	0.362014
8000000	1	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
8000000	2	1.876823	0.938412	1.958576	0.979288	1.768473	0.884236
8000000	4	2.474174	0.618544	3.590003	0.897501	2.125461	0.531365
8000000	8	4.731245	0.591406	6.204032	0.775504	2.734382	0.341798
8000000	16	7.427712	0.464232	7.733914	0.483370	3.217208	0.201075
8000000	32	13.058307	0.408072	13.856836	0.433026	6.892724	0.215398
15625000	1	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
15625000	2	19.954434	9.977217	1.952994	0.976497	1.816681	0.908340
15625000	4	36.164321	9.041080	3.565706	0.891427	1.977218	0.494305
15625000	8	72.806325	9.100791	6.670994	0.833874	2.951501	0.368938
15625000	16	112.734475	7.045905	7.375495	0.460968	3.295528	0.205971
15625000	32	271.833854	8.494808	12.966793	0.405212	7.017140	0.219286