

msu-eps-converted-to.pdf

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА  
Факультет вычислительной математики и кибернетики  
Кафедра системного анализа

Отчет по заданию

**«Реализация параллельного алгоритма  
с использованием технологии MPI»**

*Студент 616 группы*  
М. Н. Преображенский

31 oct 2025

# Содержание

# 1 Введение

Цель работы — решить двумерную задачу Дирихле для уравнения Пуассона в криволинейной области методом фиктивных областей, реализовать вычисления с использованием библиотеки MPI и исследовать масштабируемость по числу процессов на ПВС IBM Polus.

## 2 Математическая постановка задачи

Рассматривается задача Пуассона в криволинейной области  $D \subset \mathbb{R}^2$ , ограниченной контуром  $\gamma$ :

$$-\Delta u = f(x, y), \quad (x, y) \in D, \quad (1)$$

с граничным условием Дирихле первого рода

$$u(x, y) = 0, \quad (x, y) \in \gamma. \quad (2)$$

В данной работе  $f(x, y) \equiv 1$ . Для **варианта 10** область  $D$  задаётся неравенствами

$$D = \{(x, y) : x^2 - 4y^2 > 1, 1 < x < 3\},$$

то есть область ограничена дугой гиперболы и отрезком прямой  $x = 3$ .

## 3 Краткое описание численного метода решения

Далее изложен метод применительно к варианту 10.

### 3.1. Метод фиктивных областей и переформулировка задачи

Пусть  $D \subset \Pi = \{(x, y) : A_1 < x < B_1, A_2 < y < B_2\}$  — охватывающий прямоугольник,  $\widehat{D} = \Pi \setminus D$  — фиктивная область. В  $\Pi$  решается задача

$$-\frac{\partial}{\partial x}(k u_x) - \frac{\partial}{\partial y}(k u_y) = F(x, y), \quad (x, y) \in \Pi \setminus \gamma, \quad u|_{\partial\Pi} = 0, \quad (3)$$

где кусочно-постоянный коэффициент

$$k(x, y) = \begin{cases} 1, & (x, y) \in D, \\ 1/\varepsilon, & (x, y) \in \widehat{D}, \end{cases} \quad \varepsilon = \max(h_x, h_y)^2.$$

Правая часть берётся как  $F \equiv f \equiv 1$  (внутри  $D$ ) и затухает в  $\widehat{D}$  по определению (??).

### 3.2. Сетка и нотация

Покроем  $\Pi$  равномерной сеткой  $\omega_h$  с внутренними узлами  $M \times N$ , шаги  $h_x = \frac{B_1 - A_1}{M+1}$ ,  $h_y = \frac{B_2 - A_2}{N+1}$ . Обозначим полуцелые точки  $x_{i \pm \frac{1}{2}} = x_i \pm \frac{h_x}{2}$ ,  $y_{j \pm \frac{1}{2}} = y_j \pm \frac{h_y}{2}$ .

### 3.3. Разностная схема (5-точечный шаблон с переменными коэффициентами)

Дифференциальный оператор аппроксимируем дивергентной схемой:

$$-\frac{1}{h_x} \left( a_{i+1,j} \frac{w_{i+1,j} - w_{i,j}}{h_x} - a_{i,j} \frac{w_{i,j} - w_{i-1,j}}{h_x} \right) - \frac{1}{h_y} \left( b_{i,j+1} \frac{w_{i,j+1} - w_{i,j}}{h_y} - b_{i,j} \frac{w_{i,j} - w_{i,j-1}}{h_y} \right) = F_{ij}, \quad (4)$$

для  $i = 1, \dots, M$ ,  $j = 1, \dots, N$ , где *граничные коэффициенты* определяются интегралами

$$a_{i,j} = \frac{1}{h_y} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} k(x_{i-\frac{1}{2}}, t) dt, \quad b_{i,j} = \frac{1}{h_x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} k(t, y_{j-\frac{1}{2}}) dt. \quad (5)$$

Правая часть ячейки

$$F_{ij} = \frac{1}{h_x h_y} \iint_{\Pi_{ij}} F(x, y) dx dy, \quad \Pi_{ij} = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]. \quad (6)$$

Границные узлы прямоугольника  $\partial\Pi$  задаются условием  $w_{ij} = 0$  и исключаются из системы. Получаем СЛАУ  $Aw = F$  с самосопряжённым положительно-определенным оператором (см. методичку: доказательство SPD через интегральную форму энергии).

**Практическое вычисление  $a_{i,j}$  и  $b_{i,j}$ .** Так как  $k$  кусочно-постоянна ( $1$  или  $1/\varepsilon$ ), интегралы (?) считаются аналитически как доля длины соответствующей грани внутри  $D$ :

$$a_{i,j} = \frac{\ell_{i,j}^{(D)}}{h_y} \cdot 1 + \left(1 - \frac{\ell_{i,j}^{(D)}}{h_y}\right) \cdot \frac{1}{\varepsilon}, \quad b_{i,j} = \frac{\tilde{\ell}_{i,j}^{(D)}}{h_x} \cdot 1 + \left(1 - \frac{\tilde{\ell}_{i,j}^{(D)}}{h_x}\right) \cdot \frac{1}{\varepsilon}.$$

Здесь  $\ell_{i,j}^{(D)}$  — длина пересечения вертикального отрезка  $[x_{i-\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]$  с  $D$ , а  $\tilde{\ell}_{i,j}^{(D)}$  — длина пересечения горизонтального отрезка  $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}]$  с  $D$ . Для варианта 10 граница задаётся  $|y| < \frac{1}{2}\sqrt{x^2 - 1}$  при  $1 < x < 3$ , поэтому:

$$\ell_{i,j}^{(D)} = \max \left( 0, \min(y_{j+\frac{1}{2}}, \frac{1}{2}\sqrt{x_{i-\frac{1}{2}}^2 - 1}) - \max(y_{j-\frac{1}{2}}, -\frac{1}{2}\sqrt{x_{i-\frac{1}{2}}^2 - 1}) \right),$$

$$\tilde{\ell}_{i,j}^{(D)} = \max \left( 0, \min(x_{i+\frac{1}{2}}, 3) - \max(x_{i-\frac{1}{2}}, \sqrt{1 + 4y_{j-\frac{1}{2}}^2}) \right).$$

То есть мы считаем (5) аналитически, а не через усреднение по узлам.

**Практическое вычисление  $F_{ij}$ .** Если  $\Pi_{ij} \subset D$ , то  $F_{ij} \approx f(x_i, y_j) = 1$ . Если  $\Pi_{ij} \subset \widehat{D}$ , то  $F_{ij} = 0$ . В смешанном случае  $F_{ij} \approx \frac{S_{ij}}{h_x h_y} \cdot 1$ , где  $S_{ij} = \text{mes}(\Pi_{ij} \cap D)$  — площадь пересечения; криволинейную границу внутри ячейки можно линеаризовать (методичка). На практике удобно оценивать  $S_{ij}$  субсемплингом  $q \times q$  (например,  $q = 4$ ).

**Выбор  $\varepsilon$ .** По заданию и методичке берём  $\varepsilon = \max(h_x, h_y)^2$ . Это даёт «жёсткое» подавление фиктивной области без ухудшения обусловленности сверх сеточного уровня.

## 4 Краткое описание реализации MPI-решения

Ниже приведена **строкура кода** и то, как организовано распараллеливание с помощью MPI. Численный метод соответствует схеме (??)–(??).

### 4.1. Основные этапы программы

- Геометрия варианта 10 задаётся функциями `y_cap(x)` и `in_D(x,y)`: область  $D = \{1 < x < 3, |y| < \frac{1}{2}\sqrt{x^2 - 1}\}$ .
- **Декомпозиция по строкам.** После вызова `MPI_Init` размер сетки  $M \times N$  и число процессов `size` известны всем. Каждый процесс по своему номеру `rank` вычисляет локальный диапазон строк  $i \in [i_{\text{start}}, i_{\text{end}}]$  ( поля `i_start`, `local_M`) и использует локальные индексы `ID_local`, `IX_X_local`, `IX_Y_local` для обращения к массивам.
- **Сборка коэффициентов  $a_{i,j}$  и  $b_{i,j}$ .** Для своих строк каждый процесс заполняет массивы `ax` и `by`, соответствующие гранным коэффициентам (??). Длины пересечений граней с областью  $D$  вычисляются аналитически, после чего строится смесь 1 и  $1/\varepsilon$  по формулам из раздела ??.
- **Формирование правой части  $F_{ij}$ .** В локальном прямоугольнике процесс интегрирует правую часть по каждой ячейке с помощью субсемплинга  $q \times q$  (`SS=4`) и записывает значения в вектор `F`.
- **Построение диагонали матрицы.** Массив `A_diag` заполняется по соседним гранным коэффициентам  $a_{i\pm 1,j}$  и  $b_{i,j\pm 1}$  в точном соответствии с дискретным оператором; он используется и в матвекторе, и в диагональном предобуславливателе Якоби.
- **Матрично-векторное произведение.** Лямбда `matvec` реализует применение пяти-точечного оператора к вектору `r` с учётом переменных коэффициентов и граничных условий.
- **Скалярное произведение и нормы.** Лямбда `dot` считает локальную сумму и затем вызывает `MPI_Allreduce`, получая глобальное значение скалярного произведения.

- **Итерационный метод PCG.** В основном цикле реализован предобусловленный метод сопряжённых градиентов:  $u$  — приближение решения,  $r$  — невязка,  $z=D^{-1}r, p^-, Ap^-$ .  $\|r\|/\|b\| \leq 10^{-8}$  или достижение максимального числа итераций.
- Сбор решения. После окончания итераций все локальные блоки вектора решения собираются на нулевом процессе с помощью MPI\_Gather/MPI\_Gatherv, где формируется файл solution.csv с колонками  $(x, y, u)$ .

## 4.2. Где используется MPI

- **Распределение данных по процессам.** Строки сетки  $i = 1, \dots, M$  делятся по процессам почти поровну; каждый процесс хранит только свои строки всех массивов ( $ax, by, F, A_{diag}, u, r, p, Ap$ ).
- **Halo-обмен при матвекторе.** Внутри matvec вызывается лямбда exchange\_halo: соседние по  $i$  процессы асинхронно обмениваются граничными строками вектора (MPI\_Isend/MPI\_Irecv), после чего каждый процесс может корректно использовать значения  $u$  на стыках поддоменов.
- **Глобальные скалярные произведения.** Функция dot использует коллективную операцию MPI\_Allreduce, так что все процессы получают одинаковые значения  $(r, r)$ ,  $(r, z)$ ,  $(p, Ap)$ , необходимые для вычисления коэффициентов  $\alpha$  и  $\beta$  в PCG.
- **Сбор решения и вывод результатов.** Для записи решения в CSV-файл процесс 0 собирает все локальные блоки вектора  $u$  с помощью MPI\_Gatherv, восстанавливая глобальный порядок узлов  $(i, j)$ .
- **Инициализация и завершение.** Перед началом вычислений вызывается MPI\_Init, а после записи результата — MPI\_Finalize, что позволяет запускать программу как в параллельном, так и в последовательном режиме (при  $p = 1$ ).

## 5 Результаты тестирования программы

### 5.1 Сходимость по сетке и корректность

В таблице 1 приведено число итераций и конечная относительная невязка для последовательного запуска (1 процесс) на разных сетках.

### 5.2 Сравнение последовательной и параллельной версий

Сетка  $40 \times 40$ . Величина САО считается между решениями, полученными при  $p=1$  (seq) и  $p>1$  (par).

Таблица 1: Сходимость по сетке ( $p=1$ ): число итераций, конечная относительная невязка и время решения.

Размер сетки $M \times N$	Итерации	$\ r\ /\ b\ $	Время $t$ , мс
$10 \times 10$	29	$5.091 \times 10^{-9}$	0.197
$20 \times 20$	61	$7.575 \times 10^{-9}$	0.819
$40 \times 40$	123	$9.802 \times 10^{-9}$	5.198

Таблица 2: Сетка  $40 \times 40$ : итерации, конечная невязка и время решения при разном числе MPI-процессов.

$p$	Итерации	$\ r\ /\ b\ $	Время $t$ , мс
1	123	$9.801 \times 10^{-9}$	5.198ms
4	123	$9.801 \times 10^{-9}$	3.13ms
16	123	$9.801 \times 10^{-9}$	2.54ms

### 5.3 Визуализация решения

На рис.1-4 показаны карты  $u(x, y)$  на мелкой и крупной сетках.



Рис. 1: Поле распределения потенциала  $u(x, y)$  для размера сетки: (а)  $40 \times 40$



sol2\_1200x800.png

Рис. 2: Поле распределения потенциала  $u(x, y)$  для размера сетки: (с)  $1200 \times 800$

## 6 Анализ ускорения MPI-реализации

### 6.1 Strong scaling: сетка $400 \times 600$

Таблица 3: Ускорение на  $400 \times 600$ .

$p$	Итерации	Время $t$ , с	Ускорение $s$	Эффективность $eff$
1	1520	9.475	1.000	1.000
2	1520	4.740	2.000	1.000
4	1520	3.544	2.674	0.668
8	1520	2.388	3.970	0.496
16	1520	1.413	6.710	0.419
32	1520	1.270	7.460	0.233

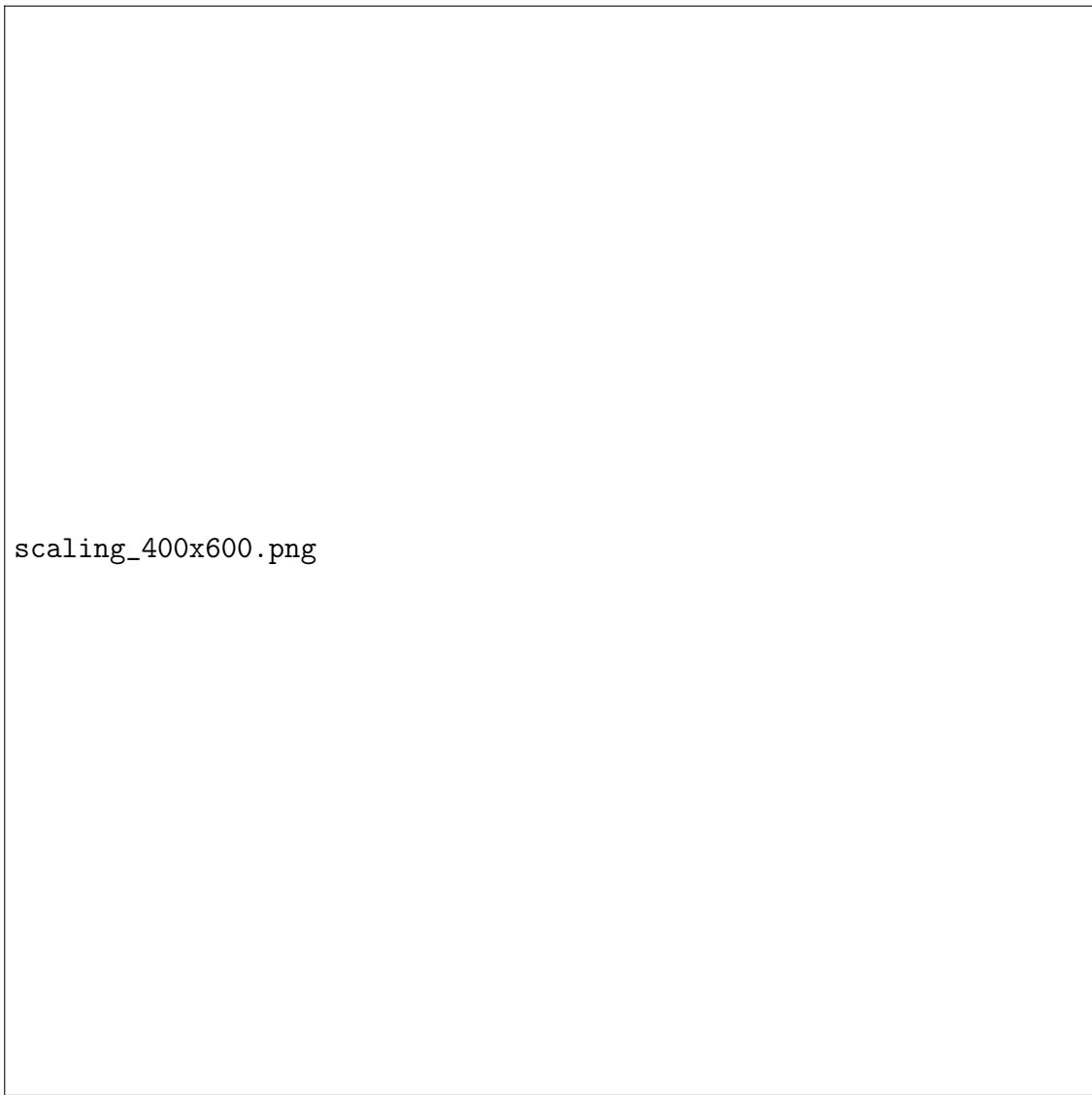


Рис. 3: График ускорения  $S_p$  для сетки  $400 \times 600$  (MPI, strong scaling).

## 6.2 Strong scaling: сетка $800 \times 1200$

Таблица 4: Ускорение на  $800 \times 1200$ .

$p$	Итерации	Время $T_p$ , с	Ускорение $S_p$	Эффективность $E_p$
1	3076	87.900	1.000	1.000
4	3076	22.404	3.924	0.981
8	3076	18.915	4.647	0.581
16	3076	16.916	5.195	0.188
32	3076	14.616	6.015	0.188

scaling\_800x1200.png

Рис. 4: График ускорения  $S_p$  для сетки  $800 \times 1200$  (MPI, strong scaling).

Таблица 5: Ускорение на  $40 \times 40$  (part3).

$p$	Итерации	Время $T_p$ , с	Ускорение $S_p$	Эффективность $E_p$
1	123	0.01376	1.000	1.000
4	123	0.00686	2.006	0.501
16	123	0.00431	3.191	0.199

### 6.3 Strong scaling: сетка $40 \times 40$ (part3)

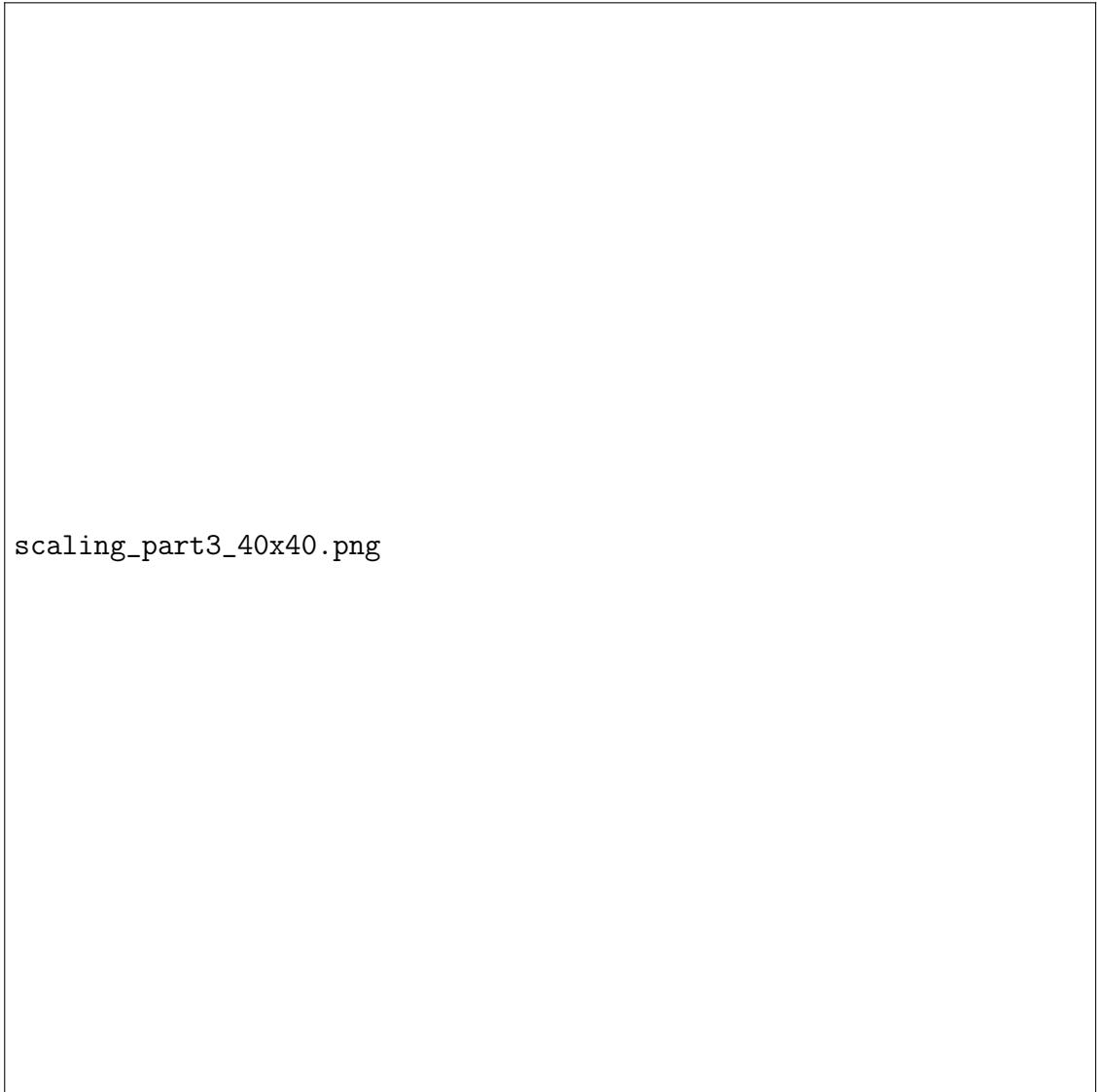


Рис. 5: График ускорения  $S_p$  для сетки  $40 \times 40$  (MPI).

## 7 Заключение

В ходе работы была реализована и исследована MPI-параллельная версия алгоритма решения уравнения Пуассона на прямоугольной сетке методом сопряжённых градиентов (CG/PCG) с методом фиктивных областей. Проведено тестирование на наборах различных размеров сеток и числа MPI-процессов.

Результаты измерений показывают, что реализованная программа демонстрирует почти идеальное ускорение при переходе с одного до двух процессов и далее — выражено сублинейный рост. Для сетки  $400 \times 600$  почти идеальное ускорение достигается при  $p = 2$  ( $S_2 \approx 2.00$ ,  $E_2 \approx 1.00$ ), при  $p = 8$  получаем  $S_8 \approx 3.97$  ( $E_8 \approx 0.50$ ), а максимальное измеренное ускорение составляет  $S_{32} \approx 7.46$  ( $E_{32} \approx 0.23$ ). Для более крупной задачи  $800 \times 1200$  масштабируемость лучше: при  $p = 4$  имеем  $S_4 \approx 3.92$  ( $E_4 \approx 0.98$ ), при  $p = 8$  —  $S_8 \approx 4.65$  ( $E_8 \approx 0.58$ ), а при  $p = 32$  —  $S_{32} \approx 6.01$  ( $E_{32} \approx 0.19$ ), что отражает более выгодное соотношение вычислительной и коммуникационной составляющих для крупной сетки. Для небольшой тестовой задачи  $40 \times 40$  (part3) из-за относительно малой вычислительной нагрузки эффективность ниже, но ускорение при переходе от  $p = 1$  к  $p = 16$  всё же достигает  $S_{16} \approx 3.19$ .

При дальнейшем увеличении числа процессов (от 8 к 16–32) ускорение продолжает расти, но прирост становится существенно менее выраженным: эффективность падает ниже 0.5 для обеих основных сеток. Это объясняется ростом накладных расходов на коммуникации между процессами, конкуренцией за общие ресурсы памяти (memory bandwidth), а также ограничениями кэш-иерархии и подсистемы межсоединений. Таким образом, на используемой системе практически целесообразно использовать диапазон примерно от 4 до 8 процессов, обеспечивающий наилучший баланс между производительностью и эффективностью.

Полученные результаты подтверждают корректность параллельной реализации на MPI и её эффективность при решении задач с большими размерами сеток. Дальнейшее повышение производительности возможно за счёт оптимизации работы с памятью, улучшения схемы распределения нагрузки, использования NUMA-распределения и векторизации вычислений.