Задание на спецкурс CUDA, 6 курс РК6

Селиверстов Е.Ю.

17 ноября 2012 года

Версия документа 1.2.

1 Метод Якоби решения СЛАУ

Представлена система размерности n.

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{b} \tag{1}$$

Размерность матрицы $\mathbf{A}$ – m строк и n столбцов, векторов $\mathbf{X}$ и $\mathbf{b}$ - n.
Запишем следующие матрицы:

$$\mathbf{D} = diag(\mathbf{A}) \tag{2}$$

$$\mathbf{B} = \mathbf{D} - \mathbf{A} \tag{3}$$

Матрица $\mathbf{H}$:

$$\mathbf{H} = \mathbf{D}^{-1} \cdot \mathbf{B} \tag{4}$$

$$\mathbf{d} = \mathbf{D}^{-1} \cdot \mathbf{b} \tag{5}$$

Итерационное стационарное уравнение решения СЛАУ (1):

$$\mathbf{x}^{r+1} = \mathbf{H} \cdot \mathbf{x}^{r} + \mathbf{d} \tag{6}$$

Здесь $r = 0 \ldots$ ..., $\mathbf{x}^0$ – начальное приближение
Отметим, что матрицы $\mathbf{D}$, $\mathbf{B}$ и векторы $\mathbf{d}$ вычисляются до начала итераций.
Матрица A должна быть хорошо обусловлена ($cond(\mathbf{A}) < 100$). В задании генерируется матрица с диагональным преобладанием. Достаточным условием сходимости итерационного процесса (6) является $|\mathbf{H}| < 1$.

2 Схема распараллеливания

Метод Якоби является явным итерационным методом и хорошо подходит для распараллеливания.
Допустим, что используем графический процессор архитектуры CUDA с M мультипроцессорами и N процессорами в мультипроцессоре.
Логично применить блочный метод распараллеливания умножения и сложения матриц. Предположим размер блока в матрице W.

$$G = n \, div \, W \tag{7}$$

Выгодно иметь $n \, mod \, W = 0$, но это совершенно не обязательно.
Рекомендуемая конфигурация вычислительной сети:

- размер блока $W \times W \times 1$
- размер сети $G \times G$

В ядре рекомендуется выполнять итерации метода Якоби.

3 Критерий останова

$$\Delta F = \sum_{i=1}^{n} \|x_i^{k+1} - x_i^k\| \tag{8}$$

$$\Delta F < \varepsilon \, , \quad \varepsilon = 10^{-6} \tag{9}$$

Критерий останова рекомендуется проверять на хост-системе.

4 Требования к программе

4.1 Распараллеливание

Необходимо разработать как минимум одно параллельное CUDA-ядро.

- итерации метода Якоби – обязательно
- вычисление критерия останова ΔF – по желанию
- вычисление матрицы H и вектора d – по желанию, метод обращения найдите самостоятельно

4.2 Входные данные

Программе на вход подаются имена файлов с матрицами **A**, **b**, **H**, **d**, **X** в формате GNU Octave.

Формат записи векторов – на каждой строке файла по очередному значению вектора. Формат записи матриц – на каждой строке файла по строке матрицы, значения разделены пробелом. В начале файла следуют заголовки Octave.

Вектор **X** является выходным, программа записывает в него результат решения СЛАУ.

Пример: `cuda-program a.txt b.txt h.txt d.txt x.txt`

Для каждого варианта задания выдаются файлы с матрицами условия **A**, **b**, **H**, **d** и с точным ответом $\mathbf{X}^\star$.

Конфигурация вычислительной сети (размер блока W) может также варьироваться при приеме задания.

4.3 Проверка данных

Сгенерировать тестовые данные и проверить свое решение можно с помощью GNU Octave. Язык Octave условно совместим с MATLAB, поэтому программы можно запускать и в нем.

Проверка решения СЛАУ Пусть найденное решение записано в файле `x.txt`.

```
load "a.txt"
load "b.txt"
load "x.txt"
fdelta = max(abs(A * x - b))
```

Полный актуальный текст скрипта в `check.m`.

Создание матриц СЛАУ Создаются случайные матрицы, без учета устойчивости численного решения. В файл `xexact.txt` записывается точное решение.

Полный актуальный текст скрипта в `generate.m`.

```
% version 1.2
n = 1000
maxval = 10
A = rand(n,n) * maxval;
for i = 1:n
  A(i,i) = rand*maxval + sum(A(i,:)) - A(i,i);
endfor
xexact = rand(n,1) * maxval - maxval/2 ;
b = A * xexact ;
D = diag(diag(A)) ;
B = D - A ;
H = inv(D) * B ;
d = inv(D) * b ;
fdelta = max(abs(H * xexact + d - xexact)) ;
disp("Solution delta:"), disp(fdelta);
disp("Matrix norm of H: "), disp(norm(H));
disp("Condition number of A: "), disp(cond(A));
save "a.txt" A
save "b.txt" b
save "h.txt" H
save "d.txt" d
save "xexact.txt" xexact
```