информационные технологии (ПИТ 2019): труды Международной научно-технической конференции. 2019. С. 28-31.

2. Выгодчикова И.Ю., Гусятников В.Н., Акимова С.А. Модель формирования инвестиционного портфеля с использованием минимаксного критерия // Вестник Саратовского государственного социально-экономического университета. 2018. № 3 (72). С. 170-174.

3. Выгодчикова И.Ю., Селиванова А.А. Оценивание риска портфельного инвестирования на базе иерархической модели // Известия Саратовского университета. Новая серия. Серия: Экономика. Управление. Право. 2016. Т.16. Выпуск 1. С. 80-85.

4. Markovitz H.M. Portfolio selection // J. of Finances. 1952. Vol. 7, №1.

Д.Л. Головашкин[1], Л.В. Яблокова[2]

БЛОЧНЫЙ АЛГОРИТМ МЕТОДА ЯКОБИ ДЛЯ РЕШЕНИЯ НЕЯВНЫХ СЕТОЧНЫХ УРАВНЕНИЙ. ОДНОМЕРНЫЙ СЛУЧАЙ

([1] Институт систем обработки изображений РАН – филиал ФНИЦ «Кристаллография и фотоника» РАН, [2] Самарский университет)

Введение

Метод Якоби несмотря на простоту, а возможно именно благодаря ей, издавна привлекал пристальное внимание разработчиков эффективных алгоритмов решения различных сеточных уравнений. Так, еще тридцать лет назад на его примере Джеймс Ортега [1] иллюстрировал приемы синтеза векторных и параллельных алгоритмов. Не забывали про метод Якоби и далее; в объемном и тщательно проработанном исследовании [2] демонстрируются преимущества и недостатки различных подходов к построению блочных вычислительных процедур, их практическая реализация на многопроцессорных ЭВМ и графических ускорителях именно для обсуждаемого численного метода.

В работе [3] метод Якоби реализуется на конвейерных ускорителях вычислений, а в [4] предлагается обобщенный подход для произвольных параллельных архитектур. При этом без рассмотрения остался важный случай производства расчетов в рамках одного вычислительного потока с хранением в памяти только двух итерационных приближений, имеющий как практическую (не у всех пользователей в распоряжении имеются суперкомпьютеры с большим объемом памяти и множеством потоков), так и методическую ценность (по коду параллельного алгоритма зачастую значительно сложнее разобраться в тонкостях организации блочных вычислений). Устранению данного недостатка и посвящена предлагаемая работа.

1. Векторные алгоритмы метода Якоби

Для простоты рассмотрим одномерное уравнение Лапласа $d^2U/dx^2=0$, где функция U определена на отрезке $0\le x\le 1$, с краевыми условиями $U(0)=0$, $U(1)=1$.

Очевидная замена производной конечной разностью приводит к системе сеточных уравнений $U_{i-1}-2U_i+U_{i+1}=0$ при $2\le i\le N-1$, где N – количество узлов сеточной области. Дискретизация краевых условий примет вид: $U_1=1$ и $U_N=0$.

Классический векторный алгоритм метода Якоби для решения этой системы, записанный на языке фортран (стандарт fortran 90 допускает векторную нотацию), выглядит следующим образом:

```
do t=1,Q  ! цикл по итерациям
   U(2:N-1)=(U(1:N-2)+U(3:N))/2.0  ! в силу трехдиагональности системы
end do
```

где Q – наперед заданное количество итераций. Особенностью алгоритма, кроме векторизации вычислений, является явное использование всего одного одномерного массива для хранения данных.

В ходе практической реализации алгоритма на процессоре Intel Core i5-9400 2,9 Gh, операционной системе Ububtu 18.04.1, с использованием компилятора gfortran 7.4 получены результаты, представленные во втором столбце Таблицы 1. При этом дискретизация сеточной области принималась N=4e8, а число итераций Q=20.

Таблица 1. Сравнение различных алгоритмов метода Якоби

алгоритм	однослойный векторный	однослойный скалярный	двухслойный векторный
длительность расчетов (сек.)	11,10	5,23	3,91
занимаемая память (Гб.)	варьируется от 1,5 до 3,0	1,5	3,0
ускорение	1	2,12	2,84

В третий столбец Таблицы 1 помещены результаты эксперимента со следующим скалярным алгоритмом, в котором не допускается производство векторных операций.

```
do t=1,Q  ! цикл по итерациям
   mem2=U(1) ! сохранение значения "слева"
   do k=2,N-1
      mem1=U(k)  ! сохранения текущего значения
      U(k)=(mem2+U(k+1))/2.0  ! вычисления по основной формуле метода
      mem2=mem1  ! перезапись сохраненных значений
   end do
end do
```

Для авторов настоящей работы явилось некоторой неожиданностью (ведь векторизация должна ускорять вычисления!) увеличение, как длительности вычислений, так и объема занимаемой памяти при переходе от скалярного алгоритма к векторному. Причиной тому является автоматическая организация работы с памятью в однослойном векторном алгоритме, направленная на предотвращение преждевременного затирания данных, вследствие чего ее

объем постоянно варьируется в ходе вычислений, достигая двукратного увеличения от ожидаемого. Такая же организация в скалярном алгоритме, проведенная вручную, позволила ограничиться расчетным объемом и, как следствие, уменьшением коммуникационных издержек между различными иерархическими уровнями памяти ЭВМ, по сравнению с векторным алгоритмом. Очевидно, упомянутые издержки являются в рассматриваемом случае фактором, полностью определяющим время расчетов, в силу чего векторизация арифметических операций не привела к сокращению общей длительности вычислений.

Стремясь сохранить векторизацию и упорядочить работу с памятью, авторы смирились с двукратным ростом ее объема, обратившись к ручному регулированию записи одних данных поверх других в следующем двухслойном алгоритме.

```
do t=1,Q,2  ! цикл по итерациям через одну
   U1(2:N-1)=(U2(1:N-2)+U2(3:N))/2.0  ! первое итерационное выражение
   U2(2:N-1)=(U1(1:N-2)+U1(3:N))/2.0  ! второе итерационное выражение
end do
```

Таким образом, в массиве U1 хранятся приближенные значения на нечетных итерациях, в U2 – на четных.

Оценивая экспериментальные результаты для нового алгоритма (четвертый столбец Таблицы 1) наблюдаем его превосходство над ранее представленными. По сравнению с однослойным векторным работа с памятью стала предсказуемой, в силу чего длительность вычислений сократилась почти втрое. Выигрыш от векторизации (в одном векторном регистре размещаются 4 числа одинарной точности) перевесил упорядоченные теперь коммуникационные издержки и вычисления по авторскому двухслойному алгоритму производятся в 1,34 раза быстрее, чем по скалярному.

2. Блочный алгоритм метода Якоби

Обращаясь к классическому распределению значений приближенных решений в методе Якоби на блоки [3], выберем разбиение пространства итераций (построенного по скалярному алгоритму) в виде параллелограммов (центральные блоки) и трапеций (краевые). Звездочками на следующем рисунке обозначены границы четырех блоков, цифрами – номера массивов в двухслойном векторном алгоритме. В отличие от алгоритма из [3] здесь подлежат хранению лишь два итерационных приближения, что существенно сокращает объем занимаемой памяти, а следовательно и длительность расчетов.

```
2  2* 2  2  2  2  2* 2  2  2  2  2* 2  2  2  2  2
1  1  1* 1  1  1  1  1* 1  1  1  1  1* 1  1  1  1
2  2  2  2* 2  2  2  2  2* 2  2  2  2  2* 2  2  2
1  1  1  1  1* 1  1  1  1  1* 1  1  1  1  1* 1  1
```

Рис. 1. Пример разбиения на блоки четырех итераций при N=17, dh=4 и dl=5

Перепишем двухслойный алгоритм в следующем блочном виде, где dh – высота блока, dl - ширина.

```
do t=1,Q,dh                ! переход по блочным слоям
 wl=2; wr=dl+1;  ! работа с крайним левым блоком
 do k=2,dh,2       ! вверх по итерациям
  U1(wl:wr)=(U2(wl-1:wr-1)+U2(wl+1:wr+1))/2.0; wr=wr-1; ! нечетная итерация
  U2(wl:wr)=(U1(wl-1:wr-1)+U1(wl+1:wr+1))/2.0; wr=wr-1; ! четная итерация
 end do
 do g=2,p-1 ! проход по центральным блокам
  wl=(g-1)*dl+2; wr=g*dl+1; ! определение границ нижней стороны блока
  do k=2,dh,2 ! вверх по итерациям
   U1(wl:wr)=(U2(wl-1:wr-1)+U2(wl+1:wr+1))/2.0; wl=wl-1; wr=wr-1;
   U2(wl:wr)=(U1(wl-1:wr-1)+U1(wl+1:wr+1))/2.0; wl=wl-1; wr=wr-1;
  end do;  end do
 wl=(p-1)*dl+2; wr=N-1;  ! работа с крайним правым блоком
 do k=2,dh,2 ! вверх по итерациям
  U1(wl:wr)=(U2(wl-1:wr-1)+U2(wl+1:wr+1))/2.0; wl=wl-1; ! нечентая итерация
  U2(wl:wr)=(U1(wl-1:wr-1)+U1(wl+1:wr+1))/2.0; wl=wl-1; ! четная итерация
 end do; end do
```

Результаты его исследования для dh=Q (один блочный слой) представлены в Таблице 2, где под ускорением понималось отношение длительности работы векторного двухслойного алгоритма к блочному при выбранной ширине блока.

Таблица 2. Исследование эффективности блочного алгоритма

dl	длительность вычислений	объем блока	ускорение
20	3,88 сек.	160 байт	1,01
200	2,09 сек.	1,56 килобайт	1,87
2000	1,94 сек.	15,63 килобайт	2,02
20000	2,08 сек.	156,25 килобайт	1,89
200000	2,33 сек.	1,53 мегабайт	1,69
2000000	3,26 сек.	15,26 мегабайт	1,20
20000000	3,85 сек.	152,59 мегабайт	1,02

Для используемого процессора объем кэш-памяти L1 - 64 килобайт, L2 - 256 килобайт, L3 - 9 мегабайт. В случае небольших значений dl (dl=20, когда алгоритм незначительно отличается от неблочного) и крупных dl (dl=20000000, когда блок значительно не умещается в кэш-память процессора) ускорения почти нет. Наилучшее ускорение (2,02) зафиксировано для блока еще целиком помещающегося в L1. Далее ускорение только падает: когда блок выходит за L1 (1,89), когда блок выходит за L2 (1,69) и когда блок выходит за L3 (1,20). То есть применение разработанного алгоритма действительно позволяет учесть иерархическую структуру памяти ЭВМ.

Благодарности

Работа выполнена при поддержке гранта РФФИ 19-07-00423 А.

Заключение

Разработанный блочный алгоритм метода Якоби позволяет многократно (до 6 раз в выбранном примере) ускорить вычисления по сравнению с классическим векторным на той же программно-аппаратной базе. Далее на его основе имеет смысл разрабатывать параллельные алгоритмы метода Якоби, которые по мнению авторов окажутся более эффективными, чем известные из [1-4].

Литература

1. Ортега, Дж. Введение в параллельные и векторные методы решения линейных систем / Дж. Ортега. – пер. с англ. – М.: Мир, 1991. – 368 с.
2. Xing, Zhou Tiling Optimizations For Stencil Computations // ph.d. thesis of the University of Illinois at Urbana-Champaign, 2013. – 133 p.
3. Alias, Christophe Bogdan Pasca, Alexandru Plesco Automatic Generation of FPGA-Specific Pipelined Accelerators / Christophe Alias, Bogdan Pasca, Alexandru Plesco // Reconfigurable Computing: Architectures, Tools and Applications: 7th International Symposium, ARC 2011, Belfast, UK, March 23-25, 2011. – p. 398-411.
4. Li, Yanhua HW/SW co-optimization for stencil computation: Beginning with a customizable core / Yanhua Li, Zhang, Y., Weiming Zheng // Tsinghua Science and Technology, 21(5), 2016. – p. 570–580.

О.К. Головнин, А.А. Мызников

ЦИФРОВЫЕ РЕШЕНИЯ ДЛЯ ПОСТРОЕНИЯ ТЕМПОРАЛЬНЫХ БАЗ БОЛЬШИХ ДАННЫХ

(Самарский университет, Университет ИТМО)

Повсеместно осуществляемая цифровизация социально-экономических и технологических процессов приводит к необходимости хранения в базах данных (БД) большого объема данных (Big Data), имеющих временную привязку и обладающих временем жизни [1, 2]. Распространенное решение – использование моделей темпоральных данных, в которые интегрируется атрибутно-временной состав, что обеспечивает возможность получения данных, актуальных на определенный момент или в определенный период времени. Проблемами исследования темпоральных моделей и БД занимались многие отечественные и зарубежные исследователи. В настоящей работе проводится аналитический обзор решений, направленных на решение задачи обеспечения темпоральности с использованием реляционной системы управления БД (СУБД).

Модель темпоральных данных строится из элементов данных и внутренних структур, отражающих изменения элементов модели во времени и