Отчет по заданию №2 «Исследование численного решения задачи Дирихле для уравнения Пуассона в прямоугольной области» по курсу «Суперкомпьютерное моделирование и технологии».

Асирян Александр, 624 группа. Вариант 2_23.

Математическая постановка задачи

В прямоугольной области $\Pi = [A_1, A_2] \times [B_1, B_2]$ требуется найти дважды гладкую функцию $u = u(x, y)$, удовлетворяющую дифференциальному уравнению:

$$-\Delta u = F(x, y), A_1 < x < A_2, B_1 < y < B_2 \quad (1)$$

и дополнительному условию:

$$u(x, y) = \varphi(x, y) \quad (2)$$

во всех граничных точках (x, y) прямоугольника. Оператор Лапласа Δ определен равенством:

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Вариант 2_23:

- $\Pi = [0, 2] \times [0, 2]$
- $F(x, y) = 2(x^2 + y^2)(1 - 2x^2y^2)\, exp(1 - x^2y^2)$
- $\varphi(x, y) = exp(1 - x^2y^2)$

Численный метод решения задачи

В расчетной области П определяется прямоугольная сетка

$$\bar{\omega}_h = \{(x_i, y_j), i = 0, 1, 2, \dots, N_1, j = 0, 1, 2, \dots, N_2\},$$

где $A_1 = x_0 < x_1 < x_2 < \dots < x_{N_1} = A_2$ – разбиение отрезка $[A_1, A_2]$ оси (ox), $B_1 = y_0 < y_1 < y_2 < \dots < y_{N_2} = B_2$ – разбиение отрезка $[B_1, B_2]$ оси (oy).

Через ω_h обозначим множество внутренних, а через γ_h – множество граничных узлов сетки $\bar{\omega}_h$. Пусть $h_i^{(1)} = x_{i+1} - x_i, i = 0, 1, 2, \dots, N_1 - 1$, $h_j^{(2)} = y_{j+1} - y_j, j = 0, 1, 2, \dots, N_2 - 1$ – переменный шаг сетки по оси абсцисс и ординат соответственно. Средние шаги сетки определяются равенствами:

$$\hbar_i^{(1)} = \frac{h_i^{(1)} + h_{i-1}^{(1)}}{2}, \hbar_j^{(2)} = \frac{h_j^{(2)} + h_{j-1}^{(2)}}{2}.$$

Рассмотрим линейное пространство H функций, заданных на сетке $\bar{\omega}_h$. Будем считать, что в пространстве H задано скалярное произведение и евклидова норма

$$(u,v)=\sum_{i=1}^{N_1-1}\sum_{j=1}^{N_2-1}\hbar_i^{(1)}\hbar_j^{(2)}u_{ij}v_{ij}\,,\|u\|=\sqrt{(u,u)},\quad(3)$$

где $u_{ij}=u(x_i,y_j),v_{ij}=v(x_i,y_j)$ – любые функции из пространства H.

Для аппроксимации уравнения Пуасона (1) воспользуемся пятиточечным разностным оператором Лапласа, который во внутренних узлах сетки определяется равенством:

$$-\Delta_h p_{ij}=\frac{1}{\hbar_i^{(1)}}\left(\frac{p_{ij}-p_{i-1,j}}{h_{i-1}^{(1)}}-\frac{p_{i+1,j}-p_{ij}}{h_i^{(1)}}\right)+\frac{1}{\hbar_j^{(2)}}\left(\frac{p_{ij}-p_{i,j-1}}{h_{j-1}^{(2)}}-\frac{p_{i,j+1}-p_{ij}}{h_j^{(2)}}\right).$$

Здесь предполагается, что функция $p=p(x_i,y_j)$ определена во всех узлах сетки $\bar{\omega}_h$.

Приближенным решением задачи (1), (2) называется функция $p=p(x_i,y_j)$, удовлетворяющая уравнениям

$$-\Delta_h p_{ij}=F\big(x_i,y_j\big),\big(x_i,y_j\big)\in\omega_h,\quad(4)$$
$$p_{ij}=\varphi\big(x_i,y_j\big),\big(x_i,y_j\big)\in\gamma_h.$$

Эти соотношения представляют собой систему линейных алгебраических уравнений с числом уравнений равным числу неизвестных и определяют единственным образом неизвестные значения p_{ij}. Совокупность уравнений (4) называется разностной схемой для задачи (1), (2).

Приближенное решение системы уравнений (4) может быть получено итерационным методом скорейшего спуска. В этом методе начальное приближение

$$p_{ij}^{(0)}=\varphi\big(x_i,y_j\big),\big(x_i,y_j\big)\in\gamma_h,$$

во внутренних узлах сетки $p_{ij}^{(0)}$ – любые числа. Метод является одношаговым. Итерация $p^{(k+1)}$ вычисляется по итерации $p^{(k)}$ согласно равенствам:

$$p_{ij}^{(k+1)}=p_{ij}^{(k)}-\tau_{k+1}r_{ij}^{(k)},$$

где невязка

$$r_{ij}^{(k)}=-\Delta_h p_{ij}-F\big(x_i,y_j\big),\big(x_i,y_j\big)\in\omega_h,\quad(5)$$
$$r_{ij}^{(k)}=0,\big(x_i,y_j\big)\in\gamma_h.$$

Итерационный параметр

$$\tau_{k+1}=\frac{(r^{(k)},r^{(k)})}{(-\Delta_h r^{(k)},r^{(k)})}.$$

Известно, что с увеличением номера итерации k последовательность сеточных функций $p^{(k)}$ сходится к точному решению p задачи (4) по норме пространства H, то есть

$$\left\|p - p^{(k)}\right\|_H \to 0, k \to +\infty.$$

Существенно большей скоростью сходимости обладает метод сопряженных градиентов. Начальное приближение $p^{(0)}$ и первая итерация $p^{(1)}$ вычисляются так же, как и в методе скорейшего спуска. Последующие итерации осуществляются по формулам:

$$p_{ij}^{(k+1)} = p_{ij}^{(k)} - \tau_{k+1} g_{ij}^{(k)}, k = 1, 2, \ldots$$

Здесь

$$\tau_{k+1} = \frac{(r^{(k)}, g^{(k)})}{(-\Delta_h g^{(k)}, g^{(k)})},$$

вектор

$$g_{ij}^{(k)} = r_{ij}^{(k)} - \alpha_k g_{ij}^{(k-1)}, k = 1, 2, \ldots,$$
$$g_{ij}^{(0)} = r_{ij}^{(0)},$$

коэффициент

$$\alpha_{k+1} = \frac{(-\Delta_h r^{(k)}, g^{(k-1)})}{(-\Delta_h g^{(k-1)}, g^{(k-1)})}.$$

Вектор невязки $r^{(k)}$ вычисляется согласно равенствам (5). Итерационный процесс останавливается, как только

$$\left\|p^{(n)} - p^{(n-1)}\right\| < \varepsilon, \quad (6)$$

где ε – заранее выбранное положительное число. В последнем неравенстве, согласно варианту, используется евклидова сеточная норма.

Для аппроксимации дифференциальной задачи используется равномерная прямоугольная сетка:

$$x_i = A_2 \frac{i}{N_1} + A_1 \left(1 - \frac{i}{N_1}\right), i = 0, 1, 2, \ldots, N_1, \quad (8)$$
$$y_j = B_2 \frac{j}{N_2} + B_1 \left(1 - \frac{j}{N_2}\right), j = 0, 1, 2, \ldots, N_2.$$

Приближенное решение разностной схемы (4) следует вычислять методом сопряженных градиентов. Для остановки итерационного процесса предлагается использовать условие (6), положив $\varepsilon = 10^{-4}$.

Постановка задачи

Для функций $F(x, y), \varphi(x, y)$:

- подобрать точное решение задачи Дирихле,

- методом сопряженных градиентов построить приближенное решение на сетке с числом узлов $N_1 = N_2 = 1000$, определить погрешность решения $\psi = \|u(x_i, y_j) - p_{ij}\|$,
- методом сопряженных градиентов построить приближенное решение на сетке с числом узлов $N_1 = N_2 = 2000$ и вновь определить погрешность решения.

Расчеты необходимо проводить на многопроцессорных вычислительных комплексах IBM Blue Gene/P и «Ломоносов», используя различное количество вычислительных узлов, указанное в требованиях к отчету. Для каждого расчета определить его продолжительность и ускорение по сравнению с аналогичным расчетом на одном вычислительном узле. При распараллеливании программы необходимо использовать двумерное разбиение области на подобласти прямоугольной формы, в каждой из которых отношение θ количества узлов по ширине и длине должно удовлетворять неравенствам $0.5 \leq \theta \leq 2$.

Проделанная работа по созданию гибридной реализации MPI/OpenMP

Процессоры разбиваются на сетку $X \times Y$, где

$$X = \begin{cases} \sqrt{n}, \text{если } \log_2 n - \text{четное} \\ \sqrt{\frac{n}{2}}, \quad \text{иначе} \end{cases}, Y = \begin{cases} X, \text{если } \log_2 n - \text{четное} \\ 2 \times X, \quad \text{иначе} \end{cases}.$$

То есть если количество процессоров n является квадратом, то сетка квадратная, иначе она делится на две части по горизонтали, каждая из которых разбивается на квадратную. Далее, с помощью функции *MPI_Cart_create* создается виртуальная топология процессоров. Координаты процессора в топологии и номера соседних процессоров вычисляются с помощью функций *MPI_Cart_coor*ds и *MPI_Cart_shift* соответственно. Например, для $n = 8$ топология будет иметь следующий вид:

0 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	6 (1, 2)	7 (1, 3)

Точки распределяются на процессоры, и каждый процессор обрабатывает только свои точки. Так как количество точек по координате сетки может не делиться нацело на количество процессоров по соответствующей координате в топологии, то лишние точки распределяются на последний процессор по этой координате. Таким образом, все процессоры, кроме нижней и правой границы, будут иметь одинаковое количество обрабатываемых точек.

Например, для $n = 8, N_1 = N_2 = N = 8$ топология будет иметь следующий вид (81 точка):

4×2	4×2	4×2	4×3
5×2	5×2	5×2	5×3

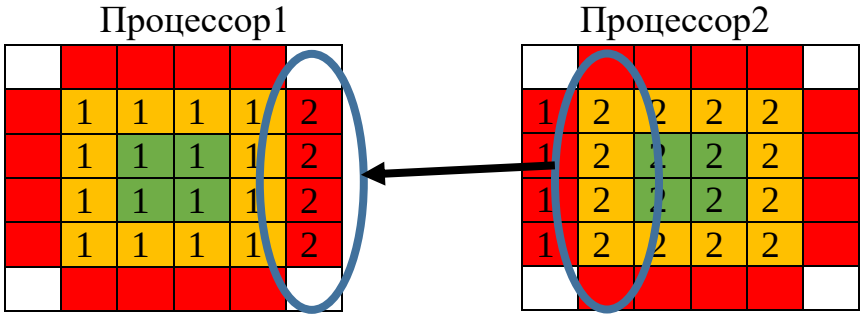
После распределения каждый процессор вычисляет значения переменных из метода для своих точек. Итерации циклов в расчетах распределяются на потоки с помощью директивы OpenMP *parallel for*. Вложенные циклы не распараллеливаются, потому что версия OpenMP не поддерживает директиву *collapse*. Для подсчета частичного скалярного произведения используется директива *reduction* для суммы.

Для вычисления оператора Лапласа требуются данные с точек соседних процессоров. Для этого реализуется обмен между процессорами с помощью команд *MPI_Isend* и *MPI_Irecv* (так как все пересылки являются парными, то можно использовать асинхронные версии). Рассылаются только граничные точки:

N - внутренние точки процессора N

- граничные точки

- точки соседнего процессора



Последовательность обменов следующая:

1. получить точки у соседа снизу
2. получить точки у соседа справа
3. отправить точки соседу сверху
4. отправить точки соседу слева
5. отправить точки соседу снизу
6. отправить точки соседу справа
7. получить точки у соседа сверху
8. получить точки у соседа слева

Для топологии, указанной выше, получаются следующая последовательность шагов, если рассматривать синхронные обмены:

0 (0:0)	1 (0:1)	2 (0:2)	3 (0:3)	4 (1:0)	5 (1:1)	6 (1:2)	7 (1:3)
.	.	.	RD7	.	.	.	SU3
.	.	.	.	.	.	RR7	SL6
.	.	RD6	.	.	.	SU2	.
.	.	RR3	SL2	.	RR6	SL5	.
.	RD5	.	SD7	.	SU1	.	RU3
.	RR2	SL1	RL2	RR5	SL4	SR7	RL6
RD4	.	SD6		SU0	.	RU2	
RR1	SL0	SR3		.	SR6	RL5	
.	SD5	RL1		.	RU1		
.	SR2			SR5	RL4		
SD4	RL0			RU0			
SR1							

XYN – send/receive, направление, номер процессора.

RD7 – получить точки снизу от процессора 7.

Чтобы посчитать итерационный параметр τ^{k+1}, коэффициент α и проверить условие остановки итерационного процесса необходимо вычислить полное скалярное произведение. Для этого, с помощью функции *MPI_Allreduce*, локальные частичные скалярные произведение суммируются и отправляются по всем процессорам.

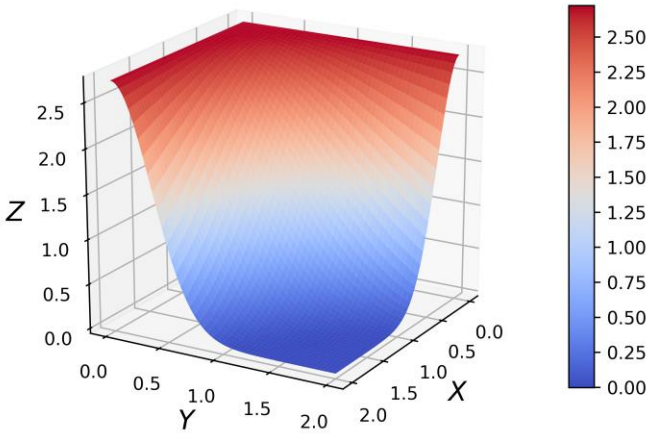
Результаты расчетов

Погрешность

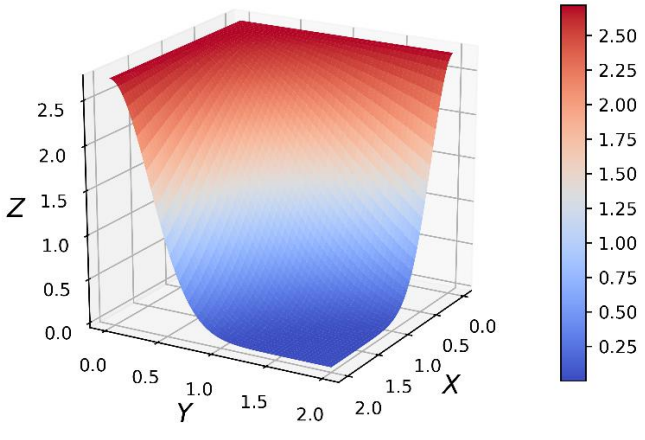
1000 x 1000: 0.011485

2000 x 2000: 0.011669

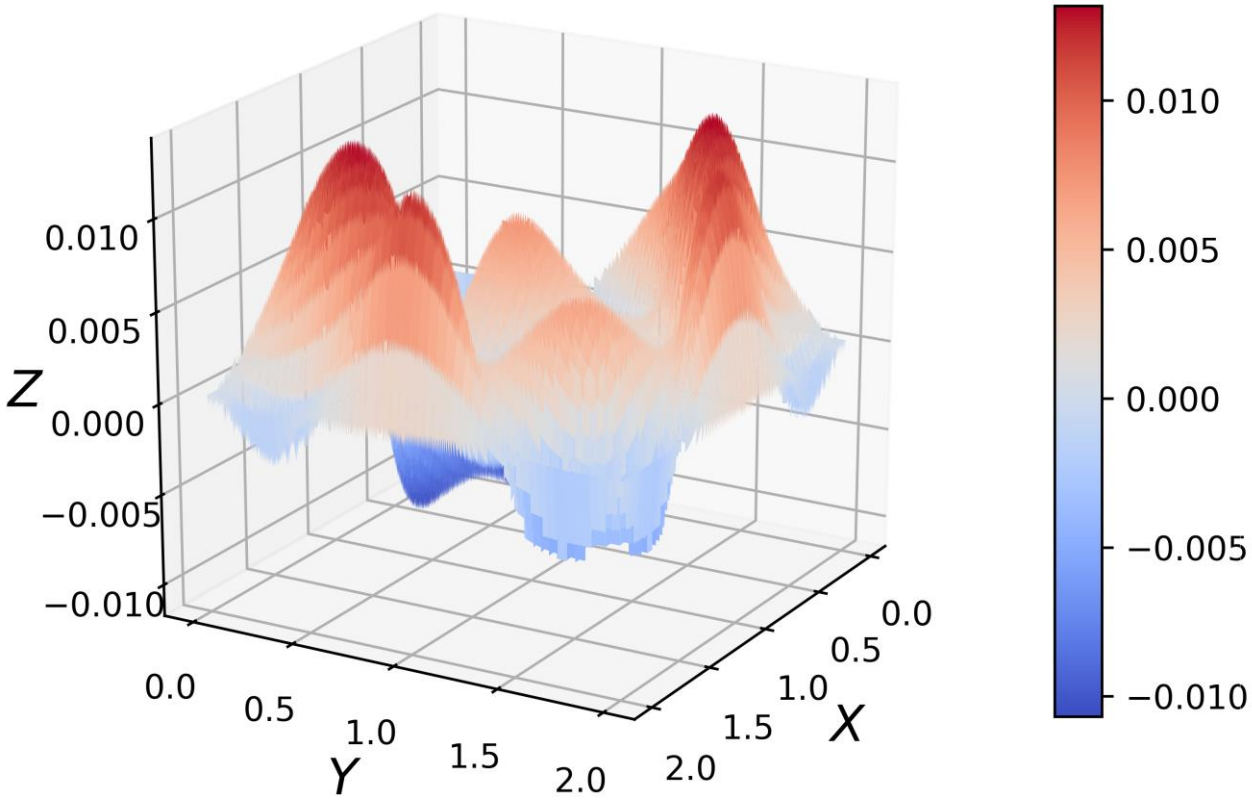
Приближенное решение $\boldsymbol{p_{ij}}$



Точное решение $\boldsymbol{u(x_i, y_j)}$



Погрешность $\boldsymbol{\psi = u(x_i, y_j) - p_{ij}}$



Время и ускорение

Время рассчитывалось как усредненное значение трех запусков.

Ускорение $S = \frac{T_1}{T_p}$.

Blue Gene/P

MPI:

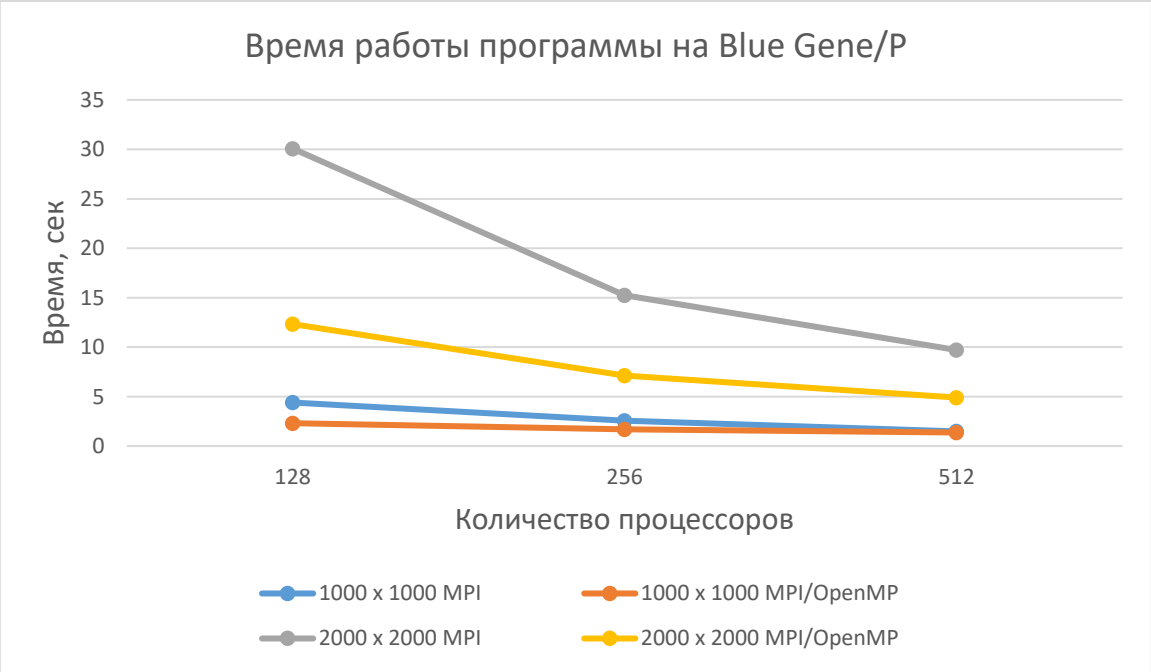
строка компиляции: mpixlcxx -O3 task2.cpp -o task2

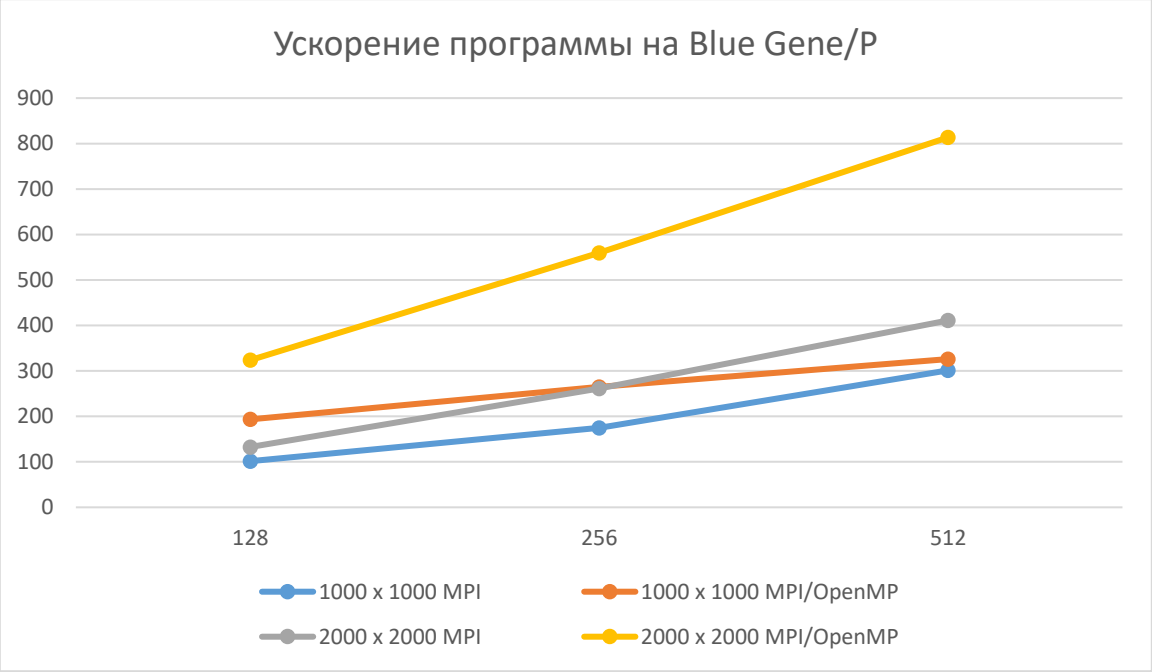
Число процессоров N_p	Число точек сетки N^3	Время решения T	Ускорение S
1	1000 x 1000	446.785	1.000000
128	1000 x 1000	4.400	101.547951
256	1000 x 1000	2.554	174.959449
512	1000 x 1000	1.483	301.209406
1	2000 x 2000	3991.830	1.000000
128	2000 x 2000	30.072	132.742372
256	2000 x 2000	15.265	261.497294
512	2000 x 2000	9.712	411.013497

MPI/OpenMP:

строка компиляции: mpixlcxx_r -O3 -qsmp=omp task2.cpp -o task2_omp

Число процессоров N_p	Число точек сетки N^3	Время решения T	Ускорение S
1	1000 x 1000	152.835	2,923316
128	1000 x 1000	2.309	193,4858
256	1000 x 1000	1.685	265,1884
512	1000 x 1000	1.370	326,0479
1	2000 x 2000	1359.220	2,936854
128	2000 x 2000	12.330	323,7621
256	2000 x 2000	7.126	560,14
512	2000 x 2000	4.907	813,5447

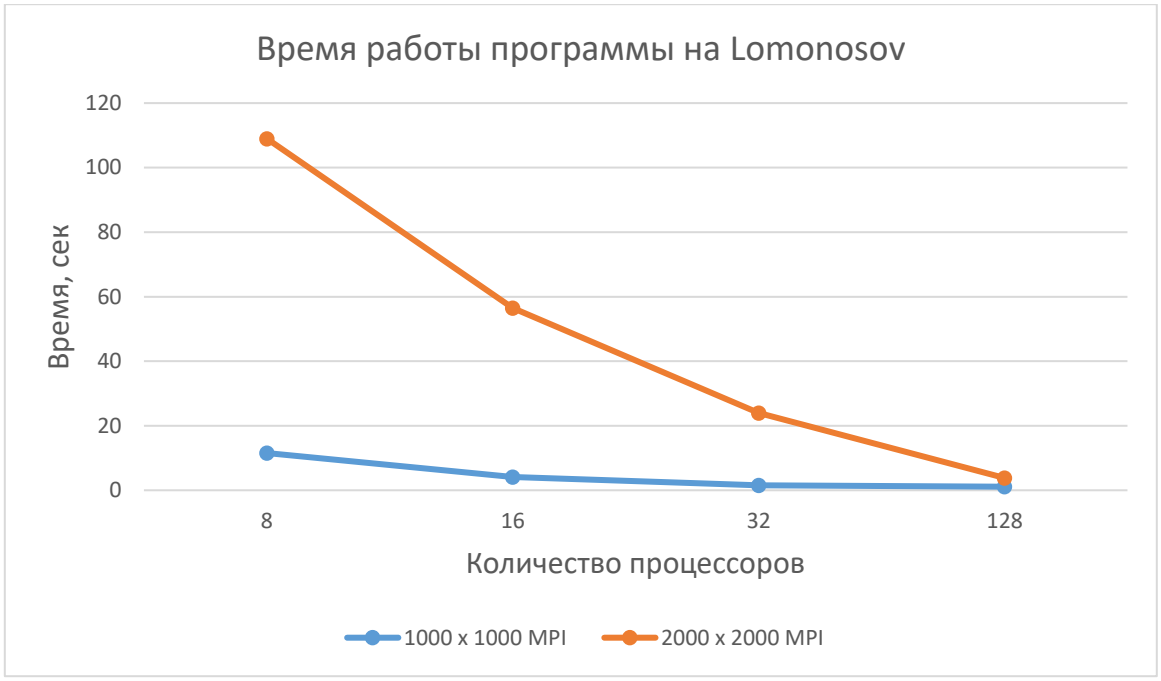


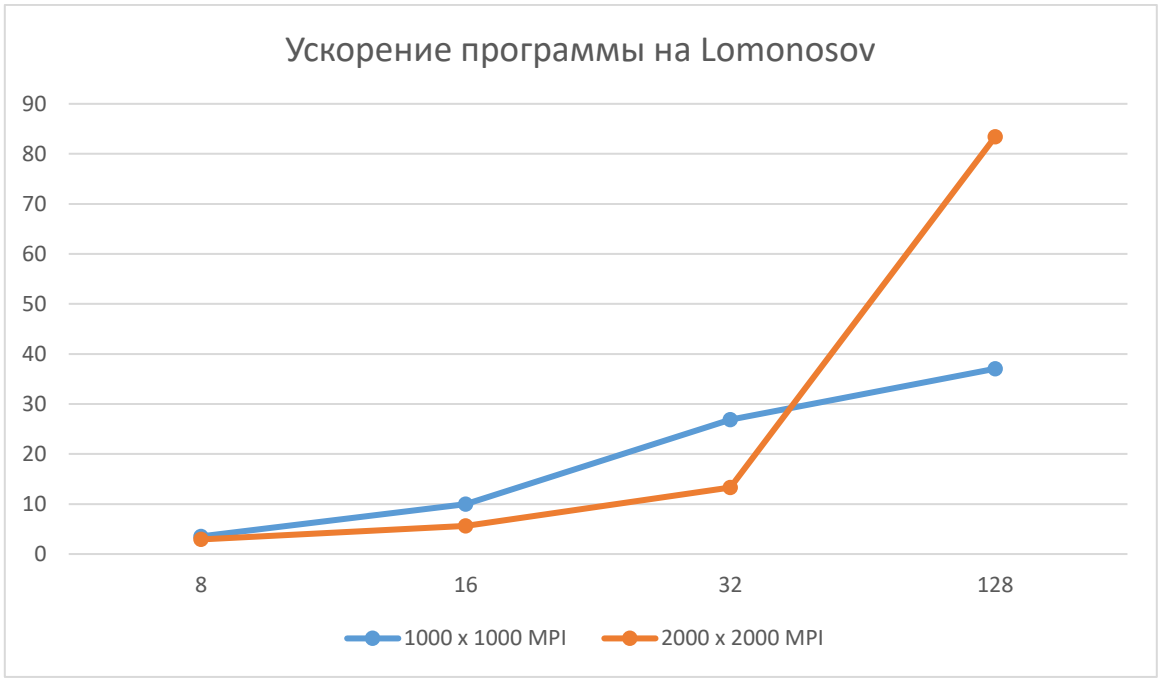


Lomonosov

строка компиляции: mpicxx -O3 task2.cpp -o task2

Число процессоров N_p	Число точек сетки N^3	Время решения T	Ускорение S
1	1000 x 1000	40.801	1.000000
8	1000 x 1000	11.539	3.536073
16	1000 x 1000	4.093	9.967780
32	1000 x 1000	1.518	26.881058
128	1000 x 1000	1.102	37.039683
1	2000 x 2000	318.373	1.000000
8	2000 x 2000	108.983	2.921315
16	2000 x 2000	56.529	5.632065
32	2000 x 2000	23.965	13.285135
128	2000 x 2000	3.818	83.391712


Время работы программы на Lomonosov
120
100
80
60
40
20
0
Время, сек
8
16
32
128
Количество процессоров
1000 x 1000 MPI
2000 x 2000 MPI


Ускорение программы на Lomonosov
90
80
70
60
50
40
30
20
10
0
8
16
32
128
1000 x 1000 MPI
2000 x 2000 MPI