Эффективное программирование современных микропроцессоров и мультипроцессоров

Практическое задание 1

Цель: научиться разрабатывать простые программы численного моделирования, применять базовые средства оптимизации программ, выполнять оценку и анализ производительности программ, пользоваться средствами профилирования.

Постановка задачи

1. Разработать программу в соответствии с одним из вариантов. Проверить правильность её работы.
2. Выполнить базовую оптимизацию программы (вручную и с помощью компилятора) с целью минимизации времени её работы. На каждом этапе оптимизации измерять время работы программы. Отмечать в отчёте успешные и неуспешные попытки оптимизации. Для оценки времени работы программы использовать одни и те же параметры: $N_X = N_Y \approx 8000 - 10000$, $N_T \approx 100 - 120$.
3. Для наиболее быстро работающего варианта программы с помощью средств профилирования выполнить следующие действия:
    a. Построить граф вызовов программы (картинку), определить «горячие точки» программы с точностью до функций.
    b. Построить аннотированный листинг программы, определить «горячие точки» программы с точностью до строк исходного кода и машинных команд.
    c. Собирая информацию о соответствующих событиях, получить с помощью профилирования следующие характеристики исполнения программы в целом:
        i. среднее число тактов на микрооперацию (или микроопераций на такт),
        ii. процент кэш-промахов для кэшей первого и последнего уровней,
        iii. процент неправильно предсказанных переходов.
    d. Сделать предположение о том, что является основной причиной временных затрат (вычислительные операции, обращения в память, выполнение команд перехода, …).
4. На roofline-модели отметить точку, соответствующую программе. Требуемые характеристики оценить исходя из анализа кода в «горячей точке» программы.

Варианты

1. Задача 1, тип данных float,
2. Задача 1, тип данных double,
3. Задача 2, тип данных float,
4. Задача 2, тип данных double.

Отчёт

Отчёт по работе должен содержать:

- ФИО, группа, номер лабораторной работы, номер варианта
- Задание (коротко)
- Полное название процессора, на котором происходило тестирование
- Текст первого работающего варианта программы (в приложении)
- Текст самого быстрого варианта программы (в приложении)
- Описание использованных способов оптимизации программы с результатами
- Результаты профилирования программы
    - Граф вызовов (картинка), обозначение «горячей точки»
    - Ассемблерный листинг «горячей точки» программы (можно в приложении)
    - Характеристики исполнения программы
    - Предположения об основных причинах временных затрат
- Roofline-модель с точкой, соответствующей программе.
- Вывод

Задача 1. Решение волнового уравнения методом конечных объёмов

Параметры программы

- Вход:
  - Размеры сетки: N_X, N_Y
  - Число шагов: N_T
- Выход:
  - Время счёта

Описание алгоритма

Алгоритм моделирует распространение волны в двумерной области, инициированной импульсом из заданного узла сетки. В начальный момент времени значения искомой функции U на сетке инициализируются нулями. На каждом шаге моделирования значения искомой функции пересчитываются по заданной формуле.

Параметры алгоритма:

- Пространство:
  - Область моделирования: $[X_A : X_B] \times [Y_A : Y_B] = [0.0 : 4.0] \times [0.0 : 4.0]$.
  - Пространственная сетка, $N_X \times N_Y$ узлов с номерами (i, j): $i = 0 \dots N_Y–1$, $j = 0 \dots N_X–1$.
  - Шаги сетки по пространству: $h_X = (X_B – X_A) / (N_X – 1)$, $h_Y = (Y_B – Y_A) / (N_Y – 1)$.
  - Координаты узлов сетки: $X_j = X_1 + j \cdot h_X$, $Y_i = Y_1 + i \cdot h_Y$.
- Время:
  - Последовательность номеров моментов времени (шагов расчёта): $n = 0, 1, \dots N_T$.
  - Величина шага по времени (между последовательными моментами времени):
    - $\tau = 0.01$, для $N_X \le 1000$ и $N_Y \le 1000$,
    - $\tau = 0.001$, для $N_X > 1000$ или $N_Y > 1000$.
  - Значение времени в момент n: $T_n = n \cdot \tau$.
- Источник импульса:
  - Координаты узла с источником импульса: $S_X = 1$, $S_Y = N_Y / 2$.
  - Значение функции источника в момент времени n в узле сетки (i, j):
    - $f_{i,j}^n = \exp(–(2\pi f_0 \cdot (n \cdot \tau – t_0))^2 / \gamma^2) \cdot \sin(2\pi f_0 \cdot (n \cdot \tau – t_0))$, где $f_0 = 1.0$, $t_0 = 1.5$, $\gamma = 4.0$, если $j = S_X$ и $i = S_Y$.
    - $f_{i,j}^n = 0$, если $j \ne S_X$ или $i \ne S_Y$.
- Сеточные значения:
  - Фазовая скорость (характеристика пространства, отражает скорость распространения волны):
    - $P_{i,j} = 0.1 \cdot 0.1$, если $j < N_X / 2$,
    - $P_{i,j} = 0.2 \cdot 0.2$, если $j \ge N_X / 2$.
  - Искомая функция, значения в моменты времени n = 0 и n = –1:
    - $U_{i,j}^{-1} = U_{i,j}^0 = 0.0$, при $i = 0 \dots N_Y–1$, $j = 0 \dots N_X–1$.

При реализации следует считать, что $U_{i,j}^n$ и $P_{i,j}$ могут принимать произвольные значения, и их следует задавать массивами. Также следует считать, что значения $f_{i,j}^n$ не равны нулю только в одной заданной точке, и можно их задавать не массивом.

Шаг алгоритма (двухслойная явная схема):

$$U_{i,j}^{n+1} = 2U_{i,j}^n - U_{i,j}^{n-1} + \tau^2 \left[f_{i,j}^n + \frac{\left(U_{i,j+1}^n - U_{i,j}^n\right)\left(P_{i-1,j} + P_{i,j}\right) + \left(U_{i,j-1}^n - U_{i,j}^n\right)\left(P_{i-1,j-1} + P_{i,j-1}\right)}{2h_x^2} + \right.$$
$$\left. + \frac{\left(U_{i+1,j}^n - U_{i,j}^n\right)\left(P_{i,j-1} + P_{i,j}\right) + \left(U_{i-1,j}^n - U_{i,j}^n\right)\left(P_{i-1,j-1} + P_{i-1,j}\right)}{2h_y^2} \right],$$

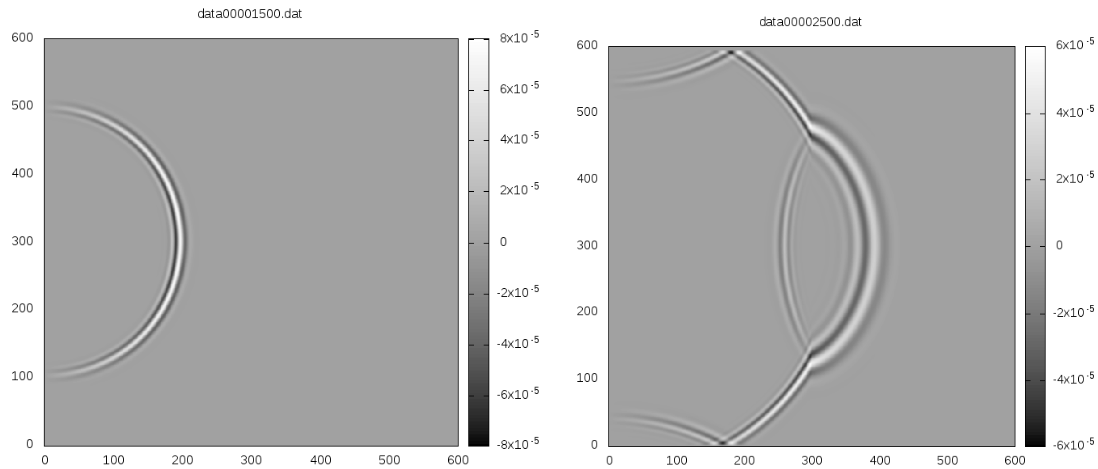
где $i = 1 \dots N_Y–2$, $j = 1 \dots N_X–2$.

Контроль расчёта

Для контроля правильности расчёта следует после каждого шага n вычислять значение:

$U_{max}^{n} = \max_{i,j} | U_{i,j}^{n} |$.

При корректной работе алгоритма это значение не должно неограниченно увеличиваться, а должно оставаться в некоторых небольших пределах. При модификациях программы значение U_{max}^{n} для данной итерации n должно сохраняться. Также для проверки необходимо нарисовать распределение искомой функции U. Примеры результата расчёта для $N_X = N_Y = 600$, $N_T = 1500$ и $N_T = 2500$ приведены на рисунке.



Следующий скрипт для программы gnuplot позволяет нарисовать распределение искомой функции из массива, заданного в файле в бинарном формате (элементы типа double).

```
nx=600
ny=600
filename="double00001500.dat"
set terminal png size 700,600
set output filename.".png"
set xrange[-1:nx]
set yrange[-1:ny]
set palette gray
set title filename
plot filename binary array=(ny,nx) format="%lf" with image
```

Для типа данных float в последней строке следует использовать параметр format="%f".

Задача 2. Решение уравнения Пуассона методом Якоби

Параметры программы

- Вход:
  - Размеры сетки: N_X, N_Y
  - Число шагов: N_T
- Выход:
  - Время счёта

Описание алгоритма

Алгоритм моделирует установление стационарного распределение тепла в пластинке с заданным распределением источников и стоков тепла. В начальный момент времени значения искомой функции на сетке инициализируются нулями. На каждом шаге моделирования значения искомой функции пересчитываются по заданной формуле.

Параметры алгоритма:

- Пространство:
  - Область моделирования: $[X_A : X_B] \times [Y_A : Y_B] = [0.0 : 4.0] \times [0.0 : 4.0]$.

- Пространственная сетка, $N_X \times N_Y$ узлов с номерами (i, j): $i = 0 \dots N_Y-1$, $j = 0 \dots N_X-1$.
- Шаги сетки по пространству: $h_X = (X_B - X_A) / (N_X - 1)$, $h_Y = (Y_B - Y_A) / (N_Y - 1)$.
- Координаты узлов сетки: $X_j = X_1 + j \cdot h_X$, $Y_i = Y_1 + i \cdot h_Y$.

- Итерации:
  - Последовательность итераций: $n = 0, 1, \dots N_T$.
- Сеточные значения:
  - Распределение источников и стоков тепла (характеристика пространства):
    - $\rho_{i,j} = 0.1$, если $(X_j - X_{S1})^2 + (Y_i - Y_{S1})^2 < R^2$
    - $\rho_{i,j} = -0.1$, если $(X_j - X_{S2})^2 + (Y_i - Y_{S2})^2 < R^2$
    - $\rho_{i,j} = 0.0$, иначе.
    - Здесь:
      - $X_{S1} = X_A + (X_B - X_A) / 3$, $Y_{S1} = Y_A + (Y_B - Y_A) \cdot 2 / 3$,
      - $X_{S2} = X_A + (X_B - X_A) \cdot 2 / 3$, $Y_{S2} = Y_A + (Y_B - Y_A) / 3$,
      - $R = 0.1 \cdot \min(X_B - X_A, Y_B - Y_A)$
  - Искомая функция, начальные значения:
    - $\Phi_{i,j}^0 = 0.0$, при $i = 0 \dots N_Y-1$, $j = 0 \dots N_X-1$.

При реализации следует считать, что $\Phi_{i,j}^n$ и $\rho_{i,j}$ могут принимать произвольные значения, и их следует задавать массивами.

Шаг алгоритма (9-точечный шаблон, метод Якоби):

$$\Phi_{i,j}^{n+1} = \frac{1}{5\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right)} \left[\frac{1}{2}\left(\frac{5}{h_x^2} - \frac{1}{h_y^2}\right)\left(\Phi_{i,j-1}^n + \Phi_{i,j+1}^n\right) + \frac{1}{2}\left(\frac{5}{h_y^2} - \frac{1}{h_x^2}\right)\left(\Phi_{i-1,j}^n + \Phi_{i+1,j}^n\right) + \right.$$
$$+ \frac{1}{4}\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right)\left(\Phi_{i-1,j-1}^n + \Phi_{i-1,j+1}^n + \Phi_{i+1,j-1}^n + \Phi_{i+1,j+1}^n\right) +$$
$$\left. + 2\rho_{i,j} + \frac{1}{4}\left(\rho_{i-1,j} + \rho_{i+1,j} + \rho_{i,j-1} + \rho_{i,j+1}\right) \right]$$

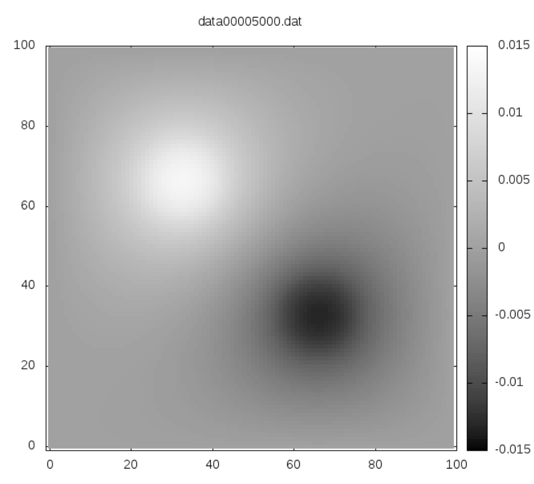
где $i = 1 \dots N_Y-2$, $j = 1 \dots N_X-2$.

Контроль расчёта

Для проверки правильности расчёта необходимо после каждой итерации вычислять значение:

$\delta^{n+1} = \max_{i,j} | \Phi_{i,j}^{n+1} - \Phi_{i,j}^n |$.

При корректной работе алгоритма это значение должно на каждой очередной итерации уменьшаться. При модификациях программы значение δ^n для данной итерации n должно сохраняться. Также для проверки необходимо нарисовать распределение искомой функции Ф. Примеры результата расчёта для $N_X = N_Y = 100$, $N_T = 5000$ приведены на рисунке.



Скрипт программы gnuplot аналогичен скрипту из задачи 1.