ВМК МГУ

Отчет о выполнении задания №3 по курсу «Суперкомпьютерное моделирование и технологии»

Исполнитель:
студент факультета ВМК МГУ
кафедры АСВК группы 620
А.А. Ковальчук

Москва, 2016

Содержание отчета

Математическая постановка задачи

Вариант 10, набор 3, равномерная сетка, максимум-норма.

Задача Дирихле для уравнения Пуассона в прямоугольной области:
В прямоугольной области $\Pi = [-2,2]\times[-2,2]$ требуется найти дважды гладкую функцию $u = u(x,y)$, удовлетворяющую дифференциальному уравнению

$$-\Delta u = (x^2 + y^2)\sin(xy), \qquad -2 < x < 2, -2 < y < 2$$

и дополнительному условию

$$u(x,y) = 1 + \sin(x,y)$$

во всех граничных точках (x,y) прямоугольника.
Оператор Лапласа Δ определен равенством: $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$.

Численные методы решения

Для аппроксимации дифференциальной задачи используется равномерная прямоугольная сетка с максимум-нормой.
Приближенное решение задачи разностной схемы вычисляется методом сопряженных градиентов. Для остановки итерационного процесса используется $\varepsilon = 10^{-4}$ в качестве оценки разности итераций.
При распараллеливании программы используется двумерное разбиение области на подобласти прямоугольной формы, в каждой их которых отношение θ количества узлов по ширине и длине удовлетворяет неравенствам $0.5 \le \theta \le 2$.

Описание гибридной реализации MPI/CUDA

Для решения задачи с использованием технологии MPI рассматриваемая область разбивается на подобласти (число подобластей равно числу процессов). Для этого используется вызов функции *MPI_Cart_create*, которая возвращает новый коммуникатор. Далее, каждый процесс, используя функции *MPI_Cart_coords* и *MPI_Cart_shift* может получить свое положение в сетке процессов и ранки соседних процессов, а также рассчитать тот диапазон точек матрицы, которые ему необходимо рассчитать.

В процессе работы алгоритма процессам необходимо знать значения из областей, рассчитываемых другими процессами. Для этого процессы обмениваются граничными областями с использованием функции *MPI_Sendrecv*. Для пересылки строк создается тип *данных MPI_Type_contiguous*. Для пересылки столбцов используется тип данных *MPI_Type_vector*.

Для подсчета скалярного произведения необходимо вычислять сумму по всей области. Данный шаг состоит из того, что каждый процесс рассчитывает локальную сумму в своей области, а затем используется операция *MPI_Allreduce* с функцией аггрегации *MPI_SUM* для того, чтобы все процессы получили общую сумму. Аналогичные операции производятся и при расчете величины невязки, однако там используется функция аггрегации *MPI_MAX* (максимум-норма).

Для распараллеливания с использованием технологии CUDA каждый процесс хранит на устройстве копию своей локальной обрабатываемой области и ее окрестность, необходимую для расчета оператора Лапласа, при этом все операции, за исключением инициализации производятся на CUDA. Были реализованы следующие ядра:

- kReduceSum – ядро агрегации суммы
- kReduceMax – ядро агрегации максимума
- kCalculateResidualVector – ядро расчета вектора невязки
- kCalculateProduct – ядро расчета скалярного произведения
- kCalculateProductLaplasian – ядро расчета скалярного произведение, при этом для первого аргумента применяется оператор Лапласа
- kCalculateBasisVect – ядро расчета промежуточного вектора
- kCalculateNextSolution – ядро расчета следующего решения
- kCalculateResiduals – ядро расчета ошибки

Для осуществления обменов процессов своими границами осуществляется их выгрузка в основную память, обмен с помощью MPI, и обновление значений на устройстве (для копирования строк используется функция cudaMemcpy, а для копирования столбцов – cudaMemcpy2D).

Поскольку в рамках каждого узла процессору доступны две видеокарты в SLI-режиме, а процессор одновременно может выполнять 8 потоков (8 MPI процессов), с помощью опции —ntasks-per-node=2 было введено ограничение – на одном процессоре выполняется по два процесса, таким образом с помощью cudaSetDevice(rank % 2) каждый выполняемый процесс получает по одной видеокарте.

При работе ядер каждая нить обрабатывала несколько точек. Для расчета конфигурации грида и блоков использовалась следующая стратегия:

- Каждый процесс обрабатывал $\lceil N0 * N1 \ / \ (multiProcessorCount * maxThreadsPerMultiProcessor) \rceil$ точек
- Число нитей в блоке: $\text{НОД}(maxThreadsPerBlock, maxThreadsPerMultiProcessor)$
- Число блоков: $\lceil N0 * N1 \ / \ (\text{число\_точек\_на\_нить} * \text{число\_нитей}) \rceil$

В рамках конфигурации суперкомпьютера Ломоносов число нитей в блоке при вычислении по данной формуле получалось равным 512, что соответствует оптимальному числу нитей, рассчитанному с помощью утилиты CUDA Occupancy Calculator.

Результаты расчетов

Ускорение рассчитывается по формуле $S = \frac{\text{Время решения на 1 процессоре}}{\text{Время решения на } N_p \text{ процессоров}}$.

Таблица с результатами расчетов на ПВС «Ломоносов» для MPI программы

Число процессоров N_p	Число точек сетки N^2	Время решения T (секунды)	Ускорение S по сравнению с одним процессом
1	1000 x 1000	202.063341	-
2	1000 x 1000	94.394817	2.140619182347
4	1000 x 1000	47.711379	4.235118439984726
8	1000 x 1000	25.996643	7.7726705328838035
16	1000 x 1000	13.292777	15.200987799614786
1	2000 x 2000	1602.125733	-
2	2000 x 2000	742.941428	2.156463043544262
4	2000 x 2000	375.551724	4.266058789281447
8	2000 x 2000	204.383137	7.838835221518299
16	2000 x 2000	102.769351	15.58952856479555

Погрешность приближенного решения одинакова при любом числе процессов:

- на сетке 1000 x 1000 $\psi = 0.000086799045$
- на сетке 2000 x 2000 $\psi = 0.000099594785$

Таблица с результатами расчетов на ПВС «Ломоносов» для MPI/CUDA программы

Число процессоров N_p	Число точек сетки N^2	Время решения T (секунды)	Ускорение S по сравнению с одним процессом	Ускорение S по сравнению с последовательной версией
1	1000 x 1000	6.108932	-	33.07670489702619
2	1000 x 1000	4.239837	1.4408412398872883	47.658280495217156
4	1000 x 1000	2.595469	2.353690989952105	77.85234229343521
8	1000 x 1000	2.515869	2.4281598127724457	80.31552556989256
16	1000 x 1000	1.934299	3.1582149398826145	104.46334356787655
1	2000 x 2000	45.493049	-	35.21693463544288
2	2000 x 2000	26.364071	1.7255699622414156	60.76928456914
4	2000 x 2000	14.248197	3.1928986523698404	112.4441031381
8	2000 x 2000	11.332358	4.0144380366380945	141.37620193431943

16	2000 x 2000	6.975037	6.522266333497585	229.69422714173413

Погрешность приближенного решения одинакова при любом числе процессов:

- на сетке 1000 x 1000 $\psi = 0.000086799045$
- на сетке 2000 x 2000 $\psi = 0.000099594785$

Рисунки и графики

Рисунок точного решения

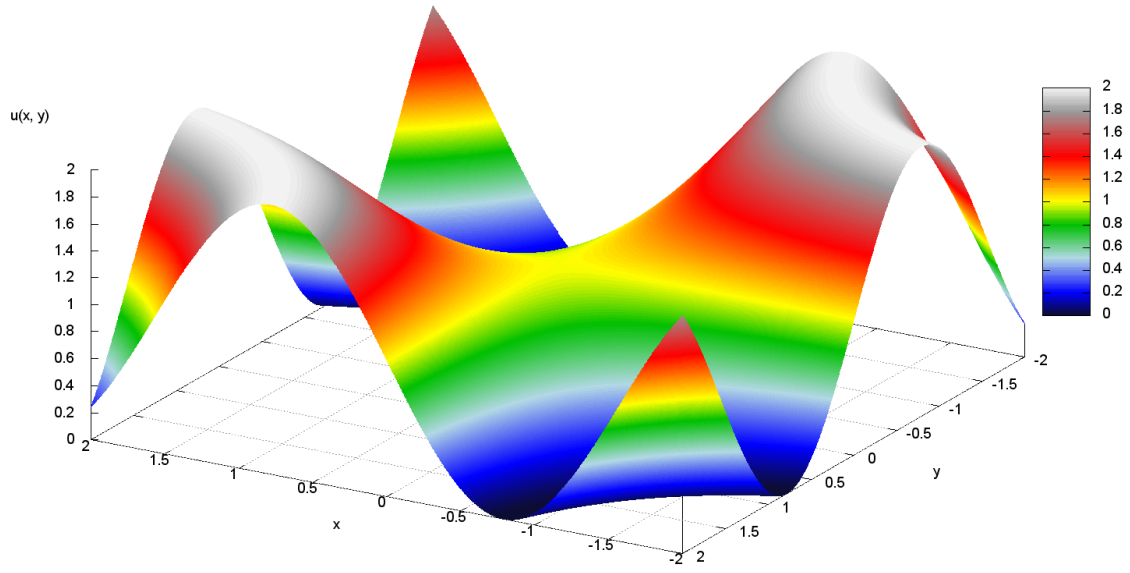


Рисунок приближенного решения на сетке 1000x1000 узлов

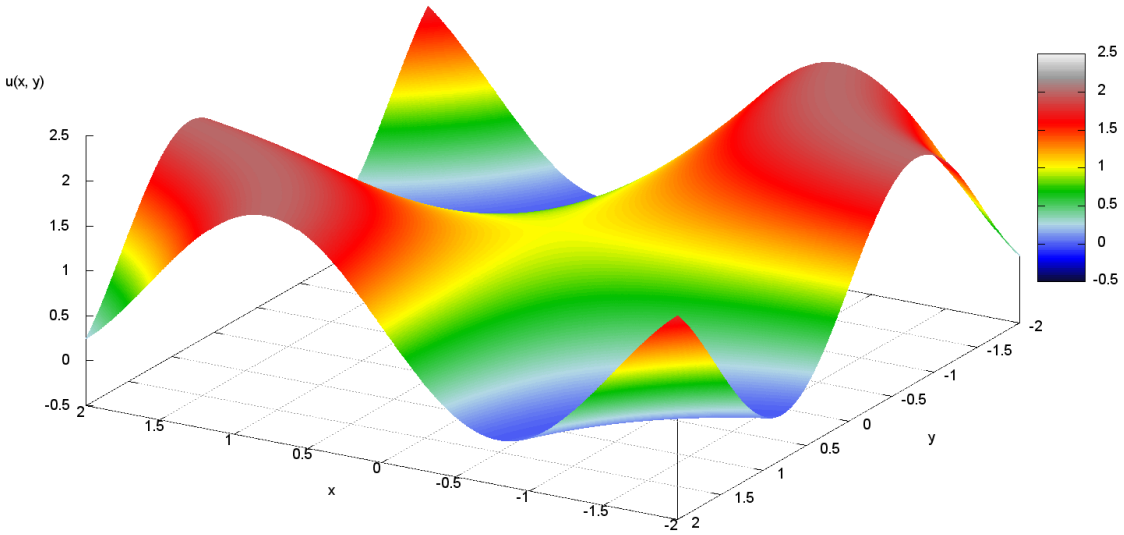


Рисунок приближенного решения на сетке 2000x2000 узлов

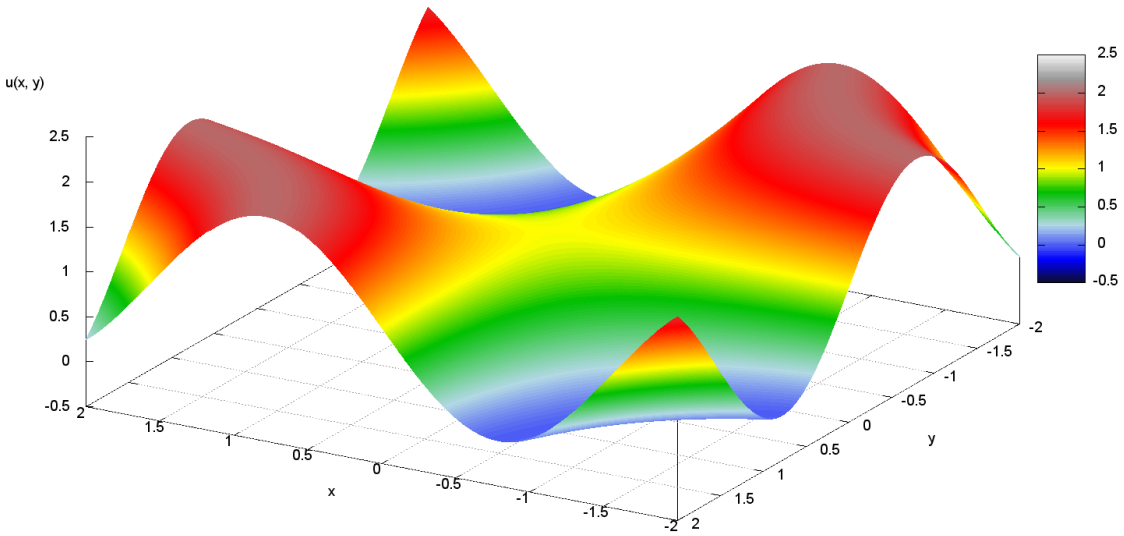
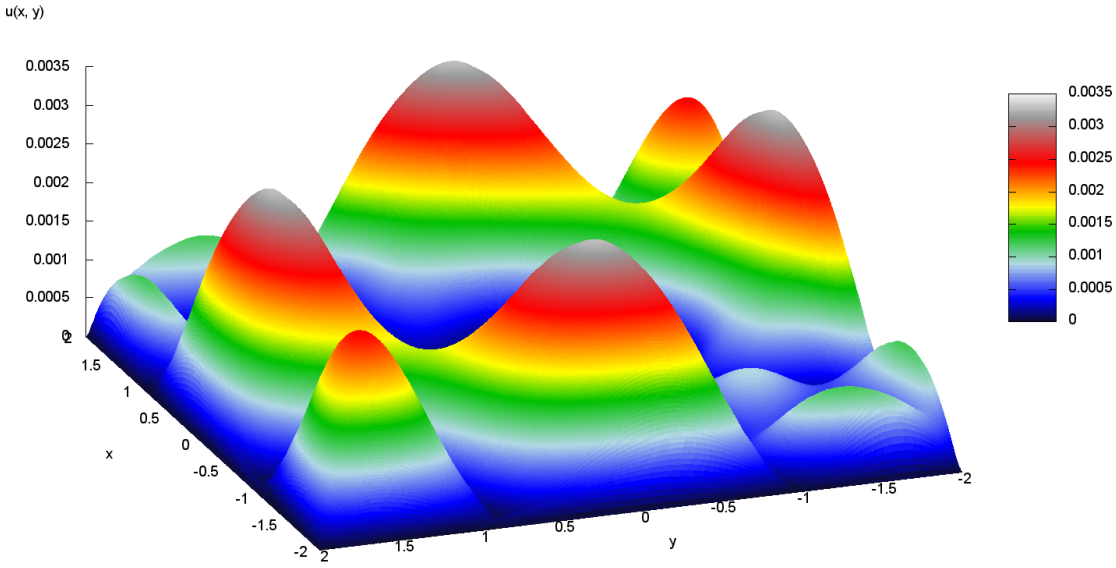
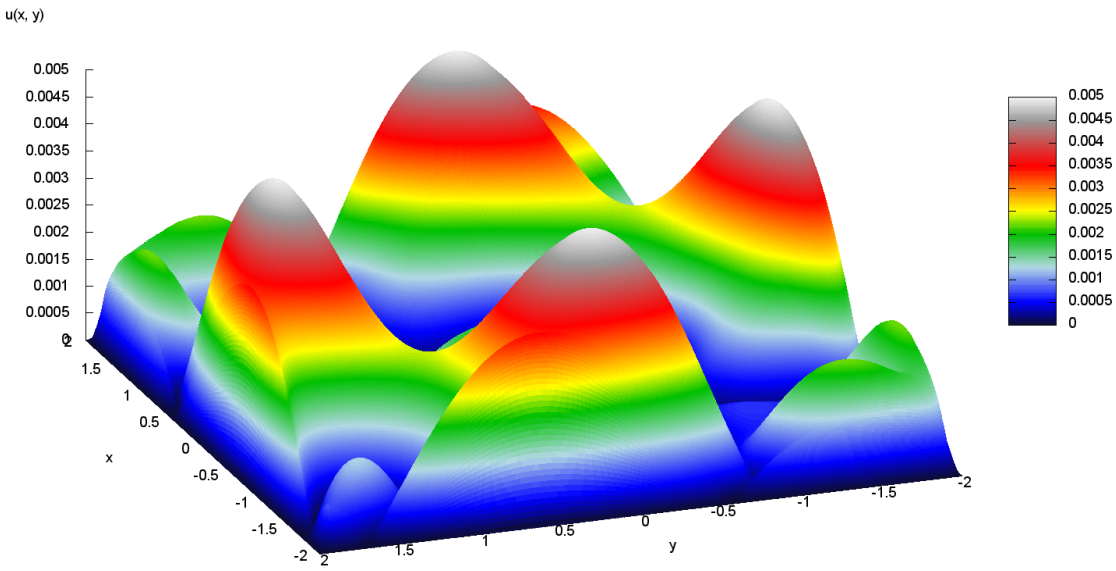


График абсолютной погрешности в каждой точке сетки 1000x1000 (графики на Ломоносове для MPI-программы и на Ломоносове для гибридной программы MPI/CUDA аналогичны)



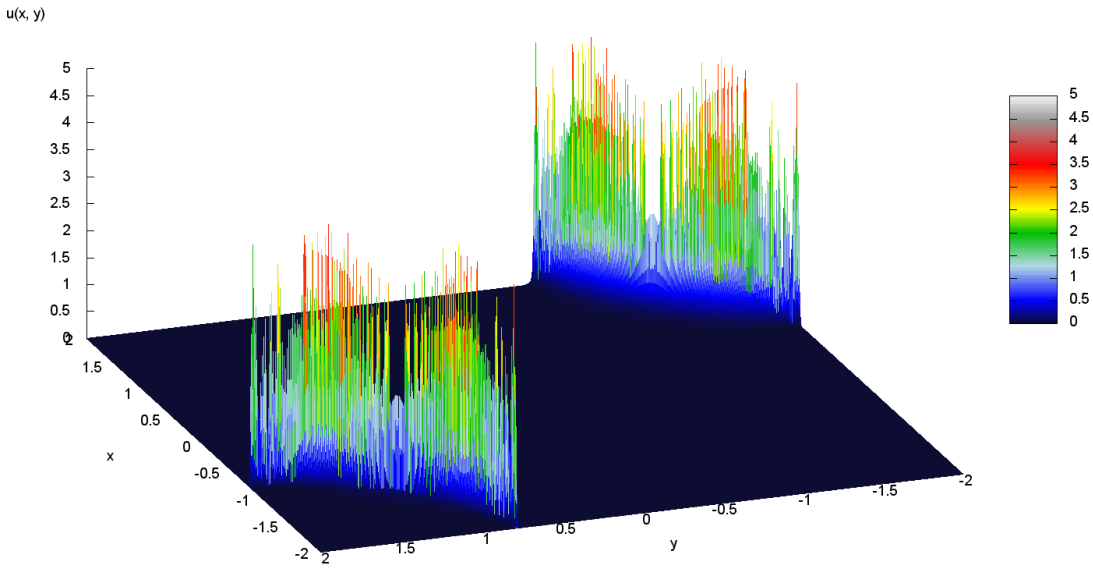
$Max\_value = 0.0032903082203561422$, $Min\_value = 4.6002535114553211e - 11$.

График абсолютной погрешности в каждой точке сетки 2000x2000 (графики на Ломоносове для MPI-программы и на Ломоносове для гибридной программы MPI/CUDA аналогичны)



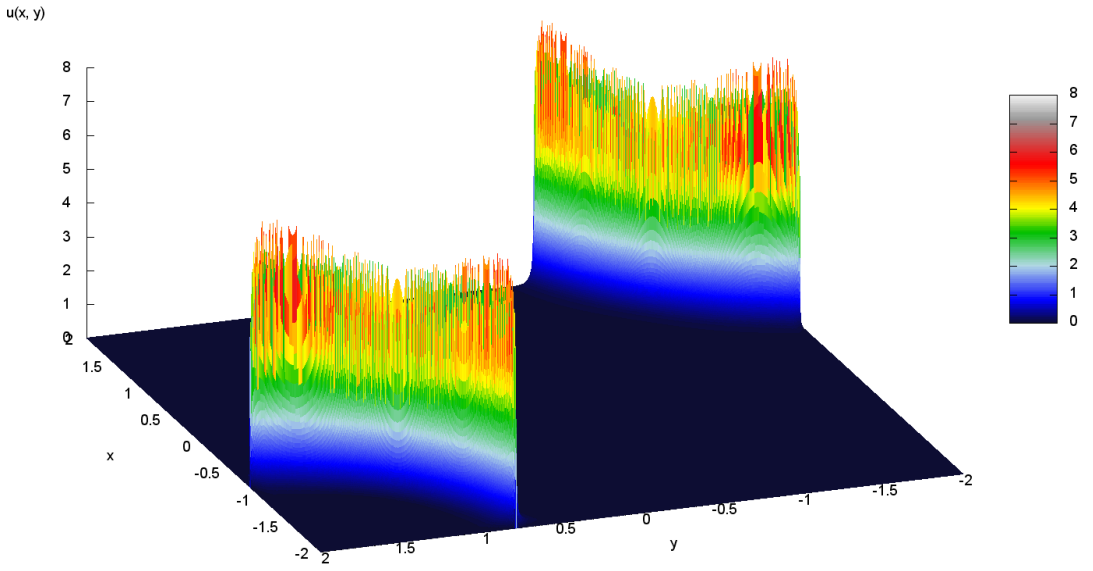
$Max\_value = 0.0049536301472701272$, $Min\_value = 5.3321347337487175e - 11$.

График логарифма относительной погрешности в каждой точке сетки 1000x1000, за вычетом граничных областей, если функция там принимает значение 0 (графики на Ломоносове для MPI-программы и на Ломоносове для гибридной программы MPI/CUDA аналогичны)



$Max\_value = 4.9158382983321571$, $Min\_value = 0.0$.

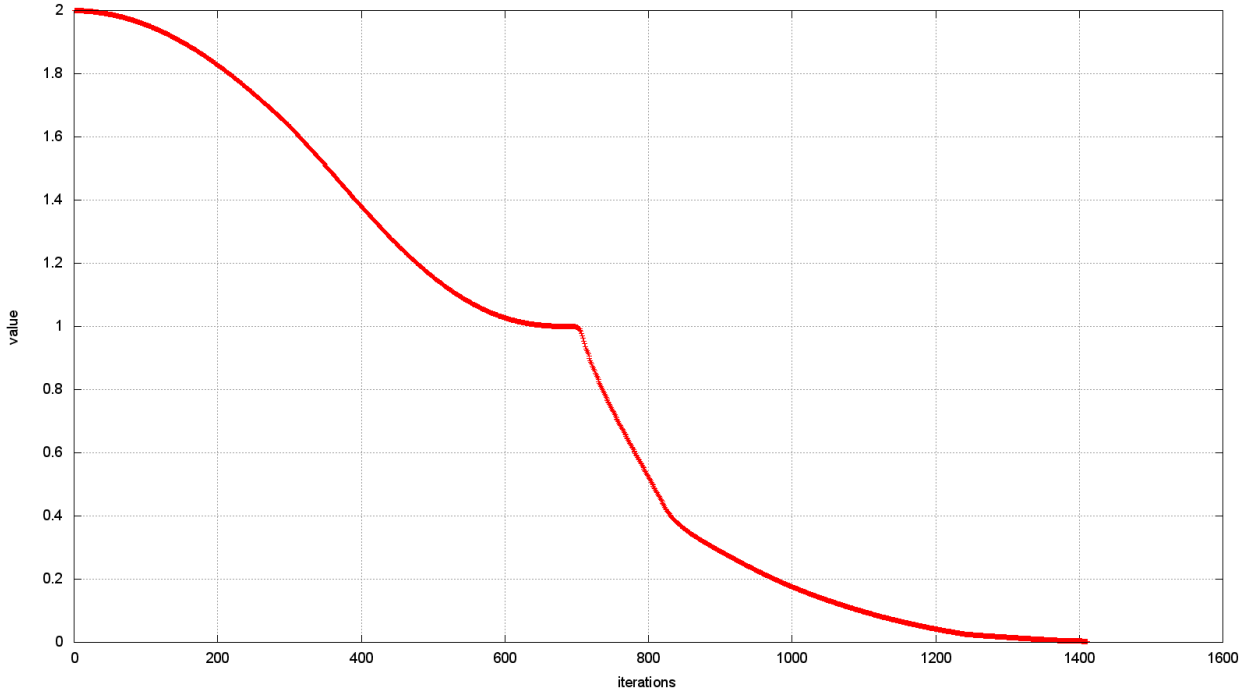
График логарифма относительной погрешности в каждой точке сетки 2000x2000, за вычетом граничных областей, если функция там принимает значение 0 (графики на Ломоносове для MPI-программы и на Ломоносове для гибридной программы MPI/CUDA аналогичны)



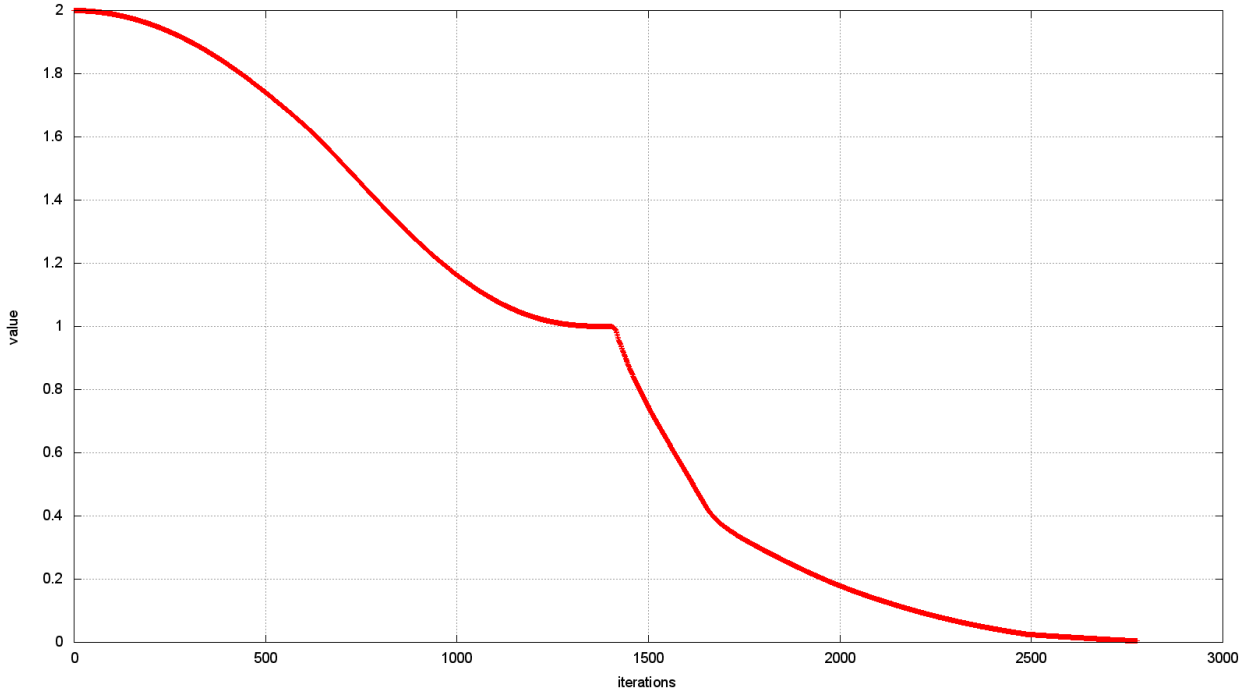
$Max\_value = 7.9140556939332818$, $Min\_value = 0.0$.

Графические изображения скорости сходимости (графики на Blue Gene/P для MPI-программы, на Blue Gene/P для гибридной программы MPI/OpenMP и ПВС «Ломоносов» для MPI программы аналогичны)

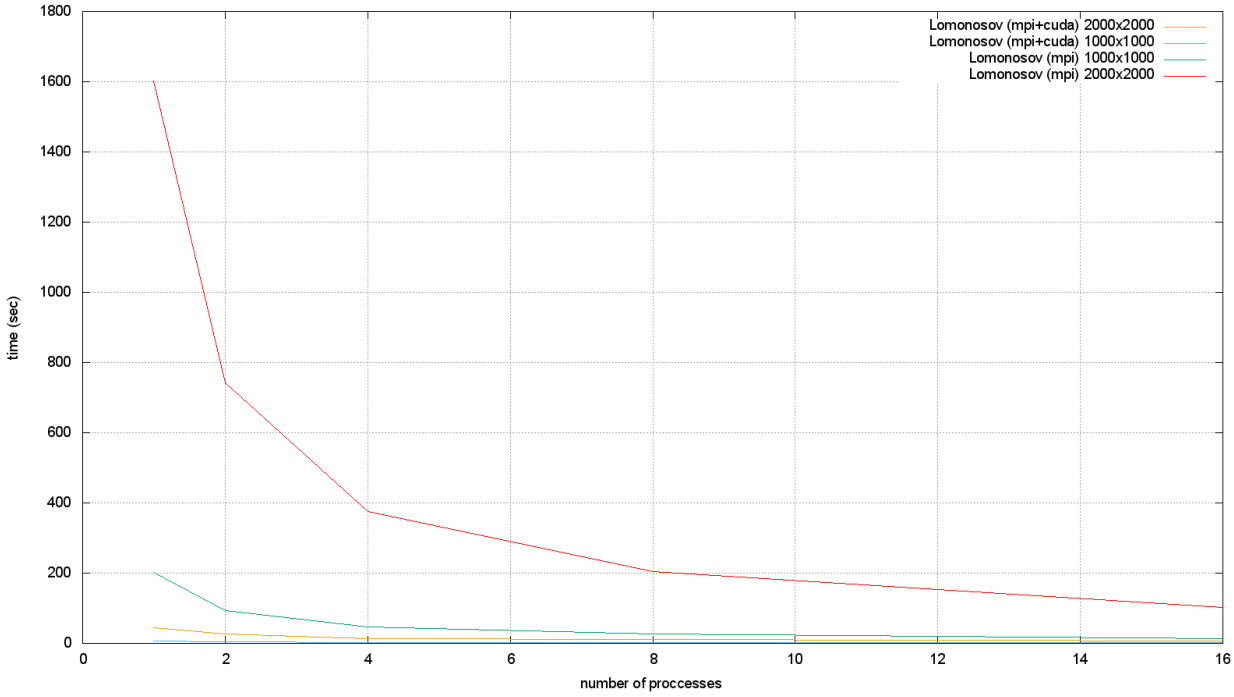
1) Сетка 1000x1000



2) Сетка 2000x2000

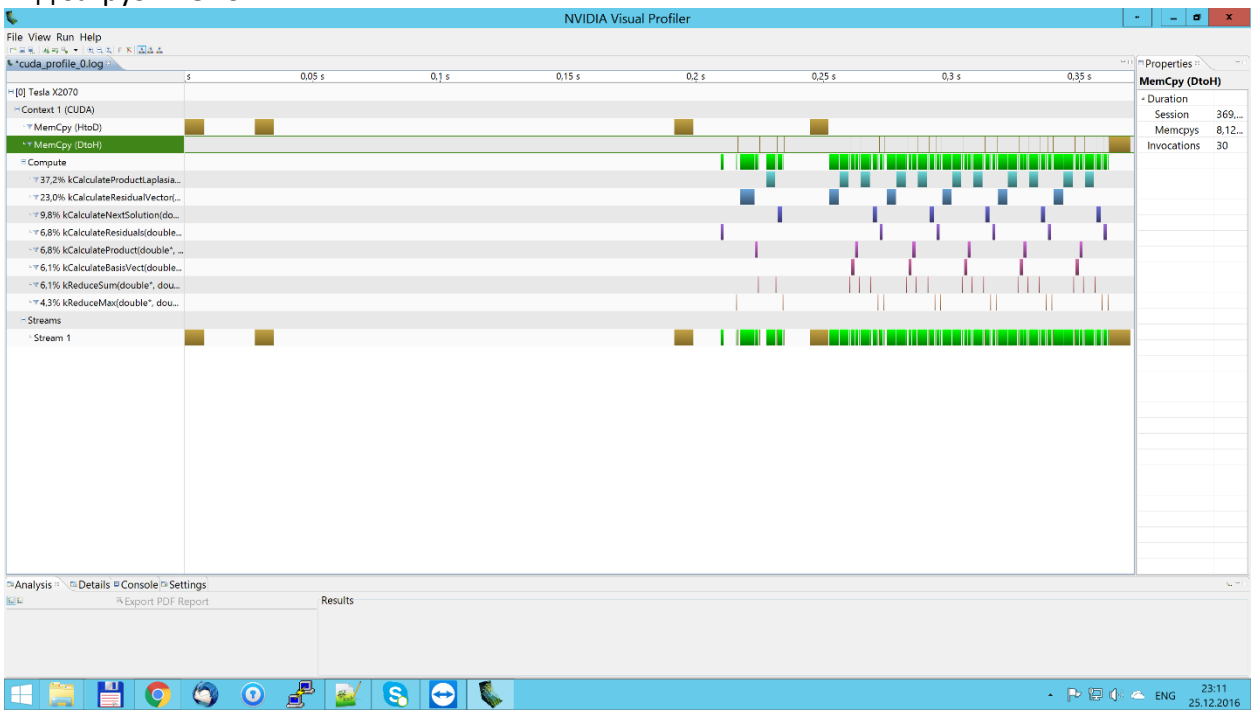


Результаты тестирования на суперкомпьютерах (зависимость времени выполнения от количества процессов)

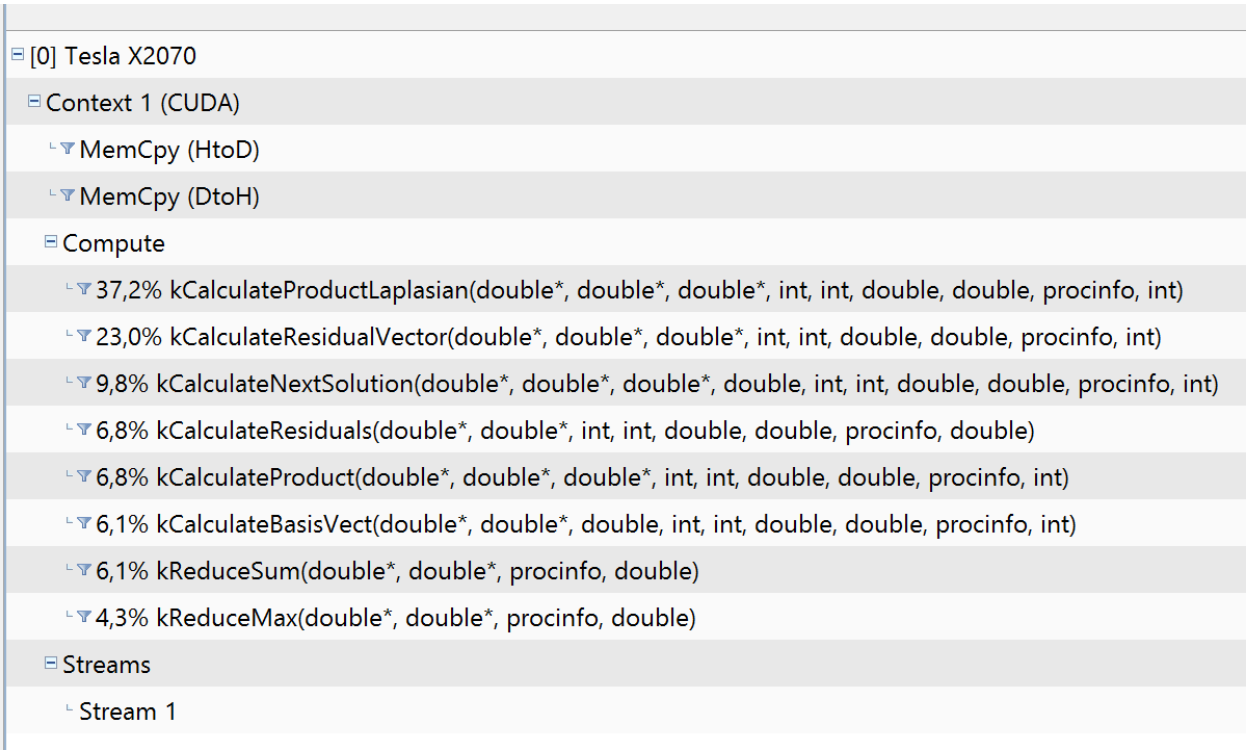


Профилирование

Профилирование было произведено для задачи с сеткой 1000x1000 по одной итерации метода скорейшего спуска и пяти итерациям метода сопряженных градиентов на суперкомпьютере Ломоносов для одного процесса. На рисунке ниже представлен общий вид загрузки GPU.



На рисунке ниже представлено распределение времени загрузки GPU различными ядрами. Как можно видеть, наибольшую часть времени занимает вычисление элементов суммы в скалярном произведении, в котором используется разностный оператор Лапласа в ядре kCalculateProductLaplasian, а также вычисление невязки, в котором также участвует оператор Лапласа в ядре kCalculateResidualVector.



Приложение к отчету

К отчету прилагается архив с исходными кодами: MPI – версия и гибридная версия MPI/CUDA.