Отчёт по практическому заданию №2.

Ю. Н. Лукашкина

julialukashkina@gmail.ru

МГУ имени М. В. Ломоносова, Москва

27 ноября 2016 г.

Аннотация

Данный документ содержит отчет по практической работе №2 курса «Суперкомпьютерное моделирование и технологии» ВМК МГУ.

1 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

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

и дополнительному условию

$$u(x, y) = \varphi(x, y) \tag{2}$$

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

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

$$F(x, y) = (x^2 + y^2)\sin(xy), \quad \varphi(x, y) = 1 + \sin(xy),$$

Прямоугольник $\Pi = [0, 2] \times [0, 2]$.

Было подобрано точное решение задачи Дирихле: $u(x, y) = 1 + \sin(xy)$.

2 Численный метод решения задачи

В расчётной области Π определяется прямоугольная сетка

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

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

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

$$\hbar_i^{(1)} = 0.5\big(h_i^{(1)} + h_{i-1}^{(1)}\big), \quad \hbar_j^{(2)} = 0.5\big(h_j^{(2)} + h_{j-1}^{(2)}\big)$$

Рассмотрим линейное пространство H функций, заданных на сетке ω_h. Будем считать, что в пространстве H задано скалярное произведение и максимум норма

$$\big(u,v\big)=\sum_{i=1}^{N_1-1}\sum_{j=1}^{N_2-1}\hbar_i^{(1)}\hbar_j^{(2)}u_{ij}v_{ij},\quad \|u\|=\max_{\substack{0<i<N_1\\0<j<N_2}}|u_{i,j}| \tag{3}$$

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

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

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

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

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

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

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) назвается разностной схемой для задачи (1), (2).

Приближенное решение системы уравнений (4) может быть получено методом сопряжённых градиентов. Начальное приближение $p^{(0)}$ и первая итерация $p^{(1)}$ вычисляются так:

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

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

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

Здесь

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

вектор

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

$$g_{ij}^{(0)}=r_{ij}^{(0)},$$

коэффициент

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

Вектор невязки $r^{(k)}$ вычисляется по следующим формулам:

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

Итерационный процесс останавливается, как только

$$||p^{(n)} - p^{(n-1)}|| < \varepsilon,$$

по евклидовой норме, где $epsilon$ — заранее выбранное положительное число ($epsilon = 10^{-4}$).

3 Описание работы по созданию MPI / OpenMP реализации

Все операции, кроме вычисления разностного оператора Лапласа, производятся независимо для каждого элемента матрицы. Это позволяет эффективно распараллелить работу с матрицами по процессорам, каждый из которых будет работать только с своей частью исходной матрицы. Для вычисления величин, которые требуют всю матрицу (таких как норма, скалярное произведение двух матриц и т.д.), процессоры будут синхронизироваться. Распараллеливание матрицы будет происходить по блокам/клеткам.

Чтобы корректно производить вычисление разностного оператора Лапласа каждый процессор кроме своей части матрицы будет хранить ещё граничные строки и столбцы. То есть одну предыдущую и одну последующую строчки относительно выделенных ему собственных строк (для процессоров, работающих с граничными частями матрицы соответствующие строки всегда будут пустыми). Аналогично будут храниться и дублирующие столбцы. Перед и после каждой операции подсчёта разностного оператора Лапласа процессоры будут обмениваться этими пограничными строками и столбцами.

3.1 Разделение матрицы

Исходная матрица делится между процессорами на клетки примерно одинакового размера. Для вычисления количества клеток по каждой из размерностей матрицы была использована функция `SplitFunction`, прилагающаяся к заданию. Число процессоров p является входным параметром и должно быть степенью двойки $p = 2^{power}$. Данную степень можно представить в следующем виде: $power = p_1 + p_2$, где $2^{p_1}, 2^{p_2}$ — число процессоров по первой и второй размерностям матрицы соотвественно.

3.2 MPI

Были использованы следующие функции:

— Для вычисления максимум нормы и подсчёта суммы.

```
MPI_Allreduce(...);
```

— Для обмена строками матрицы.

```
MPI_Sendrecv(...);
```

— Для подсчёта времени.

```
MPI_Wtime();
```

3.3 OpenMP

Были использованы следующие директивы:

— Для распараллеливания цикла.

```
#pragma omp parallel for
```

— Для подсчёта суммы.

```
#pragma omp parallel for reduction(+:sum)
```

— Для оформления критической секции программы.

```
#pragma omp critical
```

Таблица 1. Таблица с результатами расчётов на ПВС IBM Blue Gene/P

Число процессоров N_p	Число точек сетки N^3	Время решения T	Ускорение S	Эффективность
32	1000×1000	100.291		
128	1000×1000	26.2803	3.82	0.96
256	1000×1000	13.9082	7.21	0.90
512	1000×1000	7.46693	13.43	0.84
32	2000×2000	787.29		
128	2000×2000	201.831	3.90	0.98
256	2000×2000	102.624	7.67	0.96
512	2000×2000	52.2764	15.06	0.94

3.4 Вывод результатов программы

Результатом работы программы является вывод в консоль времени работы программы и погрешности решения. Также каждый процессор сохраняет в отдельные файлы свои части получившейся приближенной матрицы решения и матрицы точного решения.

4 Результаты расчётов

4.1 Blue Gene/P

Для ПВС IBM Blue Gene/P ускорение расчитывалось относительно 32 узлов. По таблице 1 видно, что реализованная программа распараллеливается хорошо. Для сетки из 1000 точек минимальная эффективность = 0.84, для сетки из 2000 точек = 0.94. На сетке большего размера достигается большее ускорение и эффективность в среднем выше. На обеих сетках эффективность уменьшается с ростом числа процессоров.

Гибридная реализация MPI/OpenMP (см. таблицу 2) по времени работает в ≈ 1.3 быстрее, чем просто MPI версия. Ускорение S_2 (относительно 32 процессоров, негибридная реализация) также примерно в 1.3 выше, чем у негибридной программы. Однако значение ускорения (S_1) и эффективности хуже, чем у MPI реализации. С увеличением числа процессоров эффективность всё больше удаляется от 1, на 512 процессорах эффективность значительно падает. Это, скорее всего, связано с тем, что размер клетки, выделяемой каждому процессору становится всё меньше и накладные расходы на управление потоками не компенсируются полученным ускорением. Этот эффект особенно хорошо заметен на сетке из 1000 точек.

4.2 ПВС «Ломоносов»

Для ПВС «Ломоносов» ускорение расчитывалось относительно одного процессора (см. таблицу 3). На обеих сетках для небольшого числа процессоров эффективность примерно одинаковая. Значения ускорения на сетках в 1000 и 2000 узлов для небольшого числа процессоров отличаются незначительно. При конфигурации запуска с 128 процессорами и 1000 узлами эффективность ощутимо уменьшается. Это связано с тем, что затраты на передачу информации между процессорами становятся слишком высокими. На сетке в 2000 узлов для конфигурации с 128 процессорами размер клетки, выделяемой каждому процессору, в 2 раза больше, чем на сетке в 1000 узлов и эффективность не сильно отклоняется от средней.

4.3 Blue Gene/P (MPI реализация) и ПВС «Ломоносов»

При одинаковом числе процессоров время работы на ПВС «Ломоносов» значительно меньше, чем на Blue Gene/P. Для 32 процессоров ≈ 14 и 116 против ≈ 100 и 787. Для 128 процессоров ≈ 4 и 30 против ≈ 26 и 102. Значение эффективности в среднем выше на Blue Gene/P.

5 Рисунки решения

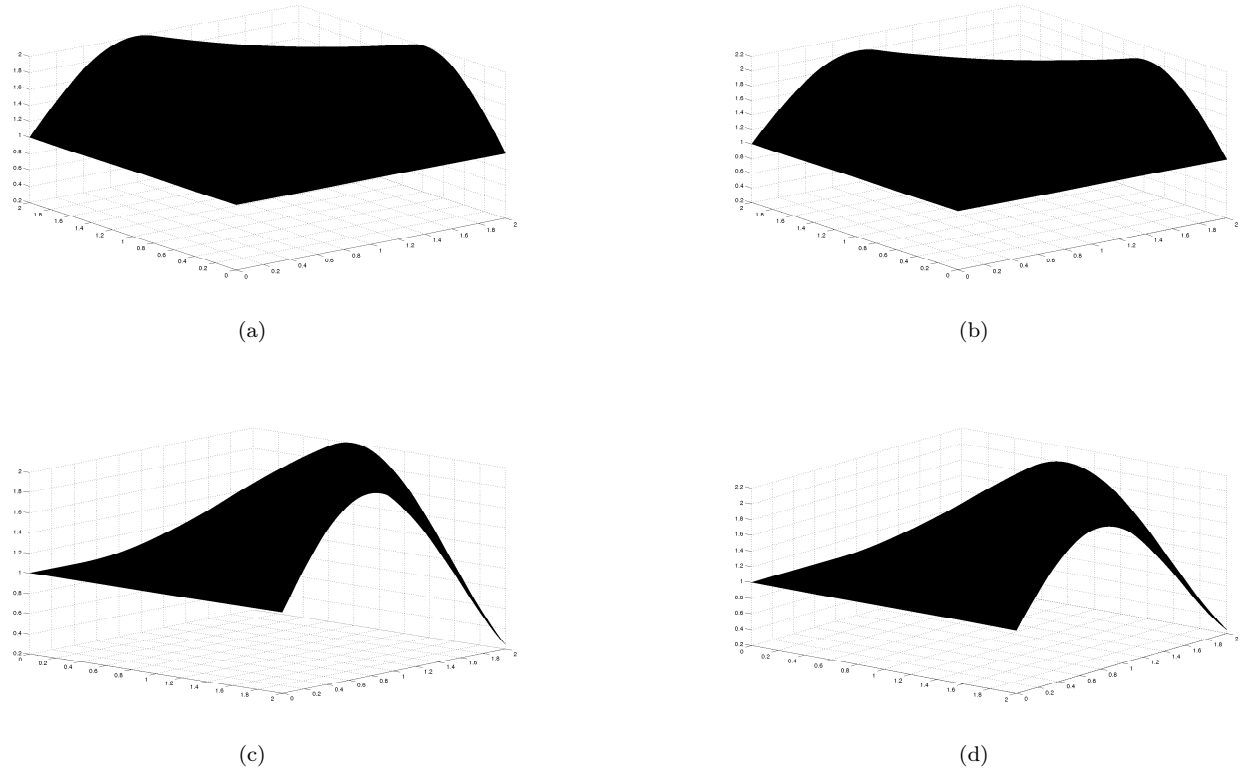
На рисунке 1 представлены точное и приближенное решение на сетке в 2000 точек. Погрешность решения составила 0.00784845. Сбор и построение целой матрицы по сохраненным каждым процессором части производился в matlab. Рисунки также были построены в matlab.

Таблица 2. Таблица с результатами расчётов на ПВС IBM Blue Gene/P гибридной программы MPI/OpenMP. Ускорение S_1 относительно гибридной программы с 32 процессорами, ускорение S_2 относительно негибридной программы с 32 процессорами

Число процессоров N_p	Число точек сетки N^3	Время решения T	Ускорение S_1	Эффективность	Ускорение S_2
32	1000×1000	69.6141			1.44
128	1000×1000	18.4298	3.78	0.95	5.44
256	1000×1000	10.06	6.92	0.86	9.97
512	1000×1000	7.5553	9.20	0.58	13.27
32	2000×2000	481.0016			1.64
128	2000×2000	123.512	3.89	0.97	6.37
256	2000×2000	64.0308	7.51	0.94	12.29
512	2000×2000	38.2781	12.56	0.78	20.57

Таблица 3. Таблица с результатами расчётов на ПВС «Ломоносов»

Число процессоров N_p	Число точек сетки N^3	Время решения T	Ускорение S	Эффективность
1	1000×1000	379.135		
8	1000×1000	56.7781	6.78	0.85
16	1000×1000	27.7419	13.67	0.85
32	1000×1000	13.9754	27.13	0.85
128	1000×1000	4.39367	86.3	0.68
1	2000×2000	2979.66		
8	2000×2000	478.44	6.227	0.78
16	2000×2000	241.593	12.33	0.78
32	2000×2000	115.166	25.87	0.80
128	2000×2000	29.626	100.58	0.79



(a) (b)

(c) (d)

Рис. 1. Рисунки решения: (a), (c) точное решение, (b), (d) приближенное, погрешность решения = 0.00784845

6 Выводы

Выполненная программная реализация хорошо масштабируется. Однако при росте числа процессоров и уменьшении размера клетки накладные расходы на распараллеливание уменьшают эффективность параллельной программы.